Control theory applied to a re-entrant manufacturing line

B. Rem

SE 420343

Master's Thesis

Supervisor: Prof.dr.ir. J.E. Rooda
Coach: Dr.ir. A.A.J. Lefeber

EINDHOVEN UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING
SYSTEMS ENGINEERING GROUP

Eindhoven, April 2003

Preface

This master's thesis is a result of a final assignment of a five years curriculum for becoming a Master of Science in Mechanical Engineering at Eindhoven University of Technology. The final assignment is performed at the Systems Engineering Group, and covers a period of one year.

The Systems Engineering Group aims at developing methods, techniques, and tools for the design of advanced industrial systems. This research project is performed within the Optimization and Control research theme. Research at the Optimization and Control theme engages in developing controller designs and optimization tools starting from computer models of manufacturing systems and machines. These tools should enable an effective and systematic decision making for the design and control of advanced industrial systems. The objective of this research is to investigate a control framework that opens up the possibility to control manufacturing systems by using control theory.

First of all, I would like to thank my coach, Erjen Lefeber, without whom this thesis would not have realized its current form. Gratitude also goes out to professor Rooda for his interest in my research. Further, I would like to thank my fellow students at the Systems Engineering Group for providing a pleasant work environment, and useful input for my research project. Furthermore, I would like to thank my family, friends, and girlfriend for their support during my graduation.

Bart Rem
Eindhoven, April 2003

Summary

Currently, control of manufacturing systems is mostly performed by heuristic control methods such as push, pull, conwip, or hybrid strategies. Although heuristic control methods can in many cases be sufficient, their selection and design remains largely a process of trial and error. On the other hand, in mechanical and electrical engineering, controllers are usually designed using control theory such as pole placement, optimal control, or model predictive control. Since control theory is a well developed field of science, it forms an appealing alternative to the heuristic approach. Consequently, a novel control framework is designed that opens up the possibility to control manufacturing systems using control theory.

The control framework states the following. Derive a model for a manufacturing system—referred to as the approximation model—to which standard control theory can be applied. Next, design a controller for the approximation model. The controller and the approximation model are uncoupled, and the controller is coupled to the manufacturing system—represented by a discrete event model—via two conversion steps. The first conversion step translates the controller output into events for the discrete event model, the second conversion step translates the output of the discrete event model into an input for the controller. Summarizing, the control framework consists of the following elements: (i) discrete event model (manufacturing system), (ii) approximation model, (iii) controller, and (iv) two conversion steps.

This thesis investigates two approaches to design the elements of the proposed control framework. Pole placement and model predictive control (MPC) are used to control a four step re-entrant manufacturing line with batching based on a model by [Dia00].

For the first approach all elements are chosen as modest as possible. The approximation model is a fluid model in state-space form $\dot{x} = Bu$. The controller is designed using pole placement. The controller output is coupled to the discrete event model via a conversion algorithm. This algorithm translates a continuous control signal $u(k)$ into events, by comparing the desired production $\int u(k)$ and the actual production. The simulation results of the first approach are encouraging. Pole placement can be used to control the manufacturing system output towards a setpoint step change. For low utilizations the buffer levels and control signals display periodic patterns. When the utilization is increased from 80% until 96%, these periodic patterns gradually disappear. A point for future research is integrator windup in the conversion algorithm that occurs for unrealistic combinations of reference wip and utilization.

For the second approach pole placement is exchanged for MPC, and the fluid model is extended to include the mean process times of lots as pure time delay. Further, the same conversion algorithm is used. It is demonstrated that MPC can also control the system output towards a setpoint step change. Even for time varying reference signals the

MPC controller still displays promising tracking qualities. Further, transient behavior is differently effected by the MPC tuning parameters for the discrete event model controlled by MPC, than for the approximation model controlled by MPC. Besides, the steady state behavior is only effected for high utilizations by the MPC tuning parameters. It is shown that a conflict involving the sample time arises when the approximation model with delays is made discrete. Due to the sample time conflict, and the fact that the time delays do not influence the steady state performance of the discrete event model controlled by MPC, the time delays are removed from the approximation model. Furthermore, the use of hard constraints on the control signal in the form of input constraints remains questionable.

Comparing pole placement to MPC demonstrates that for low utilization $(0.5 < \rho < 0.8)$ both controllers result in the same steady state performance measured in flow time and wip, since for these utilizations both controllers achieve the reference signal y_{ref} which fixes two of three variables in Little's equation. Besides, for high utilizations $(0.8 < \rho < 1.0)$ MPC obtains lower flow times than pole placement. Overall the difference between pole placement and MPC measured by steady state performance of the controlled discrete event model is small. A clearer distinction between the performance of both controllers might be obtained when instead of steady state behavior, transient behavior is compared. It is suspected that the strength of the control framework partly lies in the field of optimizing transient behavior.

Concluding, the proposed control framework shows that the four step re-entrant manufacturing line can be controlled using pole placement and model predictive control. Exploring the boundaries of the investigated approaches, brought interesting areas of future research to the attention.

Samenvatting (In Dutch)

Het besturen van fabricagesystemen gebeurt hedendaags veelal door gebruik te maken van vuistregels, ook wel heuristische besturingsmethoden genoemd, zoals *push*, *pull*, *conwip* of hybride combinaties hiervan. Alhoewel heuristische besturingsmethoden in veel gevallen kunnen volstaan, blijft hun selectie en ontwerp overwegend een kwestie van *trial and error* en ervaring. Daarentegen worden besturingen voor mechanische, elektrische, en chemische toepassingen, vaak ontworpen door gebruik te maken van gereedschappen uit de regeltechniek. Aangezien regeltechniek een goed ontwikkeld vakgebied is, vormt het een aantrekkelijk alternatief voor de genoemde heuristische aanpak. Zodoende is er een nieuw raamwerk ontwikkeld dat de mogelijkheid creëert om regeltechniek te gebruiken voor het regelen van fabricagesystemen.

Het raamwerk bestaat uit de volgende stappen. Ontwerp allereerst een benaderingsmodel voor het fabricagesysteem waarop standaard regeltechniek kan worden toegepast. Ontwerp vervolgens een regelaar voor het benaderingsmodel. Ontkoppel daarna de regelaar en het benaderingsmodel, en koppel het fabricagesysteem—beschreven door een *discrete event* model—vast aan de regelaar met behulp van twee conversiestappen. De eerste conversiestap vertaalt de uitgang van de regelaar naar *events*, de tweede conversiestap vertaalt de uitgang van het *discrete event* model naar een ingang voor de regelaar. Samengevat bestaat het raamwerk uit de volgende elementen: (i) *discrete event* model (fabricagesysteem), (ii) benaderingsmodel, (iii) regelaar, en (iv) twee conversiestappen.

Dit afstudeerverslag onderzoekt twee methoden voor het ontwerpen van de elementen van het raamwerk. Poolplaatsing en *model predictive control* MPC worden toegepast voor het regelen van een vierstaps *re-entrant* fabricagesysteem gebaseerd op een model van [Dia00].

In de eerste methode zijn alle elementen zo bescheiden mogelijk gekozen. Het benaderingsmodel is een vloeistofmodel met toestandsvergelijking $\dot{x} = Bu$. De regelaar is ontworpen met behulp van poolplaatsing. De uitgang van de regelaar wordt gekoppeld aan het *discrete event* model via een conversiealgoritme. Dit algoritme vertaalt een continu regelsignaal $u(k)$ naar *events*, door de wenselijke productie $\int u(k)$ te vergelijken met de werkelijke productie. De simulatieresultaten van de eerste aanpak zijn bemoedigend. Met poolplaatsing is het mogelijk de uitgang van het fabricagesysteem te regelen. Hiervoor is een stapvormig referentiesignaal gebruikt bij een constante doorzet. Voor een lage bezettingsgraad vertonen de bufferniveaus en regelsignalen periodieke patronen. Wanneer de bezettingsgraad opgevoerd wordt van 80% tot 96% verdwijnen deze periodieke patronen langzaam. Een punt voor toekomstig onderzoek is het voorkomen van *integrator windup* in het conversiealgoritme (of de regelaar) voor onrealistische combinaties van wip en bezettingsgraad.

In de tweede methode is poolplaasting vervangen door MPC, en is het vloeistofmodel uitgebreid met de gemiddelde procestijden van producten in de vorm van Padé benaderde

tijdsvertragingen. Verder wordt hetzelfde conversiealgoritme gebruikt. Het is aangetoond dat naast poolplaatsing ook MPC gebruikt kan worden om de systeemuitgang naar een stapvormige referentiesignaal te regelen. Zelfs voor tijdsvarierende referentiesignalen vertoont de MPC regelaar nog steeds goede volgkwaliteiten. Verder beïnvloeden de MPC afstemmingsparameters, het transiente gedrag van het MPC geregelde *discrete event* model, anders dan het transiente gedrag van het MPC geregelde benaderingsmodel. Het is aangetoond dat, indien het benaderingsmodel gediscretiseerd wordt, er een conflict optreedt omtrent de *sample time*. Wegens dit conflict, alsmede wegens het feit dat de benaderde tijdsvertragingen een niet te onderscheiden invloed hebben op de langlopende prestatie van het *discrete event* model geregeld door MPC, is er voor gekozen de benaderde tijdsvertragingen uit het vloeistofmodel te verwijderen.

De vergelijking tussen poolplaatsing en MPC toont aan dat de langlopende prestatie, gemeten in doorlooptijd en *wip*, voor een lage bezettingsgraad ($0.5 < \rho < 0.8$) hetzelfde is. Dit komt doordat voor een lage bezettingsgraad beide regelaars het gevraagde referentiesignaal kunnen volgen waardoor twee van de drie grootheden in Little's vergelijking vastliggen. Daarnaast behaalt MPC voor een hoge bezettingsgraad een betere prestatie (lagere doorlooptijd) dan poolplaatsing. Over het algemeen is het verschil tussen poolplaatsing en MPC, gemeten in de langlopende prestatie, klein. Een duidelijker verschil tussen beide regelaars is wellicht te zien, als in plaats van het langlopende gedrag, het transiente gedrag beschouwd wordt. Vermoed wordt dat de kracht van het raamwerk zich gedeeltelijk bevindt in het optimaliseren van transient gedrag.

Concluderend, het voorgestelde raamwerk heeft laten zien dat het vierstaps *re-entrant* fabricagesysteem geregeld kan worden met zowel poolplaatsing als MPC. Het verkennen van de grenzen van de bestudeerde methoden, heeft interessante ideeën voor toekomstig onderzoek onder de aandacht gebracht.

Contents

Table of symbols and abbreviations

Symbols

δ	Mean throughput
λ	Mean arrival rate
μ	Mean process rate
ρ	Utilization
ρ_B	Utilization batch machine
ρ_S	Utilization single lot machine
φ	Mean flow time
Γ	Input matrix (discrete state-space) of size $(n \times i)$
Φ	System matrix (discrete state-space) of size $(n \times n)$
$e(k)$	Error between state and reference state at time k
h	Sample time
i	Number of inputs
m	Length of control horizon
n	Number of states
o	Number of outputs
p	Length of prediction horizon
$p(k)$	Desired production at time k
$\bar{p}(k)$	Actual production at time k
t_e	Mean process time
u	Input vector of size $(i \times 1)$
w	Mean wip level
x	State vector of size $(n \times 1)$
y	Output vector of size $(o \times 1)$
A	System matrix (continuous state-space) of size $(n \times n)$
B	Input matrix (continuous state-space) of size $(n \times i)$
C	Output matrix of size $(o \times n)$
$\mathcal{C}_N(B, A)$	Controllability matrix of size $(n \times ni)$
D	Output/Input matrix of size $(o \times i)$
$H(s)$	Transfer function matrix of size $(o \times i)$

N	Pole placement matrix of size $(i \times n)$
$\mathcal{O}_N(C, A)$	Observability matrix of size $(no \times n)$
Q	Setpoint weighting matrix of size $(po \times po)$
R	Input weighting matrix of size $(mi \times mi)$
Y	Prediction matrix of size $(po \times mi)$
K_{MPC}	MPC gain matrix of size $(mi \times po)$

Abbreviations

CT	Continuous time
DE	Discrete event
DT	Discrete time
MPC	Model predictive control

Chapter 1

Introduction

The manufacturing process of integrated circuits (ICs) is a very complex process. On an area no larger than a finger nail, millions of microscopic components are placed. Multiple layers of these components (such as transistors, diodes, and resistors) are carefully positioned on top of each other; layer upon layer forms the three dimensional structure of a chip. The control of the wafer flow through a factory is also complicated. A cause for this complication is the *re-entrant* nature of wafer manufacturing. In a re-entrant system, parts visit the same machine more than once during their production cycle. Re-entrant manufacturing lines are specifically seen in the semiconductor industry; because each layer on a wafer is created one at a time. After a layer is created the wafer leaves the machine to undergo other processing steps whereupon it returns to the same machine to form the next layer. Common wafers exist of about 25 layers [Sem03]. Each layer is created by a lithography[1] machine.

Lithography machines belong to the most expensive machines of a semiconductor factory. With only a few lithography machines, a semiconductor factory processes a large variety of products, each consisting of multiple layers. All these products of different type and stage of processing (age) compete for capacity of the same machine. Since the machine only has a finite capacity, choices should be made about which product of which age to process next. These choices form the essence of a scheduling or control strategy. The goal of a control strategy is to improve performance measures of the system. In manufacturing systems, performance is usually measured by throughput, cycle time or cycle time variation. A control strategy should roughly answer two questions: (i) when a machine becomes available, which product of which age should be processed next, (ii) at which moment should a new product be released into the line? The first question concerns lot scheduling/sequencing, while the second question concerns input control [Wei88].

Scheduling policies or control strategies for re-rentant lines have attracted increasing attention since the 1980's. This short introduction does not attempt to provide a comprehensive survey on the history of controlling semiconductor manufacturing lines. What is intended, is to show that manufacturing lines are usually controlled by sets of priority rules, sometimes referred to as heuristics. It is possible to design heuristics that under given circumstances perform well. However, designing a good heuristic remains mainly a case of experience, trial and error, and rules of thumb. Numerous heuristic approaches have been

[1]Lithography is the process where desired patterns are created on the surface of a wafer by exposing a photosensitive layer to light. Detailed information on the lithography process can be found in [Gro96]

designed, and lots of hybrid combinations are possible. Common heuristic control methods are for instance: push, pull, shortest expected remaining processing time (SERPT), first buffer first serve (FBFS), last buffer first serve (LBFS), least slack policies, earliest due date (EDD), clear a fraction (CAF), fluctuation smoothing policies for mean cycle time (FSMCT), authorization by CONWIP. Kumar and Kumar [Kum93] [Kum01] examined the performance of most mentioned methods as well as stability issues of the FBFS, LBFS, FSMCT policies. Other stability issues of re-entrant lines were examined by Dai and Weiss [Dai96] as well as Banks and Dai [Ban97]. Wein [Wei88] compared various heuristics on a representative wafer fab model. Our research attempts to pursue similar questions, only approached from a different angle. The objective of this research is discussed next.

1.1 Objective

On the one hand manufacturing systems are usually controlled by well considered heuristic policies. On the other hand in mechanical and electrical engineering, controllers are usually designed using control theory. Recently the idea emerged to use control theory (such as optimal control, pole placement or model predictive control) to control discrete event models of manufacturing systems. The objective of this research is to use control theory to control a discrete event model. Since control theory is a well developed field of science, it forms an appealing alternative for heuristic control. However, control theory cannot be applied directly to a discrete event model. A framework is necessary that opens up the possibility to apply control theory to discrete event models. This framework is discussed in the next section.

1.2 Approach

In order to apply control theory to control discrete event models, a framework is necessary. This framework attempts to unite the discrete, and event driven nature of discrete event models with the continuous nature of control theory. Intuitively, it can be understood that conflicts arise when continuous control techniques are used to control discrete event models. These conflicts arise for instance because the models used to design continuous controllers (e.g. state-space models obtained from differential equations) essentially differ from discrete event models. The exact nature of these conflicts and how they are tackled becomes clear by explaining the elements of the control framework. The remainder of this section, step by step explains the elements of the control framework.

The first step of the control framework concerns modeling manufacturing systems[2] by using models to which standard control theory can be applied. These models are continuous approximation models of manufacturing systems. Figure 1.1(b) displays the first step. Continuous approximation models, or short approximation models, are often composed of differential or difference equations. The approximation model should resemble the discrete event model as close as possible. The approximation model is based on the assumption that material flows through the system can be considered as continuous flows; for instance large amounts of products are processed in a short period of time. Therefore, these sort

[2]manufacturing systems are considered to be discrete event models

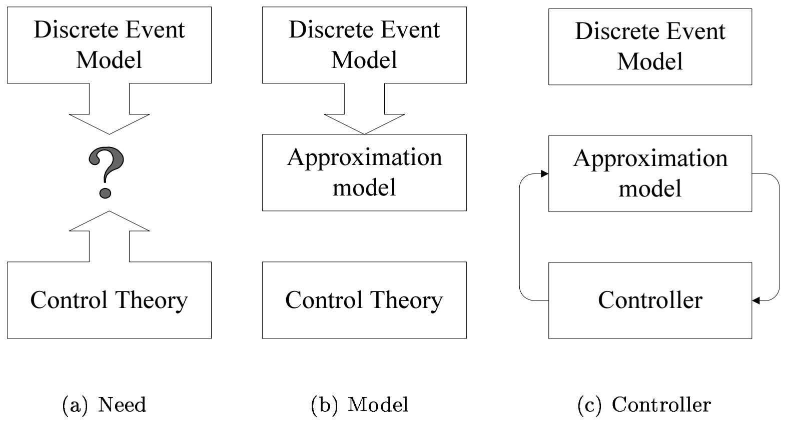


Figure 1.1: Design of approximation model and controller

of continuous approximation models—governed by differential equations—are often considered fluid/flow models. Control theory can be applied to fluid models by writing them as state-space models.

The second step of the control framework is to design a controller for the approximation model using techniques from control theory. Figure 1.1(c) displays the second step. Generally, the controller is some sort of feedback controller. The terms *feedback* and *control* can be defined in the following way [Fra94]: Feedback is the process of measuring the controlled variable and using that information to influence the value of the controlled variable. Control is the process of causing a system variable to conform to some desired value, called a *reference value*. As mentioned before, control theory is a well developed field of science were various established techniques are available. Textbooks like [Fra94], [Pol98] or any other textbook on systems and control theory can be employed.

When the closed loop between controller and approximation model exhibits the desired behavior, the controller is uncoupled from the approximation model and coupled to the discrete event model. The controller, however, was designed for an essentially different model than the discrete event model. A direct coupling between the controller and discrete event model is therefore not possible (Figure 1.2(a)). Consider for instance the output of the controller; this output should become the input of the discrete event model. However, the input of a discrete event model is not as clearly defined—and surely not the same—as the input of the approximation model. A discrete event model is event driven, which means that state changes are invoked by events. An input for a discrete event model could therefore be a series of events. A series of events is essentially different than a continuously varying signal, such as the output of the controller. This argues for some sort of event generator that can translate a continuous signal into events for the discrete event model. A similar argument holds for the output of the discrete event model and the input of the controller.

According to the previous paragraph, two conversion steps are necessary to account for conflicts between the continuous controller versus the discrete event model (Figure 1.2(b)). One conversion between output of the discrete event model and input of the controller, another between output of the controller and input of the discrete event model. The first

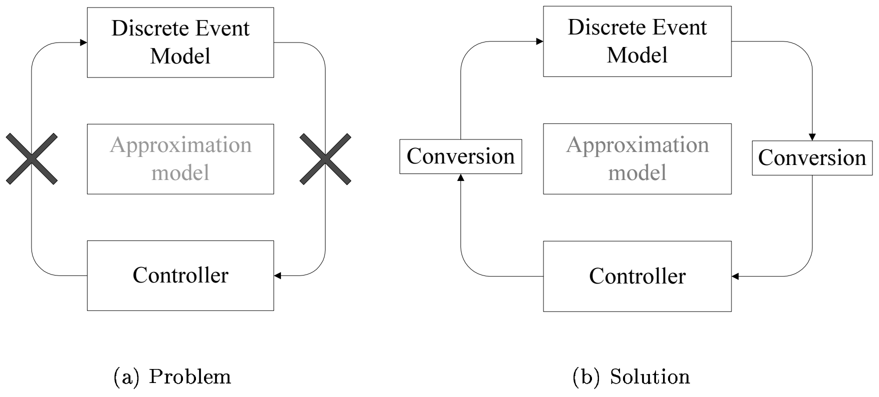


Figure 1.2: Conversion steps are necessary

conversion consists of observing the discrete output and translating it into a continuous signal. The second conversion consists of translating a continuous control signal into events for the discrete event model. For instance, when a machine becomes available it should receive a new assignment. This assignment results from an appropriate translation of the controller output.

The control framework, described above, broadly concerns two research topics. The first topic is modeling manufacturing systems using models to which control theory can be applied. The second topic is controlling manufacturing systems by designing controllers for manufacturing system models, and coupling these controllers to the original manufacturing system. Two approaches to both topics are investigated in this thesis. The following section presents the structure of this thesis.

1.3 Structure

The previous section presents the control framework studied in this research. The framework opens up the possibility to apply control theory to control discrete event models. It is pointed out that the framework consists of different elements that basically concern the modeling and control of manufacturing systems. Two approaches to design the elements of the framework are investigated. This thesis presents the interpretation (i.e. design, simulation, analysis, comparison) of the elements of the discussed control framework. The remainder of this thesis is organized in the following way.

Chapter 2 presents the manufacturing system that forms the case considered in this research. The manufacturing system possesses typical features of a semiconductor factory. Naturally, any other manufacturing system could also be considered. Further, Chapter 2 discusses the performance measures. Chapter 3 presents the first approach to design the elements of the control framework. The chapter starts by adopting a model classification, followed by a general explanation of discrete event models. Then, Section 3.3 presents the first approach to model manufacturing systems; the approach uses a fluid model in the form of a state-space model. In Section 3.4 a controller is designed for the state-space model using pole placement. Through the conversion steps, explained in Section 3.5, the controller is coupled to the discrete event model. The closed loop system is simulated, and the results are discussed in Section 3.6. Chapter 3 is concluded by a discussion that gives rise to two extensions to the first approach of the control framework. Chapter 4 takes these alterations into consideration, and presents the second approach to design the elements of the control framework. The first alteration concerns the fluid model, Section 4.1 presents an approach to design state-space models that incorporate time delays. The second alterations concerns the controller design, Section 4.2 explains how Model Predictive Control (MPC) can be used to control the discrete event model. The new closed loop system—discrete event model controlled by MPC—is simulated and discussed in Section 4.3. Chapter 5 contains the conclusions and a discussion of the presented material in this thesis. Finally, recommendations for future research are provided at the end of Chapter 5.

Chapter 2

Case Description

This chapter presents the manufacturing system considered in this thesis. The manufacturing system—a four step re-entrant flow line with batching—is based on a model by [Dia00], and previous work [Rem03]. [Dia00] modeled each step in the manufacturing system as constant vector fields (fluid model), that, when observed at fixed events, reduced to a set of piecewise linear maps. For these maps, the existence of periodic or eventually periodic orbits was investigated.

After the case is presented, this chapter continuous by explaining the performance measures used to quantify the performance of the controller. Finally, the utilization of the machines—an important quantity for stochastic systems—are determined for the considered case.

2.1 Case: re-entrant flow line with batching

This section presents the case considered in this research. The case is based on a model used by [Dia00], and concerns a manufacturing system that possesses typical features of a semiconductor manufacturing system. A short overview of the semiconductor manufacturing process clarifies these features. Detailed information on the process of semiconductor fabrication and the industrial jargon can be found on various websites [Sem03], [Int03], or in [Gro96].

A semiconductor factory produces chips or integrated circuits onto discs of silicon. These discs are called wafers. Wafers are grouped together in lots. A lot contains about 25 wafers. Modern wafers have a diameter of 300mm and can contain hundreds of chips. Wafers consist of multiple layers stacked on top of each other. The layers are created one at a time by a machine usually referred to as a lithography tool (e.g. the TWINSCAN). After each layer is created, the wafer is processed by a number of different steps. Once these steps are completed, the wafer returns to the lithography tool for the second layer. The fact that wafers experience multiple processing steps by the same machine makes the production of wafers a *re-entrant* process.

Besides re-entrant material flows, semiconductor lines display another typical feature. At some stages in the wafer fabrication process, an isolating layer of silicon dioxide needs to be formed on the wafer surface. This is done by heating the wafers in a furnace. The furnace processes groups of lots at once. A group of lots is referred to as a batch. Therefore, the furnace is called a *batch machine*. On the other hand, a machine that processes one lot at a time, is called a *single lot machine*. The number of lots in a batch is called the *batch size*. The batch size is important for the performance of the system.

The above explanation of the semiconductor manufacturing process displays two typical features that are included in the case.

1. Products are processed by two machines: a batch machine and a single lot machine.
2. The production cycle contains a re-entrant step.

Furthermore, the machines process products of two different types: type a and type b. The production cycle or production route of products is depicted in Figure 2.1. In Figure 2.1, M_B denotes the batch machine, and M_S denotes the single lot machine. The manufacturing system is a four step re-entrant flow line, therefore, both products undergo four process steps. First, a step at the batch machine; second, a step at the single lot machine; third, a step where products are fed back to the batch machine; fourth, another step at the single lot machine. After the fourth step, products are completed and leave the system.

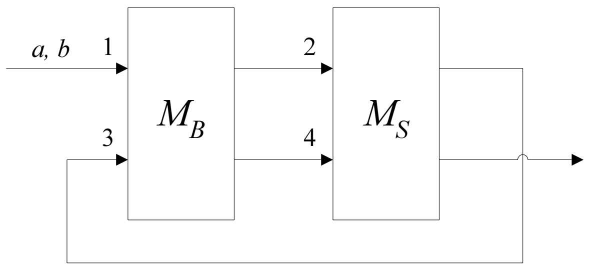


Figure 2.1: Production route of part types a and b

Figure 2.1 shows that two different product types of two different ages are in queue before each machine. Products of different type and age are stored separately in type and age specific buffers. Therefore, each machine is preceded by four buffers that contain lots waiting to be processed by that machine. This leads to the system representation shown in Figure 2.2. Here B_i with $i \in \{1, 2, \ldots, 8\}$ represent buffers; G_j and E_j with $j \in \{a, b\}$ denote respectively, the generator and the exit process of parts of type j. The buffers sequence—the buffers a lot sequentially visits during a production cycle—of both parts is given in Table 2.1.

Type		1		2		3		4	
a	B_1	$\rightarrow$	B_5	$\rightarrow$	B_3	$\rightarrow$	B_7	$\rightarrow$	E_a
b	B_2	$\rightarrow$	B_6	$\rightarrow$	B_4	$\rightarrow$	B_8	$\rightarrow$	E_b

Table 2.1: Buffer sequence of parts a and b for process step 1 to 4

Additionally, the following assumptions are made:

- All buffers are infinite buffers.

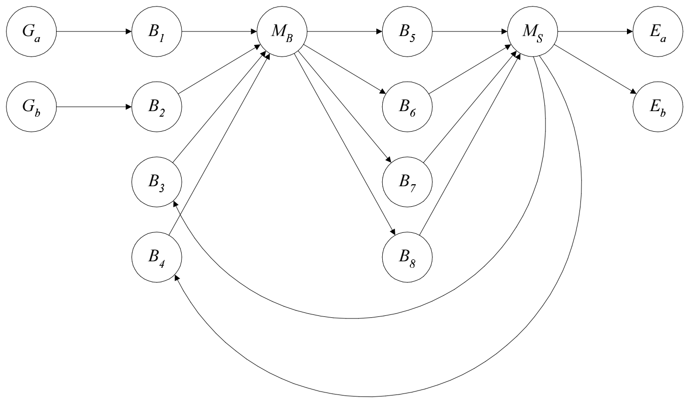


Figure 2.2: Re-entrant flow line, two machines process parts a and b

- A machine can only work on one product of a certain age and type at a time.
- The furnace runs batches of size three.
- Machines never break down and an operation always succeeds.
- Transfer times and setup times of lots are assumed to be negligibly small in comparison to the process times of lots.
- Process times of lots are stochastic and can be arbitrarily distributed. In this case they are chosen to be exponentially distributed.
- The mean process times are denoted by t_{ei} with $i \in \{1, 2, \ldots, 8\}$ and are given in the following table. Table 2.2 shows that the total process time of both parts is equal to

Type	1	2	3	4
	t_{e1}	t_{e5}	t_{e3}	t_{e7}
a	2/3	1/5	1/3	2/15
	t_{e2}	t_{e6}	t_{e4}	t_{e8}
b	1/3	2/15	2/3	1/5

Table 2.2: Process times of parts a and b for process step 1 to 4

$2/3 + 1/5 + 1/3 + 2/15 = 4/3$. Note that, the mean process time of the first two steps of part a are equal to those of the last two steps of part b; and that, the mean process time of the first two steps of part b are equal to those of the last two steps of part a.

While the properties outlined in the case above, are outside our control, several decisions are still at our disposal. These decisions are controllable inputs, and are discussed next.

Controllable inputs

Up to now, the properties that are outside our control were given. The case description leaves questions concerning the control of the re-entrant flow line unanswered. By adding a controller, products in the line start to flow. A controller for the re-entrant flow line can be seen as a number of sequencing/scheduling rules that regulate the flow of material through the system. Basically a controller should answer the following questions:

1. When a machine becomes available, should that machine remain available or start processing a new product?
2. When a machine becomes available and a new product should be processed, which of several products waiting before that machine should be processed next?
3. At what moment should a new product be inserted into the system?

Numerous different control strategies that answer these questions exist. Some typical heuristic control methods for re-entrant manufacturing lines were mentioned in Chapter 1.

2.2 Performance measures

The goal of a control strategy is to increase the performance of a system. The performance of a system is measured by certain performance measures. Performance measures create grounds to compare different control strategies on. A performance measure should be unbiased and represent a meaningful quantity. The choice of performance measure is essential to make a well-founded decision which strategy to employ.

The most basic quantities in a manufacturing system are flow time [time], throughput [lots/time], and work-in-process (wip) [lots]. Flow time of a lot is the time it takes that lot to get from one point to another; in manufacturing lines consisting of buffers and machines, the flow time is equal to the process times added with the time spend in buffers waiting to be processed. The throughput is equal to the amount of lots completed per unit time. The wip is equal to the amount of lots present in the system at a certain time. The well-known relation between mean flow time, mean throughput, and mean wip is provided by Little [Lit61]. This relation states that the mean wip w is equal to the mean throughput δ times the mean flow time φ[1]

$$w = \delta \cdot \varphi. \tag{2.1}$$

Little's law is valid in a steady state situation (long term behavior), and holds for all production lines. In the steady state situation the throughput of lots δ is equal at any cross-section of the production line.

In manufacturing systems, performance is often measured by one of the three variables related by Little's law. This is done by keeping one variable constant and maximizing or minimizing another variable. For instance, if the performance measure of a manufacturing system is the flow time; then the goal of a control strategy can be to minimize the flow time using a constant amount of wip; which through (2.1) results in maximizing the throughput. Equivalently, the throughput can be kept constant and while the goal of the control strategy is to minimize the wip; which again through (2.1) results in minimizing flow time.

The performance measure used in this research is the mean flow time. The goal of the control strategy is to minimize the mean flow time while maintaining the same level of throughput. As discussed above, this strategy results in a minimal amount of wip.

[1] Unless explicitly stated otherwise, this text considers wip, throughput and flow time to be mean values.

2.3 Utilization

The utilization of a machine is a typical measure for a production system; it represents a measure for the load of a system. The utilization can easily be explained as follows. Consider a machine to be either busy or idle. Then, the utilization of a machine is the fraction of time that machine is busy [Hop00]:

$$\text{utilization} = \frac{\text{busy time}}{\text{busy time} + \text{idle time}}\ .$$

Intuitively, it is noticed that if the rate at which material is inserted into a system is larger than the process rate of the machines, the system is overloaded and queue lengths grow to infinity. This situation should be avoided and corresponds to the situation where the utilization is greater than or equal to one. The utilization ρ of a single machine is defined as follows

$$\rho = \frac{\lambda}{\mu}, \tag{2.2}$$

with λ [lots/time] the arrival rate of lots and $\mu = 1/t_e$ [lots/time] the process rate of a machine (recall that t_e is the process time). To assure a stable system, the assumption is made that the machines are able to handle the amount of work that is inserted into the system. In other words, the utilization of the machines should be less than 1

$$\rho < 1. \tag{2.3}$$

In the considered case, described in Section 2.1, the manufacturing system consists of a batch and a single lot machine. Each machine performs two operations on two parts (a and b). Therefore, each machine handles four queues of work. Furthermore, the batch machine processes batches of lots. Denote the batch size by k. Let λ_j be the arrival rate of part j, with $j \in \{a, b\}$. Also, assume each process step is performed with a mean process rate $\mu_i = 1/t_{ei}$, with $i \in \{1, 2, \ldots, 8\}$. Then, the utilization of the batch machine ρ_B, and the utilization of the single lot machine ρ_S, can be determined as follows

$$\begin{aligned} \rho_B &= \frac{1}{k}\left(\frac{\lambda_a}{\mu_1} + \frac{\lambda_b}{\mu_2} + \frac{\lambda_a}{\mu_3} + \frac{\lambda_b}{\mu_4}\right) < 1 \\ \rho_S &= \frac{\lambda_a}{\mu_5} + \frac{\lambda_b}{\mu_6} + \frac{\lambda_a}{\mu_7} + \frac{\lambda_b}{\mu_8} < 1\ . \end{aligned} \tag{2.4}$$

Utilization and Little's steady state performance measures are used during the simulations to quantify differences between control strategies. Summarizing the material presented in this chapter: the considered case, a four step re-entrant flow line with batching, is explained; performance according to Little is explained, and the utilization of the machines is determined. The next chapter presents the first approach to design the elements of the control framework is presented.

Chapter 3

Fluid Model Controlled by Pole Placement

This chapter presents and discusses the first approach to the control framework that creates the possibility to apply control theory to control discrete event models. The control framework, outlined in Chapter 1, consists of a number of elements; these elements are step by step presented in this chapter. When all elements are presented, simulations are performed of the approximation model together with controller, followed by the discrete event model together with controller coupled through conversion steps.

As a first approach all elements are chosen as modest as possible. The approximation model is a fluid model in state-space form, the controller is designed using pole placement, and conversion is performed only on one side of the framework. In the next chapter, some neglected effects are added to the first approach, which leads to the second approach. The current chapter starts by adopting a model classification that pinpoints closer why and where problems arise that the control framework attempts to solve.

3.1 Model classification

In this section a model classification is adopted from [Zei00] [Bee00]. This widely used classification places the control framework, discussed in Chapter 1, in a more specific setting. It is used to show that conflicts arise between manufacturing systems represented by discrete event models and continuous control techniques. Three different model classes can be distinguished: discrete event (DE), continuous time (CT) and discrete time (DT).

In DE models, the state x can take on countably many values. The state changes at certain time points, these time points (not necessarily equidistant) are the times events take place. In between two adjacent points the state remains the same. DE models are event driven, which means that state changes are invoked by events. If no events occur, the model is at rest.

In CT models, the state changes continuously while time evolves. CT models can be represented by a dynamical system consisting of differential equations. Here $\dot{x}$ is the time derivative of the state, which can be a function of x, an input vector u and the time t

$$\dot{x} = f(x, u, t). \tag{3.1}$$

In DT models, the state changes at equidistant time points only. In between two points, the state remains the same. DT models can be represented by difference equations. Difference equations are the discrete equivalent of differential equations. DT models can be described by the following equation (here h denotes the sample size)

$$x(th + h) = f(x(th), u(th), h). \tag{3.2}$$

The state at time $th + h$ is a function of the state at time th, the input u at time th, and the sample size h.

The goal of this research is to apply control techniques designed for DT and CT models to control a class of DE models. The main difficulty encountered with this approach is that DE models and CT/DT models are essentially different models. DE models are driven by events, while CT/DT are driven by an input signal u. The control framework presented in Chapter 1, could through approximation models and adequate conversions steps resolve this difficulty. The discrete event model considered in this thesis is discussed next.

3.2 Discrete event model

This section deals more specifically with the discrete event model of the manufacturing system considered in this thesis. Discrete event models were introduced in the previous section. It was mentioned that, in discrete event models, state changes are invoked by events. Events can be either controllable or uncontrollable. A controllable event is for instance the start up of a new process step; while the end of that process step is an uncontrollable event. The controllable events are basically the result from the decisions taken by a controller. These decisions, or controllable inputs, were discussed in Section 2.1.

A representation of the discrete event model with a discrete event controller is presented in Figure 3.1. A complete representation of the discrete event model is given in Appendix D.1. The model consists of lot generators (G), a series of buffers and machines (B and M), lot collectors (E), and a control process (C). Generators insert lots into the line. Lots may be either generated uncontrolled, with a stochastic or deterministic arrival rate; or lots may be generated controlled, by receiving authorization from a control process. Buffers receive lots or batches of lots over one channel and send, when requested by the downstream process, lots or batches of lots over another channel. Machines, when authorized by a control process, receive lots or batches of lots over one channel, perform an operation that takes a certain stochastic amount of time (mean process times are given in Table 2.2), and subsequently send processed lots or batches of lots over another channel. At the end of the flow line, an exit process removes finished lots from the line. The exit process is considered as a finished goods buffer. Lots start to flow through the manufacturing line, by authorizations from a control process. The control process should answer the questions mentioned in Section 2.1.

The specification of discrete event models is, in the Systems Engineering Group[1], performed in the formalism χ [Roo00] [Kle01]. The implementation of the discrete event model for the manufacturing system can be found in Appendix D.2.

[1] http://se.wtb.tue.nl

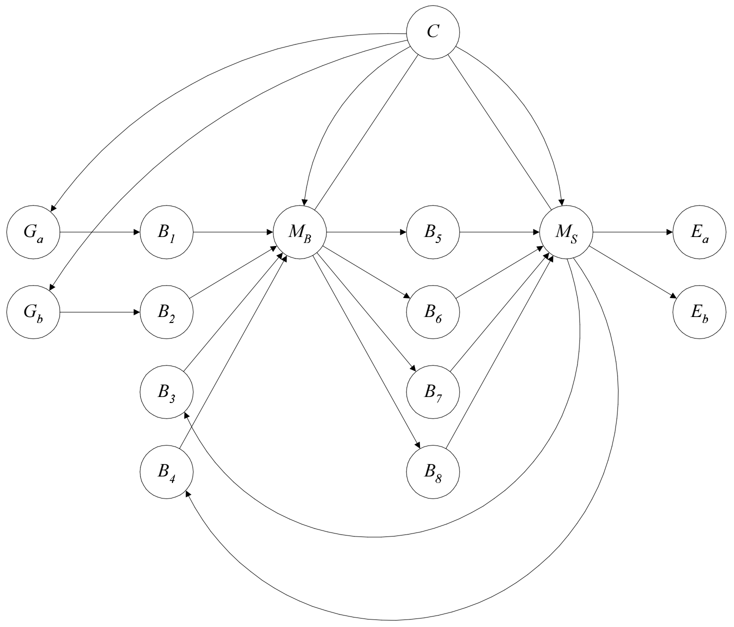


Figure 3.1: Discrete event model with controller

3.3 Fluid approximation model

This section treats the first step of the control framework discussed in Chapter 1. The first step is to derive an approximation model to which standard control theory can be applied. The approximation model should resemble the behavior of the discrete event model as close as possible. Here behavior means the dynamics of the material flow through the manufacturing system. As discussed in the previous section, the manufacturing system can be described by a discrete event model consisting of a series of buffers and machines. In the field of control theory, models are often formulated as state-space models. In this thesis, state-space models—more precisely, discrete-time linear time invariant (LTI) state-space models—are used to approximate discrete event models.

Generally a discrete-time state-space model [Fra94] is of the form

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{3.3a}$$
$$y(k) = Cx(k) + Du(k)\,, \tag{3.3b}$$

where x is an $(n \times 1)$ state vector, u is an $(i \times 1)$ input vector, and y is a $(o \times 1)$ output vector. Consequently, $\Phi \in \mathbb{R}^{n\times n}$, $\Gamma \in \mathbb{R}^{n\times i}$, $C \in \mathbb{R}^{o\times n}$ and $D \in \mathbb{R}^{o\times i}$. Here n denotes the order of the state-space model. Let, throughout this chapter,

$$\begin{aligned} n &= \text{number of states,} \\ i &= \text{number of inputs,} \\ o &= \text{number of outputs.} \end{aligned}$$

The approach to approximate discrete event models with state-space models is done by modeling the change in buffer contents (amount of lots in a buffer) throughout time. The change in buffer contents is equal to the inflow rate minus the outflow rate. Also, what ever leaves one buffer, is inserted into another. Buffers are fed from upstream buffers and feed downstream buffers. In this way a line of buffers feeding each other is obtained.

The following example clarifies the approach described above, by showing how a small flow line can be modeled by a state space model.

Example 3.3.1 *This example shows how a state-space approximation model can be derived from a discrete event model by modeling the change in buffer contents. Consider a basic flow line processing lots (Figure 3.2). The control process is not depicted in the figure, because the state-space model describes material flows, and a controller is later added with techniques from control theory. The flow line consists of a generator (G), two buffers (B_1 and B_2), two machines (M_1 and M_2), and an exit process (E). The generator inserts a lot into the line when told by the controller. The buffers are FIFO buffers. The machines process lot for lot; each lot has a stochastic process time. The exit process receives lots immediately when they finish processing by the second machine. The exit process can be seen as a finished goods buffer.*

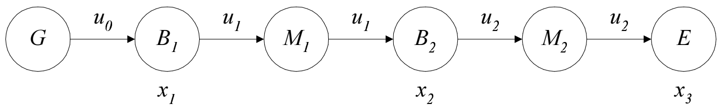


Figure 3.2: A flow line example

In the above figure, u denotes the rate at which lots are removed from one buffer and inserted into another. Further, x denotes the amount of lots present in a buffer. For a discrete time state-space model (3.3), u is the input, x is the state. Let, in this example, the output y be equal to the state: $y = x$.

The state-space approximation model is obtained from the following equations (here h is the sample size):

$$\begin{aligned} x_1(k+1) &= x_1(k) + h(u_0(k) - u_1(k)) \\ x_2(k+1) &= x_2(k) + h(u_1(k) - u_2(k)) \\ x_3(k+1) &= x_3(k) + hu_2(k) \\ y_1(k) &= x_1(k) \\ y_2(k) &= x_2(k) \\ y_3(k) &= x_3(k). \end{aligned} \tag{3.4}$$

Let,

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}, \qquad u(k) = \begin{bmatrix} u_0(k) \\ u_1(k) \\ u_2(k) \end{bmatrix}, \qquad y(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \end{bmatrix}. \tag{3.5}$$

Then, by combining (3.4) *and* (3.5) *the state-space approximation model is given by*

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(k) + h \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(k) \end{aligned} \tag{3.6}$$

Now we have a state-space model to which control theory can be applied. The model is reasonably small; since the state vector has length three, the model is said to be of order three. It should be noted that (3.6) *is not an exact representation but an approximation of the discrete event model depicted in Figure 3.2. Neglected effects are for instance (i) constraints on buffer volumes, (ii) process times of lots, (iii) capacity constraints of the machines. Some effects may be accounted for by the controller; other effects may later be included in the model.*

Fluid model of the re-entrant flow line

After reading Example 3.3.1, one might be left with the question how to deal with the demand. Modeling the demand is closely related to modeling the generator process. The generator process, or the process of releasing new lots in the line, can be either controllable or uncontrollable. If uncontrollable, then lots arrive at the first buffer with a certain arrival rate, and the controller deals with lots as soon as they arrive. If controllable, then the generator can be seen as a buffer that delivers immediately when requested and never runs empty. In the controllable case, demand may be translated into an increasing amount of finished goods. Then, demand is considered to be a reference output, traceable by a controller. In the uncontrollable case, the arrival rate of lots may be explicitly modeled as an uncontrollable reference input that can be included in the approximation model. In order to do so, an error between the state and the reference state is defined, and error dynamics are derived. The state-space model for the re-entrant flow line is derived using an uncontrollable generator process; lots arrive with a deterministic arrival rate in the first buffer. The remainder of this section first derives the system equations, and next derives the error equations.

The system equations of the re-entrant flow line are derived by using the approach of Example 3.3.1. Again, the change in buffer contents is described as the rate at which lots enter the buffer subtracted by the rate at which lots leave the buffer. The inputs and the outputs (outputs are equal to the states) are depicted in Figure 3.3. The system equations are obtained by using the property that buffers feed into each other. The buffer volume is denoted by x, and u denotes the speed at which products leave one buffer and enter another. By connecting buffers, the following equations are obtained for process step one till four of lot type a

$$\begin{aligned} x_1(k+1) &= x_1(k) + h(\lambda_a(k) - u_1(k)) \\ x_5(k+1) &= x_5(k) + h(u_1(k) - u_5(k)) \\ x_3(k+1) &= x_3(k) + h(u_5(k) - u_3(k)) \\ x_7(k+1) &= x_7(k) + h(u_3(k) - u_7(k)) \\ y_i(k) &= x_i(k) \quad \forall\, i; \end{aligned} \tag{3.7}$$

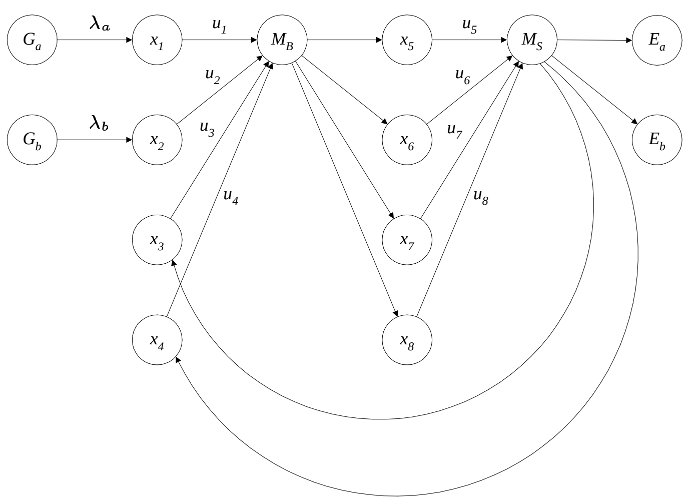


Figure 3.3: Inputs and states of the re-entrant flow line

and for process step one till four of lot type b

$$\begin{aligned} x_2(k+1) &= x_2(k) + h(\lambda_b(k) - u_2(k)) \\ x_6(k+1) &= x_6(k) + h(u_2(k) - u_6(k)) \\ x_4(k+1) &= x_4(k) + h(u_6(k) - u_4(k)) \\ x_8(k+1) &= x_8(k) + h(u_4(k) - u_8(k)) \\ y_i(k) &= x_i(k) \quad \forall\, i. \end{aligned} \tag{3.8}$$

Here λ_a and λ_b are deterministic arrival rates of respectively lot type a, and lot type b. The arrival rate, or throughput demand, is treated as an uncontrollable reference input. This uncontrollable input leads to error dynamics, which are formulated next.

Fluid model of the re-entrant flow line: error equation

This paragraph deals with deriving the state-space approximation model of the re-entrant flow line described by (3.7) and (3.8). As discussed in the paragraph above, demand is modeled as an uncontrollable input signal. In order to compose a state-space model that takes the uncontrollable input into account, an error e is defined. The error is defined as the difference between x and x_{ref}

$$e(k) \triangleq x(k) - x_{\text{ref}}(k)\ . \tag{3.9}$$

Given this error it is possible to derive an error equation. The error equation is derived by splitting the input vector $u(k)$ in two parts: an uncontrollable part $\bar{u}_1(k)$ and a controllable part $\bar{u}_2(k)$. Assume that the uncontrollable input $\bar{u}_1(k)$ is the arrival rate of lots in the first buffers (denoted by λ_j with $j \in \{a, b\}$), and the controllable input $\bar{u}_2(k)$ are the outflow/inflow rates of the buffers (denoted by $u_i(k)$ with $i \in \{1, 2, \ldots, 8\}$). Then, $\bar{u}_1(k)$

can be treated as a reference input as follows. Let,

$$x(k) = \begin{bmatrix} x_1(k) \\ x_5(k) \\ x_3(k) \\ x_7(k) \\ x_2(k) \\ x_6(k) \\ x_4(k) \\ x_8(k) \end{bmatrix}, \qquad \bar{u}_1(k) = \begin{bmatrix} \lambda_a(k) \\ \lambda_b(k) \end{bmatrix}, \qquad \bar{u}_2(k) = \begin{bmatrix} u_1(k) \\ u_5(k) \\ u_3(k) \\ u_7(k) \\ u_2(k) \\ u_6(k) \\ u_2(k) \\ u_8(k) \end{bmatrix}. \tag{3.10}$$

Then, the combining equations (3.7) and (3.8) and splitting $u(k)$ gives the following state-space model

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ &= \Phi x(k) + \Gamma_1 \bar{u}_1(k) + \Gamma_2 \bar{u}_2(k) \\ y(k) &= x(k), \end{aligned} \tag{3.11}$$

where

$$\Gamma_1 = h \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \Gamma_2 = h \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \tag{3.12}$$

and Φ is an $n \times n$ unit matrix.

By noticing that in the equilibrium situation $x(k+1) = x(k)\ \forall\ k$ it follows that

$$\bar{u}_{2\text{ref}}(k) = -\Gamma_2^{-1}(\Gamma_1 \bar{u}_1(k)) \,. \tag{3.13}$$

Given the error

$$e(k) = x(k) - x_{\text{ref}}(k) \,, \tag{3.14}$$

the error dynamics are given by

$$\begin{aligned} e(k+1) &= x(k+1) - x_{\text{ref}}(k+1) \\ &= e(k) + \Gamma_1 \bar{u}_1(k) + \Gamma_2 \bar{u}_2(k) - [\Gamma_1 \bar{u}_{1\text{ref}}(k) + \Gamma_2 \bar{u}_{2\text{ref}}(k)] \\ &= e(k) + \Gamma_2 \left[\bar{u}_2(k) - \bar{u}_{2\text{ref}}(k)\right]. \end{aligned} \tag{3.15}$$

Let

$$\tilde{u}(k) = \bar{u}_2(k) - \bar{u}_{2\text{ref}}(k) \,, \tag{3.16}$$

then the state-space model of the error equation is given by

$$e(k+1) = \Phi e(k) + \Gamma_2 \tilde{u}(k) \,, \tag{3.17}$$

where Φ is an $n \times n$ unit matrix, and Γ_2 is given in (3.12).

This section is concluded by a short revision of the material presented in this section. The revision completes the first step in the control framework, and announces the second step. An approximation model, suitable for control theory, for the re-entrant flow line is devised. The method to derive the approximation model is based on modeling changes in buffer contents and linking buffers together. The design method is clarified by means of an example of a short flow line, which led to a third order state-space model. It is mentioned that demand can be modeled in different ways. An approach is adopted where, demand is considered a reference input signal. Considering demand as a reference input signal is equivalent to a generator inserting lots at a certain rate in the first buffer. In order to include such a demand in a state-space model, an error between x and x_{ref} is defined, which led to error equation (3.17). This error equation, forms the approximation model of the discrete event model. In the following section a controller is designed for (3.17) by using pole placement. The goal of this controller is to drive the error to zero and keep it there, causing the output to become equal to the reference output. Consequently, when the reference output changes, the system output follows.

3.4 Pole placement

In the previous section, an approximation model of the re-entrant flow line was formed. For the approximation model a controller must be designed. The controller together with the approximation model (closed-loop system) should exhibit the desired behavior. A system exhibits the desired behavior when the system outputs follow a certain reference trajectory. The controller, generally a feedback controller, is designed by using techniques from control theory [Fra94]. As a start, pole placement is used to design the controller.

Pole placement is a technique where the poles, or eigenvalues, of a system are placed at certain locations in the complex plane. The pole locations determine the response of the system to input changes, for instance whether trajectories converge or diverge, approach setpoints oscillating or not. In order to ensure a stable system, all poles must have a negative real part for a continuous time system, or lie within the unit circle for a discrete time system. A profound examination of pole locations and resulting responses can be found in Chapter 3 and 8 of [Fra94].

Pole placement is used to design a feedback controller for the multiple input multiple output (MIMO) discrete error equation given by (3.17). The controller should reduce the error (3.9) to zero and keep it zero. This is achieved by multiplying the error with a matrix N given in the following feedback law

$$\tilde{u}(k) = Ne(k) \,, \tag{3.18}$$

where $N \in \mathbb{R}^{i \times n}$. Since, the state-space model is a MIMO instead of a SISO model, analytical pole placement is a little more complicated. An algorithm for pole placement that can be used is provided by [Pol98]: *Algorithm 9.5.1 Pole placement by state feedback.* However, in this thesis the MATLAB® function PLACE is used.

All poles are placed in the origin. This results in a highly damped system with a fast response. A controller with all poles in the origin is referred to as a deadbeat controller. For a deadbeat controller it holds that the error should become zero (or practically zero) within n sample times. Here n is the order of the state-space model. During the simulations this is verified. If all poles are placed in the origin, the feedback matrix N is found to be

(here h denotes the sample size)

$$N = \frac{1}{h}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \frac{1}{h}\begin{bmatrix} N_{aa} & N_{ab} \\ N_{ba} & N_{bb} \end{bmatrix} \tag{3.19}$$

To explain the elegant structure of N recall that the output of one buffer is the input of another buffer, and buffers feed downstream. Also recall that the upper four elements of the vectors $\tilde{u}(k)$ and $e(k)$ represent lot type a and the lower four elements of vectors $\tilde{u}(k)$ and $e(k)$ represent lot type b. Therefore, N can be split into four parts:

- N_{aa} determines type a inputs using type a errors
- N_{ab} determines type a inputs using type b errors
- N_{ba} determines type b inputs using type a errors
- N_{bb} determines type b inputs using type b errors

First the observation made is that N_{ab} and N_{ba} are empty matrices. This is logical, since in the state-space model type a buffers are not linked to type b buffers. Second the observation is made that N_{aa} and N_{bb} are lower triangular matrices. This is understandable, since lots flow downstream, and an error upstream cannot be corrected by modifying an input downstream. The upper and the lower part of the error and input vector are arranged from top/upstream to bottom/downstream buffers. Which results in a lower triangular matrix; the other way round would result in an upper triangular matrix.

Now a feedback matrix that reduces the error to zero exits. The new input, or control action, is simply determined by multiplying the error with the feedback matrix N. This completes the second step of the control framework. The fourth step of the control framework is to, on one side, translate the control output into an input (events) for the discrete event model, and on the other side, translate the output of the discrete event model into an input for the controller. These translations are referred to as conversion steps. The conversion steps are discussed in the following section.

3.5 Conversion algorithms

The fourth, and final step of the control framework (Figure 1.2) is to couple the controller design in the previous section to the discrete event model. Since the controller is designed for an essentially different model than the discrete event model, this coupling can only be achieved through adequate translation algorithms. These translation algorithms, or conversion steps, are discussed in this section. The conversion steps tackle conflicts between the discrete event nature of the manufacturing system and the continuous nature of the controller. Therefore, the conversion between output of DES and input of controller can be seen as a *event to analog* "E/A conversion", and the conversion between output of controller and input of DES can be seen as an *analog to event* "(A/E) conversion".

"E/A conversion"

The digital to analog (E/A) conversion step is performed between output of the discrete event system and input of the controller. Recall that the output of the discrete event model are buffer levels represented by naturals. The conversion step is chosen straightforwardly: let the output of the discrete event system be $\bar{y}(k)$ and let the input of the controller be $y(k)$, then

$$y(k) = \bar{y}(k) \, . \tag{3.20}$$

So no conversion is used. This could cause some chattering in the controller input, which is taken for granted for now. If this turns out to be a problem, filters could be used to smoothen the signal.

"A/E conversion"

The analog to digital (A/E) conversion step is a crucial step in the control framework. Through this conversion, a continuous control signal is translated into events for the discrete event model, in such a way that the intentions of the controller are guaranteed. Besides the translation into events, the conversion algorithm might be hindered by a number of other complications. These complications are for instance unmodeled effects such as: (i) capacity bounds on machines, (ii) constraints on buffer levels, (iii) a machine can only serve one queue at a time, (iv) the output of the discrete event model are naturals (buffer contents) while the output of the state-space models are reals. Above points complicate the timing of events for the discrete event model, and ask for a dynamic scheduling approach. A possible approach is discussed next.

The conversion algorithm should translate the output of the controller into events for the discrete event model. The controller and A/E conversion are depicted in Figure 3.4. The output of the controller is a vector containing flow rates of lots of a certain type and age. These flow rates are considered to be process speeds of the machines. Let this vector be $u(k)$. Since the elements of $u(k)$ are process speeds, the integral of $u(k)$ are cumulative productions. These cumulative productions, denoted by $p(k)$, are called the desired productions. Equivalently, the discrete event model processes products, which results in an actual production. Denote the actual production by $\bar{p}(k)$. The goal of the conversion algorithm is to equalize the desired and actual production of every process step, by generating events.

The conversion algorithm states the following:

> A machine should serve the queue with the largest difference between the desired production and the actual production, provided that this difference is greater than a positive threshold value, and the queue is not empty. If all queues are empty, or all actual productions are greater than the desired ones plus a threshold value, the machine is left idle until another event takes place.

One effect the conversion algorithm could take into consideration is the capacity constraint of machines. The capacity of a machine is physical bounded, since the utilization must be smaller than one (2.4). It is possible to filter the output of the controller before integrating the signal and handing it to the conversion algorithm. The reason for this prefilter is that the control output might not be realistic in the sense that it violates the utilization constraint. Then, the discrete event model might not be able to perform all throughput

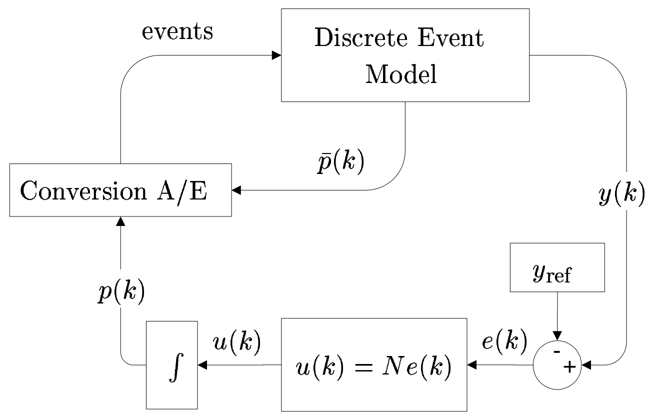


Figure 3.4: Conversion algorithm and controller

requests from the controller. In order to avoid constraint violating inputs, it is possible to saturate the output of the controller; saturation is discussed next.

Saturation

The capacity of a machine is physically bounded. This constraint is not present in the state-space model, nor in the controller. Therefore, control signals might become negative for a long time or unrealistically high. When this situation occurs, one possibility is to saturate the control signal to a realistic value. However, the trends in $u(k)$ should not be affected to much by such a saturation, else the control intentions are lost. Generally a saturation function is given by

$$u_{\text{sat}}(k) = \begin{cases} u_{\max} & \text{for } u(k) \geq u_{\max} \\ u(k) & \text{for } u_{\min} < u(k) < u_{\max} \\ u_{\min} & \text{for } u(k) \leq u_{\min} \end{cases} . \tag{3.21}$$

Here, for instance $u_{\min}$ can be chosen zero, and $u_{\max}$ a scaled down version of $u(k)$ that ensures that the utilization (2.4) is smaller than one. Simulation is used to investigate whether saturating the control output is useful. If utilization violating input signals become a problem, saturation can be used. Another approach could be to design the controller with input constraints.

The final step of the control framework is completed. A conversion algorithm that translates a continuous signal into events has been derived. The conversion attempts to equalize the desired and the actual production. Now, the total control framework can be tested by means of simulation. Simulation and analysis is the topic of the next section.

3.6 Simulations

The previous devised control framework can now be tested on the discrete event model of the re-entrant flow line described in Chapter 2. This is done by means of simulation. In the simulation, all elements of the control framework are coupled together. For this coupling to work properly, a number of tools are used. First, the simulation tools as well as the layout of the implementation are shortly be explained. Then, the approximation model together with the feedback pole placement controller are tested by analysing a step response. When the desired behavior is obtained, the controller is uncoupled from the approximation model, and coupled through conversion steps to the discrete event model. These results are analyzed at the end of this section.

Simulation layout

The simulation makes use of three different languages: χ, MATLAB® , and Python as well as an interface between Python and MATLAB® : PYMAT [Ste99]. The discrete event system is modeled in χ [Roo00] [Kle01], and forms the central model from which calls to other computing environments or databases are made. These calls are made through functions. It is possible to evaluate functions, written in Python, from within the χ simulation model [Hof01]. Also, by using the PYMAT interface, it is possible to extract the result of a function written in MATLAB® into Python. Combining these two properties results in an interesting feature, where functions written in MATLAB® can be evaluated in Python, after which the result can be imported in χ. Through PYMAT, a coupling between functionality of MATLAB® and χ exists.

The elements of the control framework are implemented in different environments. The conversion algorithm is a function in χ. The pole placement controller is a MATLAB®function that every sample period determines the new control input. The data used in the discrete event model is retrieved by Python functions from a Python database. In this way, data is stored at a central place, and compilation time of the model can be reduced. SIMULINK, a MATLAB® simulation toolbox, is used to model and test the approximation model and feedback controller. This can be done off-line.

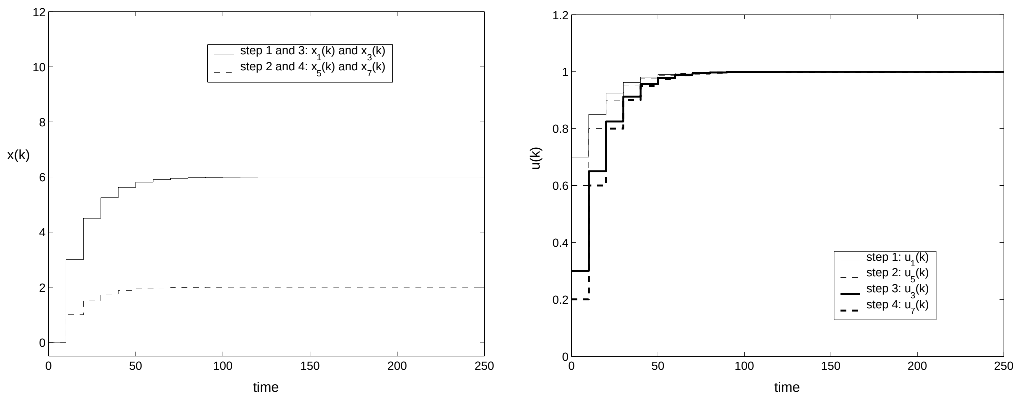
In the following paragraph the approximation model is tested together with the pole placement controller.

Approximation model and pole placement controller

The MIMO pole placement algorithm is tested by simulating the approximation model with controller in SIMULINK, and checking whether the closed-loop system has the desired behavior. The system is designed to track a constant reference signal x_{ref}. Therefore, the error between $x(k)$ and x_{ref} should become zero, and the input of the approximation model $u(k)$ should become equal to the constant input λ. The total SIMULINK model can be found in Appendix D.7. This simulation is done with a sample period $h = 10$. All poles are placed in the origin, which leads to an N given in (3.19). Other parameters are:

$$x_{\text{ref}} = \begin{bmatrix} x_B \\ x_S \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 6 \\ 6 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}, \quad x_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad t_e = \begin{bmatrix} 2/3 \\ 1/3 \\ 1/3 \\ 2/3 \\ 1/5 \\ 2/15 \\ 2/15 \\ 1/5 \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_a \\ \lambda_b \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}. \quad (3.22)$$

Simulating the state-space approximation model and controller with the parameters in (3.22), leads to the response in Figure 3.5. Since both product types display exactly the same re-



(a) buffer levels (b) control signal

Figure 3.5: Approximation model controlled by pole placement. Four process steps of product type a.

sponse, Figure 3.5 displays just the response for product type a. The figure shows that the wip levels become equal to the reference values, so the error (3.9) decreases to zero, which causes the control signal to become equal to λ. Also, note the number of steps it takes the output to become equal to the reference value. In exactly eight steps the error reduces to zero; eight is the order of the state-space model. This is a result from the deadbeat controller (all poles in the origin), since it takes a deadbeat controller exactly n steps[2] to reduce the error to zero.

Another observation concerns the trajectories of the individual control signals $u(k)$ in relation to each other. The relation of the individual trajectories can be explained as follows. The output must be driven from x_0 to x_{ref}. Since x_0 is smaller than x_{ref} the volume in the buffers $x(k)$ must be increased. To ensure that $x(k)$ is increased, by using the system equations (3.7) and (3.8), the following must hold for $u(k)$

$$\begin{array}{lr} \text{step 1} & \lambda - u_1(k) > 0 \\ \text{step 2} & u_1(k) - u_5(k) > 0 \\ \text{step 3} & u_5(k) - u_3(k) > 0 \\ \text{step 4} & u_3(k) - u_7(k) > 0 \end{array},$$

which is equivalent to

$$\lambda > u_1(k) > u_5(k) > u_3(k) > u_7(k)\,. \tag{3.23}$$

When $x(k) = x_{\text{ref}}$ it must hold that

$$\lambda = u_1(k) = u_5(k) = u_3(k) = u_7(k)\,, \tag{3.24}$$

[2]n is the order of the state-space model

and since $\lambda = 1$, it follows that $u(k)$ also becomes equal to one. Figure 3.5(b) shows that the trajectories of $u(k)$ satisfy equations (3.23) and (3.24).

Conclusion, MIMO pole placement works excellent for the state-space approximation model. However, the challenge is to control the *discrete event model* with pole placement. The simulation of the discrete event model with pole placement is discussed in the next paragraph.

Discrete event model and pole placement controller

In the previous section it is seen that the approximation model controlled by pole placement displays the desired behavior. The error reduces to zero and the input $u(k)$ becomes equal to the desired throughput. Now, the controller is uncoupled from the approximation model and coupled through conversions steps to the discrete event model. The same parameters (3.22) are used as before. As discussed earlier, for a discrete event model, utilization is an important measure; the utilizations of the machines are determined by (2.4)

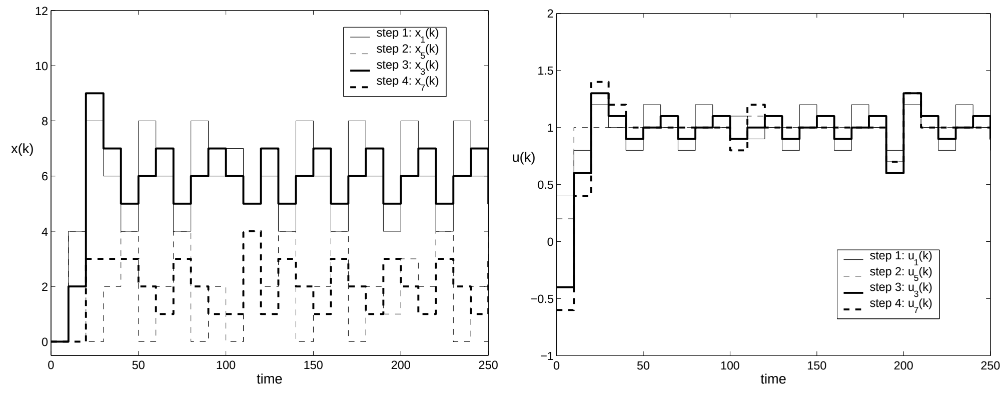
$$\begin{aligned} \rho_B &= \frac{1}{3}(\frac{2}{3} + \frac{1}{3} + \frac{1}{3} + \frac{2}{3}) = \frac{2}{3}\,, \\ \rho_S &= \frac{1}{15} + \frac{2}{15} + \frac{2}{15} + \frac{1}{15} = \frac{2}{3}\,. \end{aligned} \tag{3.25}$$

To get a first impression of the simulation, the utilizations are deliberately kept low. Later on, machines are utilized more and the boundaries of the current strategy are explored. The simulation results of the discrete event model controlled by pole placement are discussed next.

During the simulation 10.000 lots are processed, since the throughput demanded for both product types is equal ($\lambda_a = \lambda_b = 1$), approximately half are type a lots and half are type b lots. The simulation is analysed by means of a couple of figures. Figure 3.6 displays the first 250 time units of the total simulation. During this time about 500 lots are processed. The figure on the left displays the buffer values observed at the sample times for the four buffers of lot type a. The figure on the right displays the corresponding control signal. Figure 3.6 can be compared to Figure 3.5[3]. At first, the resemblance may appear small. However, a closer investigation shows that the figures indeed resemble each other. In Figure 3.5 the buffer levels become exactly equal to the reference values. In Figure 3.6 the buffer levels oscillate around the reference values, since in the discrete event model, buffer levels are naturals. The two batch buffers, $x_1(k)$ and $x_3(k)$, oscillate around six; the two single lot buffers, $x_5(k)$ and $x_7(k)$, oscillate around two. The same holds for the control signal; the control signal oscillates around λ.

Another remarkable observation is the apparent periodic patterns of the buffer levels (observed at the sample times) and the control signal. The buffer levels oscillate around the reference values without any big deviations. Only a few sets of buffer level combinations appear to occur. In order to investigate this periodic behavior further, the simulation is plotted for a longer time. Figure 3.7 displays the buffer levels and control signal of process step 3 and 4 for 3.000 time units. During this time about 6.000 lots are processed. It is seen that occasionally buffer levels are observed that fall outside the patterns, but overall the buffer levels observed at the sample times are subject to periodic patterns. Table 3.1 focuses closer on these patterns. The table highlights the buffer levels and control signals for all

[3]mind the slightly different y-axis for $u(k)$



(a) buffer levels (b) control signal

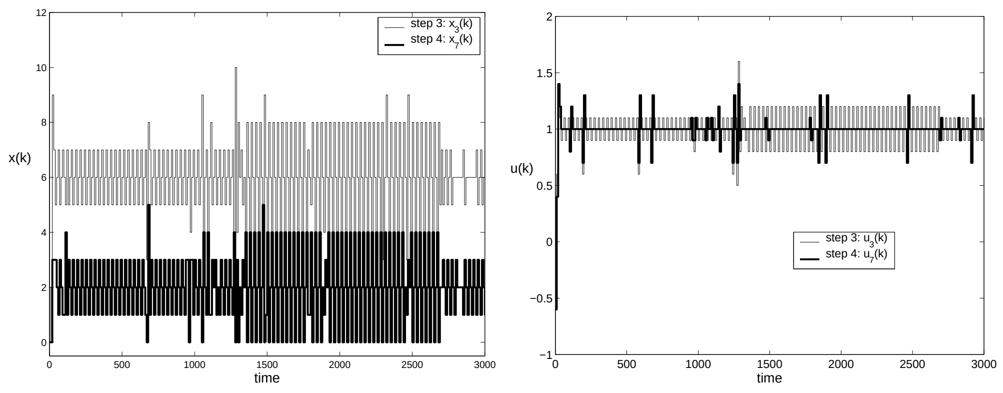
Figure 3.6: Discrete event model controlled by pole placement: first 250 lots of product type a

buffers during a period of 15 samples. Examining the pattern of $x_5(k)$, it is seen that the pattern $2-4-0$ repeats itself only to be interrupted once by the pattern $2-3-1$. The other buffer levels, and the control signals also display a pattern of period three. The values of the corresponding control signals can be explained by examining the pole placement feedback matrix N (3.19) and recalling that inequalities (3.23) must hold to increase (or the opposite to decrease) the buffer values.

time	610	620	630	640	650	660	670	680	690	700	710	720	730	740	750
$x_1(k)$	6	4	8	6	4	8	6	7	5	6	4	8	6	4	8
$x_2(k)$	6	4	8	6	4	8	6	7	5	6	4	8	6	4	8
$x_3(k)$	7	5	6	7	5	6	7	3	8	7	5	6	7	5	6
$x_4(k)$	7	5	6	7	5	6	7	8	3	7	5	6	7	5	6
$x_5(k)$	2	4	0	2	4	0	2	3	1	2	4	0	2	4	0
$x_6(k)$	2	4	0	2	4	0	2	1	3	2	4	0	2	4	0
$x_7(k)$	1	3	2	1	3	2	1	0	5	1	3	2	1	3	2
$x_8(k)$	1	3	2	1	3	2	1	0	5	1	3	2	1	3	2
$u_1(k)$	1.0	0.8	1.2	1.0	0.8	1.2	1.0	1.1	0.9	1.0	0.8	1.2	1.0	0.8	1.2
$u_2(k)$	1.0	0.8	1.2	1.0	0.8	1.2	1.0	1.1	0.9	1.0	0.8	1.2	1.0	0.8	1.2
$u_3(k)$	1.1	0.9	1.0	1.1	0.9	1.0	1.1	0.9	1.0	1.1	0.9	1.0	1.1	0.9	1.0
$u_4(k)$	1.1	0.9	1.0	1.1	0.9	1.0	1.1	1.2	0.7	1.1	0.9	1.0	1.1	0.9	1.0
$u_5(k)$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.2	0.8	1.0	1.0	1.0	1.0	1.0	1.0
$u_6(k)$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$u_7(k)$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	1.3	1.0	1.0	1.0	1.0	1.0	1.0
$u_8(k)$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 3.1: Periodicity in the buffer levels and the control signal

Another fact that makes the patterns remarkable is that the patterns appear by observing the buffer levels at the sample times; in between these sample times the actual buffer levels in the discrete event model are not evidently periodic. Figure 3.8 shows the difference between the actual buffer levels and the buffer levels observed by the controller at the sample times. During one sample the buffer level changes approximately 15 times. At the sample times, the controller observes the discrete event model, and determines an input signal. Then the



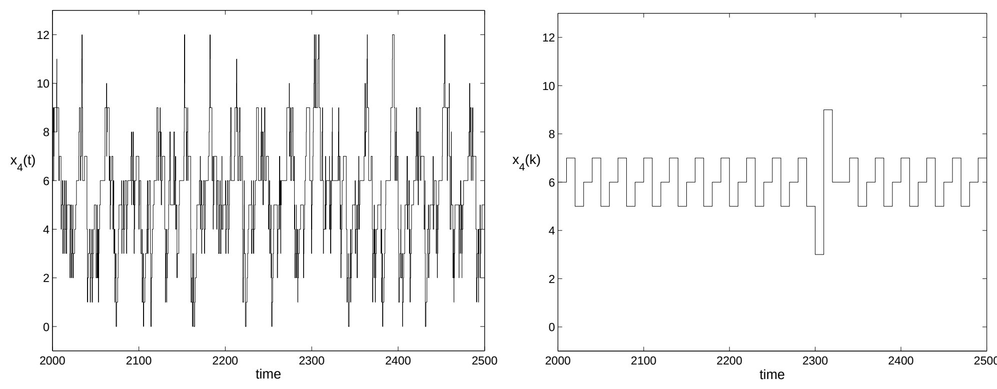
(a) buffer levels (b) control signal

Figure 3.7: Buffer levels and control signal for a utilization of 2/3. Periodicity in buffer levels and control signal

conversion algorithm starts to work, and ensures that all requested production is performed over the following sample period. For a low throughput ($\lambda = 1$, $\rho = 2/3$), the repetitive control signal causes the buffer levels observed at the sample times to also display a repetitive behavior. An thorough explanation for the periodic behavior remains to be found. This concludes the examination of the discrete event model controlled by pole placement for low utilization levels. The next simulations investigate the performance of the system when the utilization is increased.

The question arises how the performance of the discrete event model controlled by pole placement is effected by an increased utilization. Related to this, is the question what remains of the discussed periodic patterns when the utilization is increased. Equivalently, how do the repetitive control signals influence the flow time and the wip. These questions are dealt with by examining two simulations. First, a simulation is performed where the wip and the flow time are investigated as a function of the utilization. The simulation will show that signals are more periodic for a lower throughput, and supposedly more periodic signals lead to a better performance. Second, a simulation is performed where the influence of the sample time is briefly investigated. From these simulations it becomes clear what possibilities of the current strategy are, and where improvements can be made.

During the first simulation, the wip and the flow time are examined while the utilization is increased. The utilization is increased by increasing the reference throughput: λ is increased from 0.8 with steps of 0.05 until 1.3, and from 1.3 with steps of 0.02 until 1.46. To ensure that the mean values are reliable, the simulation is repeated at least 30 times for every value of the throughput. This simulation layout is a standard layout used by the Systems Engineering Group. Basically, it states the following: when the simulation is repeated 30 times, the central limit theorem determines whether the output parameter (e.g. flow time) is accurate enough. The output parameter is accurate enough when the confidence interval divided by the mean value of the output parameter is smaller than 5 percent. If the output parameter is not accurate enough, the simulation is repeated until the output parameter is accurate enough. The described simulation layout is used for several simulations also in the following chapter. During every simulation 20.000 lots are processed,



(a) buffer level in discrete event model

(b) buffer level in discrete event model observed by the controller at the sample times

Figure 3.8: Buffer levels observed at the sample times appear periodic

of which the first 2.000 lots are ignored. The same parameters are used as before (3.22), except for the varying λ. Figure 3.9 displays the result of the above described simulation.

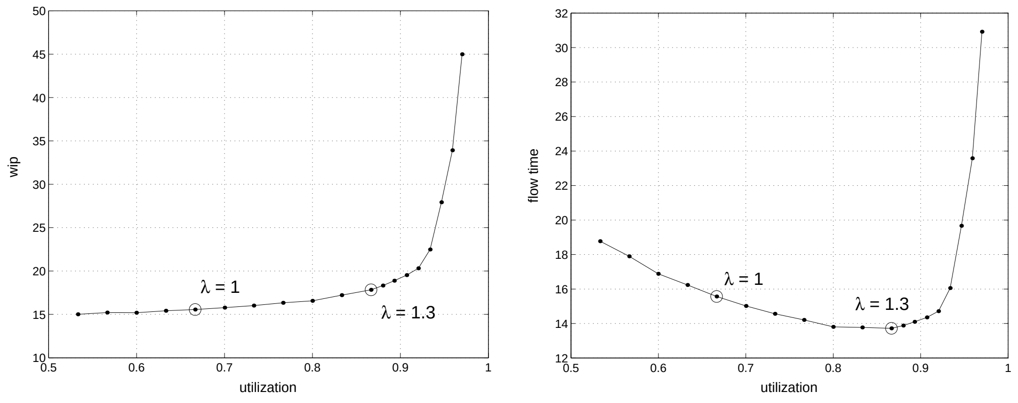
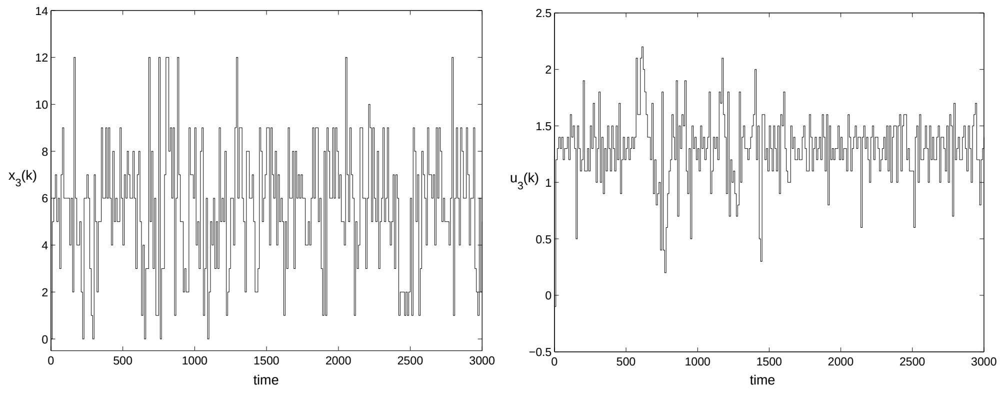


Figure 3.9: Mean wip and mean flow time of lot type a as a function of the utilization

Figure 3.9(a) shows the expected behavior: as the utilization increases, the wip in the discrete event model increases asymptotically towards one. A utilization of one is never reached since the system contains variability. In variable systems, machines cannot be utilized 100 percent of their time [Hop00]. Figure 3.9(b) requires some more explaining, since strangely enough, for low utilizations the flow time increases. This increase in flow time can be explained using Little's (2.1) law. According to Little it holds that if the throughput is increased while the same amount of wip is used, the flow time must decrease. The controller is designed to follow a reference wip level. The reference wip level is equal to six lots for the batch buffers and two lots for the single lot buffers. Apparently, the reference wip is chosen too high for the desired throughput, because when the throughput is increased, still the same amount of wip—or just a bit more—is used, resulting in a lower flow time. This means that in the low throughput case, lots enter the line so slow that they spend a fair amount of time waiting until the reference wip level is reached. If the lots are inserted faster, this waiting time is decreased, and so is the flow time. For a reference wip of six and two (3.22), and a sample time of ten, the minimal flow time is reached somewhere between 80 and 90 percent utilization. This approach can be used to determine a good target throughput for a given reference wip and sample time.

To answer the question what remains of the periodic patterns when the utilization is increased, let us take a closer look at buffer levels and control signals for a throughput $\lambda = 1.3$, which is equivalent to a utilization $\rho = 0.8667$. For $\lambda = 1$ the buffer levels and control signal were shown in Figure 3.7. Figure 3.10 displays buffer level $x_3(k)$ and control signal $u_3(k)$ for $\lambda = 1.3$ (the mean of $u_3(k) = 1.297$, the mean of $x_3(k) = 5.69$). The



(a) buffer level (b) control signal

Figure 3.10: Buffer level and control signal for a utilization of 0.8667

periodic patterns, that were evident for a throughput of 1, are no longer easily recognized for a throughput of 1.3. The reduction in periodicity appears to be caused by the stochastic effects in the discrete event model. It seems that for a higher throughput the stochastic effects in the discrete event model become more important. A possible reason might be that due to the higher arrival rate of lots, the probability to observe the system in a higher state (higher buffer levels) increases. For a system with exponential arrivals and service times, this can be shown by formulating a Markov chain. Then, the buffer levels observed at the sample times variate more, and since the control signal is linearly related to the buffer levels, also the control signal varies more. More research is needed to determine the exact cause of the loss of periodicity. When the utilization is pushed further towards one, the variability on the buffer levels and the control signal become even greater. The reference wip levels are no longer reached, so the error between the reference wip and the actual wip is increasing. Due to an increasing error, and an increasing variance on the buffer levels, the controller demands more and more throughput from the discrete event model. This increase in throughput reveals itself in a higher control signal; the point is reached where the demanded throughput can no longer be processed by the discrete event model. At this point the conversion algorithm needs improvement, since the conversion remembers all throughput ever demanded by the controller. If the controller demands an unrealistic amount of throughput—because the reference wip level cannot be reached—the conversion algorithm should correct this. More research into a conversion algorithm for highly utilized systems is needed.

During the above simulation discussion the boundaries of the current strategy are investigated and some points of improvement turned up. Also, the above discussion led to suspect that a lower flow time (or wip) can be achieved when the control signal becomes more periodic. A question that remains is what influence the sample time has on the flow time and the periodic patterns. The second simulation briefly investigates this subject. In the second simulation, the sample time is varied. The same simulation setup and parameters that led to Figure 3.9 are used, except now a sample time of 100 instead of 10 is used. During one sample, each buffer level changes about 150 times. This is hardly an adequate control strategy, since the controller almost never reacts. Figure 3.11 displays the simulation results. It is seen that for a larger sample time, the flow time is considerable

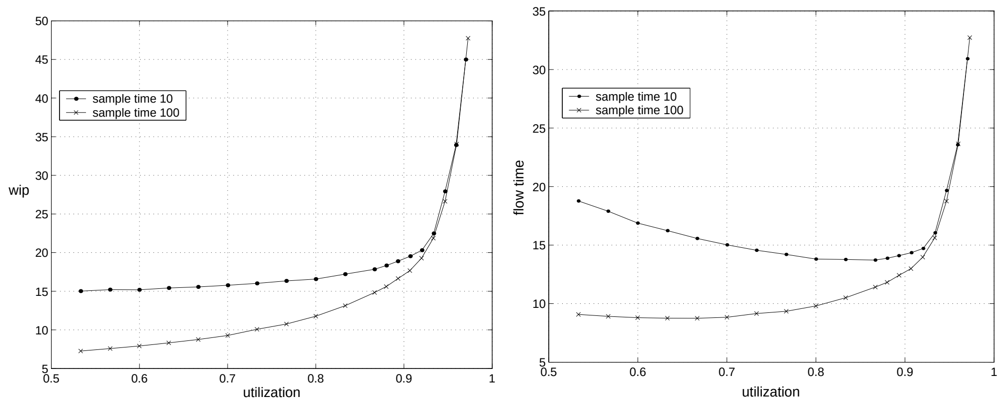


Figure 3.11: Mean wip and mean flow time of lot type a for a small and a large sample time as a function of the utilization

lower. This seems to be a good result, a lower flow time and less wip indicate a better performance. However, while the performance of the discrete event model might be better, the performance of the controller decays. Recall that the controller was designed to follow a reference signal. Figure 3.11 shows that not the reference wip level, but a lower wip level is achieved. Note that the lower wip level could also be achieved by reducing the reference wip, and keeping the smaller sample time. Due to the increased sample time, the controller acts on outdated information from the discrete event model. It remains a point of discussion why the controller with a larger sample time achieves of all possible flow times, precisely lower ones. An explanation for the lower flow time might be the reduction of the probability to observe rare (i.e. high) buffer levels in the system. If less rare buffer levels are observed, then the control signal is likely to become more periodic, which could be a reason for a lower flow time. More research is needed to understand why a lower flow time instead of a higher one is achieved, and what role periodic control signals play in this context. The influence of the sample time should be investigated further. For now, it is noted that a sample time choice should consider two effects: on one hand, if the sample time is too small, the controller reacts on too much stochastic effects or noise; on the other hand, if the sample time is too large, the controller does not react at all. If the demand is no longer constant—as it was always assumed—the sample time might be chosen relative to the frequency of the demand. Finally, it is suspected that if the sample time is chosen too small, the assumption that the discrete event model can be approximated by a continuous model is no longer valid. The investigation of the validity of the approximation model, as well as a further investigation of the sample time, remain issues for future research.

The simulations of the discrete event model controlled by pole placement are completed and can be concluded by a short revision. The simulation shows that it is possible to control a discrete event model by pole placement. It is shown that periodic patterns emerge for low utilizations, and that these patterns are no longer evident when the utilization is increased. Apparently, when systems are observed at discrete times, stochastic effects play a more important role in highly utilized systems than in systems with a low utilization. For highly utilized systems, it is noted that the conversion algorithm needs improvement. It is suggested that periodic input signals result in a better performance. Furthermore, it is suspected that larger sample times result in more periodic signals. However, for larger sample times the performance of the controller decays. The sample time should be a consideration between reacting too fast on stochastic changes or not reacting at all. More research in this area is needed. The following section contains a short discussion on the first approach to designing the elements of the control framework as discussed in this chapter. During this discussion the first approach is reviewed and the second approach, which forms the subject of the next chapter, is introduced.

3.7 Discussion

Chapter 3 presents a first approach to control discrete event models using continuous control techniques by means of the control framework discussed in Chapter 1. The framework consists of a number of elements. These elements are, as a start, chosen as modest as possible. The question arises whether effects present in the discrete event model but neglected by the approximation model or the controller could improve the performance of the overall strategy.

The simulation in the previous section already showed that for highly utilized systems, the controller demands a throughput level that exceeds the capacity of the machine. This could be considered a constraint violation of the controller, which is one of the neglected effects. It was noted that the conversion algorithm might correct this; however, if constraints are added to the controller, unrealistic signals could beforehand be prevented from occurring, and the problem is solved closer to the origin.

Another neglected effect is the process time of lots. In the discrete event model, lots take a certain amount of time to be processed; this time is neglected in the approximation model. Process times can be considered as time delays, since a certain amount of time passes between the moment a lot is extracted from one buffer and inserted into another. In the approximation model an input change is noticed instantaneously in the buffer before and behind the machine, while in the discrete event model the buffer behind the machine only notices this input change some time later.

On the one hand, including more information about the discrete event model in the approximation model or the controller leads to more accurate models, which therefore is expected to lead to a better performance. On the other hand, including more realistic elements increases the complexity of the model. One only wants to add those elements that contribute significantly to the performance of the overall system. However, to determine which elements are significant, still a comparison has to be made between the case where the elements are included, versus the case where the elements are excluded. A discussion on significance can only be held on basis of such a comparison, besides of course a doses of common sense.

As a comparison to the first approach discussed in this chapter, the next chapter discusses a new approach. During the new approach, two specific elements of the control framework are adjusted. First, process times are modeled as time delays and included in the approximation model. Second, a controller is design using model predictive control in stead of pole placement. Model predictive control is used since it allows the possibility to use input and state constraints. Input constraints can be used to avoid unrealistic control signals; state constraints can be used to avoid control signals that would result in negative buffer levels. Through these two changes, the neglected effects mentioned above can accounted for.

The new approach is presented in a similar way as the approach in this chapter. Chapter 4 starts by presenting a possibility to model time delays in a state-space model by Padé approximations. Again, this approach is presented by means of an example. Then, model predictive control is discussed, and a controller is devised. No changes have yet been made to the conversion algorithm, so the same algorithm is used. Finally, the simulation results are presented, followed by a comparison between the old and the new approach.

Chapter 4

Fluid Model with Delay Controlled by MPC

The objective of this research is to use techniques from control theory to control discrete event models of manufacturing systems, in order to create an alternative to heuristic control policies that are often used to control manufacturing systems. The control framework discussed in Chapter 1 sets a framework for this objective. The elements of the framework are designed and simulated in this thesis, and the overall strategy is tested.

The previous chapter presented and discussed a first approach to design the elements of the control framework. The discussion at the end of the previous chapter, suggested two main extensions to the first approach. First, the mean process times of lots are included in the approximation model. Second, the controller is designed using model predictive control (MPC) instead of pole placement. The reason for including mean process times in the approximation model is, that since process times are present in the discrete event model and not in the approximation model, the approximation assumes lots to move instantaneously from one buffer to another, while in the discrete event model lots move from one buffer to another across a machine that processes them for a certain amount of time. The reason for replacing pole placement by MPC is that MPC can handle constraints on inputs and outputs. Input constraints can be used to avoid control signals that exceed the machine capacity. Output constraints can be used to prevent control signals that result in negative buffer levels. Also, MPC naturally handles multiple input multiple output (MIMO) systems.

The chapter starts by presenting a method to include mean process times in a state-space approximation model in the form of time delays.

4.1 Fluid approximation model with delay

This section treats the first step of the control framework. During the first step of the control framework an approximation model is derived that is compatible with control theory. This section uses a similar approach as presented in Section 3.3, only now mean process times are explicitly accounted for in the approximation model. The techniques used to derive the fluid model with delay are easier explained in continuous-time than in discrete-time. Therefore, this section uses a continuous-time linear time invariant (LTI) state-space model. During the controller design a discrete-time model is created using the MATLAB® function C2D. Generally a continuous-time state-space model is given by.

$$\dot{x} = Ax + Bu \tag{4.1a}$$

$$y = Cx + Du\,, \tag{4.1b}$$

where $\dot{x}$ denotes the derivative of x, x is an $(n \times 1)$ state vector, u is an $(i \times 1)$ input vector, and y is a $(o \times 1)$ output vector. Consequently, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times i}$, $C \in \mathbb{R}^{o\times n}$ and $D \in \mathbb{R}^{o\times i}$. Here n denotes the order of the state-space model. Also,

$$\begin{aligned} n &= \text{number of states,} \\ i &= \text{number of inputs,} \\ o &= \text{number of outputs.} \end{aligned}$$

The new approach is again based on modeling buffer contents throughout time. Similar as before, a chain of buffers can be created, where buffers are fed from upstream buffers, and feed downstream buffers.

The difference between the previous approach and the new approach is that the influence of machines is partly included in the approximation model. In the approximation model machines are modeled as elements that introduce a certain amount of dead time (delay) to the system. However, a possible important difference between the discrete event model and the approximation model remains; the dead times that represent the process times are assumed *deterministic*, while in the discrete event model process times are *stochastic*. The stochastic process times cause the discrete event model to display different dynamics than the approximation model. This difference may become important and is discussed further during the simulation analysis.

Shortly stated, the following steps are taken to include the mean process times in a spate-space model. The mean process times are modeled as dead time in the system equations. Then, the system equations are transformed using Laplace, and the Laplace transforms are approximated using Padé approximations. This results in a transfer function matrix. Next, the transfer function matrix is transformed into a state-space model. For systems with a large number of inputs and outputs, state-space models can become quite large and unpleasant to work with. Techniques from Kailath [Kai80] or MATLAB® can be used to derive a minimal state-space realization of the given transfer function matrix.

A number of expressions, such as Padé approximations, transfer function matrices, and minimal state-space realizations are mentioned in the above approach. The following example—an extension of Example 3.3.1—clarifies the mentioned approach and techniques, by showing how a small flow line can be modeled by a state-space model that includes mean process times.

Example 4.1.1 *This example shows how a small flow line can be modeled by a state-space approximation model in which the mean process times of the machines are explicitly present. Consider the same flow line used in Example 3.3.1. The flow line consists of a generator (G), two buffers (B_1 and B_2), two machines (M_1 and M_2), and an exit process (E). The generator inserts a lot into the line when told by the controller. The buffers are FIFO buffers. The machines process lot for lot; each lot has a stochastic process time. The mean process times (t_{e1} and t_{e2}) are depicted above the machines. The exit process receives lots*

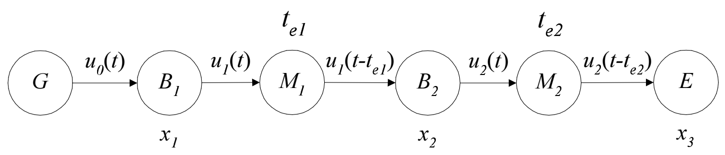


Figure 4.1: A flow line example with delay

immediately when they are finished processing by the second machine. The exit process can be seen as a finished goods buffer. In Figure 4.1, u are controllable inputs and represent the rate at which lots are subtracted from one buffer and inserted in the next. Conform the previous example, the states are equal to the outputs: $x = y$, and the outputs are again buffer levels. The difference between this example and the previous example is that previously the inflow of a buffer behind a machine equaled the out-flow of the buffer before that machine at time t, while currently the inflow of the buffer behind a machine equals the out-flow of a buffer before that machine at time $t - t_e$. Buffers are still linked together only a delay is present in the linking. This delay is the mean process time.

The system with delay is described by the following equations.

$$\begin{aligned} \dot{x}_1(t) &= u_0(t) - u_1(t) \\ \dot{x}_2(t) &= u_1(t - t_{e1}) - u_2(t) \\ \dot{x}_3(t) &= u_2(t - t_{e2}) \\ y_1(t) &= x_1(t) \\ y_2(t) &= x_2(t) \\ y_3(t) &= x_3(t) \end{aligned} \quad . \tag{4.2}$$

Since the delays t_{e1} and t_{e2} are present in $u_1(t - t_{e1})$ and $u_2(t - t_{e2})$ a state-space model cannot be derived directly. In order to form a state-space model, the delay must be approximated. This is done by using Padé approximations. The example is shortly interrupted to explain how this works.

Padé Approximations

Let a function with time delay t_e be given by $f(t - t_e)$. Then, the Laplace transform of $f(t - t_e)$ is given by

$$\mathcal{L}\{f(t - t_e)\} = F(s)e^{-t_e s} \tag{4.3}$$

Since the term $e^{-t_e s}$ is present in the transfer function, the numerator in the transfer function is no longer a polynomial expression, and forming a control system is no longer possible by standard techniques. One way to solve this problem, is by approximating $e^{-t_e s}$ around $s = 0$ by a rational function. A common method for finding such an approximation is by using Padé approximations. A description of this process can be found in Paragraph 5.7.1 of [Fra94]. In this thesis, a second order Padé approximation of $e^{-t_e s}$ is used, given by the following expression

$$e^{-t_e s} \cong \frac{(t_e s)^2 - 6t_e s + 12}{(t_e s)^2 + 6t_e s + 12} \ . \tag{4.4}$$

To clarify the effect of Padé approximations a bit more, consider Figure 4.2. Figure 4.2 displays the step response of two transfer functions: (i) a transfer function with a time delay of one given by $\frac{1}{s}e^{-s}$, and (ii) a transfer function where the time delay is approximation by a 2nd order Padé approximation given by $\frac{1}{s}\frac{s^2-6s+12}{s^2+6s+12}$.

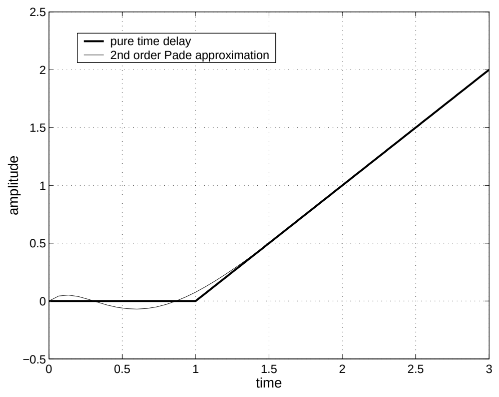


Figure 4.2: A comparison of step responses: a time delay of one approximated by a 2nd order Padé approximation.

Example 4.1.1 Continued *From this point on, the procedure to form the state-space model consists the following two steps: (i) compose a transfer function matrix, (ii) derive the desired state-space model from the transfer function matrix. Transfer functions can be composed between all inputs and outputs, when these transfer functions are put in a matrix, this matrix is called a transfer function matrix. A description of a transfer function matrix is given in Appendix A.1. To form a transfer function matrix, transform* (4.2) *to the frequency domain using Laplace* [1]

$$\begin{aligned} sX_1 &= U_0 - U_1 \\ sX_2 &= U_1 e^{-(t_{e1}s)} - U_2 \\ sX_3 &= U_2 e^{-(t_{e2}s)} \\ Y_i &= X_i \quad \forall\, i \end{aligned} . \tag{4.5}$$

Then, write down the individual transfer functions as follows

$$\begin{array}{lll} H_{10} = \dfrac{Y_1}{U_0} = \dfrac{1}{s} & H_{11} = \dfrac{Y_1}{U_1} = -\dfrac{1}{s} & H_{12} = \dfrac{Y_1}{U_2} = 0 \\ H_{20} = \dfrac{Y_2}{U_0} = 0 & H_{21} = \dfrac{Y_2}{U_1} = \dfrac{1}{s}e^{-t_{e1}s} & H_{22} = \dfrac{Y_2}{U_2} = -\dfrac{1}{s}\,, \\ H_{30} = \dfrac{Y_3}{U_0} = 0 & H_{31} = \dfrac{Y_3}{U_1} = 0 & H_{32} = \dfrac{Y_3}{U_2} = \dfrac{1}{s}e^{-t_{e2}s} \end{array} \tag{4.6}$$

and form the transfer function matrix,

$$H(s) = \begin{bmatrix} H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \\ H_{30} & H_{31} & H_{32} \end{bmatrix} = \begin{bmatrix} \frac{1}{s} & \frac{-1}{s} & 0 \\ 0 & \frac{1}{s}e^{-(t_{e1}s)} & \frac{-1}{s} \\ 0 & 0 & \frac{1}{s}e^{-(t_{e2}s)} \end{bmatrix}, \tag{4.7}$$

[1] capitals are used for the transformed variables

where $e^{-(t_e s)}$ *is given by* (4.4).

Given a transfer function matrix it is possible to derive a state-space model. This is called making a state-space realization from a transfer function matrix. The example is interrupted to explain two approaches to form a state-space model from a given transfer function matrix.

From transfer function matrix to state-space model

It is possible to transform a transfer function matrix into a state-space model. This process is called making a state-space realization. To prevent the order of the state-space model from becoming too large (using to many states), a minimal state-space realization is recommended. Two approaches to derive the realization are suggested in this thesis. The first approach uses MATLAB® functions; the second approach uses relations from Kailath [Kai80].

- MATLAB® approach

  The CONTROL SYSTEM TOOLBOX from MATLAB® has an extensive amount of useful functions to derive state-space models. A possible method to derive a minimal state-space model is by using the following functions: (i) TF with option '`iodelay`' obtains a system with delay, (ii) PADE approximates the delays with Padé approximations, leading to (4.7), (iii) TFM2SS transforms the transfer function matrix into a state-space model, (iv) SS with option '`min`' obtains a minimal realization of the state-space model. Function TFM2SS requires the least common multiple of the transfer function matrix. A function that easily extract this information is TFMLCM given in Appendix D.6. The procedure is attractive since no complicated functions are used. A down side, however, is that the '`min`' option of SS destroys any recognizable patterns in the state-space matrices for high order (order size 20) models. By using the Kailath approach these patterns remain visible.

- Kailath approach

  Kailath [Kai80] provides well developed theories for polynomial matrix methods and control theory. Chapter 6 of [Kai80] contains theories for deriving various canonical forms and minimal realizations for MIMO systems. To obtain a minimal state-space realization, this thesis employs a procedure following steps stated in Chapter 6 of [Kai80]. The procedure and an explanation of the necessary polynomial matrix methods can be found in Appendix A.

The procedure may seem hard to tackle at first, but it does result in more elegant matrices than the MATLAB® variant. To attempt to provide more insight, the procedure is applied to the small flow line with delay presented in Figure 4.1 and described by (4.2). Example 4.1.1 continuous by applying the Kailath approach.

Example 4.1.1 Continued *The system equations with delay* (4.2) *were approximated, which led to the transfer function matrix* (4.7)*. At this point, the transfer function matrix is transformed into a minimal state-space model by using the Kailath approach presented in Appendix A. The remainder of this example presents the steps of the realization procedure applied to the small flow line with delay. The result is a state-space approximation model of minimal order in which time delays are present. Assume in this example, the process times have values:* $t_{e1} = 1$*, and* $t_{e2} = 1$*. Then, the transfer function matrix* (4.7) *becomes*

$$H(s) = \begin{bmatrix} \frac{1}{s} & \frac{-1}{s} & 0 \\ 0 & \frac{s^2-6s+12}{s(s^2+6s+12)} & \frac{-1}{s} \\ 0 & 0 & \frac{s^2-6s+12}{s(s^2+6s+12)} \end{bmatrix} . \tag{4.8}$$

The least common multiple $d(s)$, and the remaining $N(s)$ given by (A.6) are found to be

$$d(s) = s(s^2+6s+12) \; , \tag{4.9}$$

$$N(s) = \begin{bmatrix} s^2+6s+12 & -s^2-6s-12 & 0 \\ 0 & s^2-6s+12 & -s^2-6s-1 \\ 0 & 0 & s^2-6s+12 \end{bmatrix} . \tag{4.10}$$

The Smith form (Appendix A.4) of $N(s)$ is given by

$$\Lambda(s) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & (s^2+6s+12)(s^2-6s+12)^2 \end{bmatrix} . \tag{4.11}$$

The denominator matrix $D_0(s)$ is too large to display here. Matrix $\Psi_R(s)$ (A.27) *is given by*

$$\Psi_R(s) = \begin{bmatrix} s(s^2+6\,s+12) & 0 & 0 \\ 0 & s(s^2+6\,s+12) & 0 \\ 0 & 0 & s \end{bmatrix} \tag{4.12}$$

Therefore, the minimal degree of the system (McMillan degree (A.21)*) is equal to*

$$\begin{aligned} n_{\min} &= \sum \deg \psi_i(s) \\ &= 2*3+1 = 7. \end{aligned} \tag{4.13}$$

The Hermite form (Appendix A.6) of $D_0(s)$ is given by

$$D_H(s) = \begin{bmatrix} s & 0 & 0 \\ 0 & s(s^2+6\,s+12) & 0 \\ 0 & 0 & s(s^2+6\,s+12) \end{bmatrix} . \tag{4.14}$$

$D_H(s)$ is split in two parts (A.30)*: D_{hr} and D_{lr}. Since D_{hr} represents the coefficients of the highest order of s, and the diagonal elements are monic*[2]*, it follows that D_{hr} is a unit matrix of size $(i \times i)$. Parts D_{hr} and the transpose of D_{lr} are given by*

$$D_{hr} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} , \qquad D_{lr}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 12 & 0 \end{bmatrix} , \tag{4.15}$$

and the vector l_i (A.33) *is found to be*

$$l_i = \begin{bmatrix} 1 & 3 & 3 \end{bmatrix} . \tag{4.16}$$

[2]A polynomial is said to be monic if the highest power of s has coefficient one

The matrix $\bar{N}(s)$ (A.41) *is given by*

$$\bar{N}(s) = \begin{bmatrix} 1 & 0 & 0 & -s^2-6\,s-12 & 0 & 0 & 0 \\ 0 & 0 & 0 & s^2-6\,s+12 & 0 & 0 & -s^2-6\,s-12 \\ 0 & 0 & 0 & 0 & 0 & 0 & s^2-6\,s+12 \end{bmatrix}. \tag{4.17}$$

All necessary ingredients are presented for the final steps of the procedure. During step 7, 8, and 9, the matrices A, B, C that form state-space model (4.1) are obtained. When all steps are correctly executed the following state-space model is obtained.

$$\dot{x} = Ax + Bu \tag{4.18a}$$
$$y = Cx\,, \tag{4.18b}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -12 & 0 & 0 & 0 \\ 0 & 0 & 1 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -12 \\ 0 & 0 & 0 & 0 & 0 & 1 & -6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.19}$$

$$C = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -12 & 72 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -12 & 72 \end{bmatrix},$$

and

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}, \quad u = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \tag{4.20}$$

A state-space model is obtained that forms an approximation model of the flow line, and includes the mean process times of the machines. The order of the state-space model is seven. Compared to the state-space model without delay (3.6), four more states are needed. Since the Padé approximation is a second order approximation, each time delay adds two states. Therefore, three plus two times two equals seven, which is the order of the new state-space model.

The Kailath approach in comparison to the MATLAB® *approach creates matrices with a more transparent structure. For low order models, like the one in this example, the difference between the Kailath approach and the* MATLAB® *approach is small, however for high order* *models, like the re-entrant flow line (discussed after this example), the difference becomes apparent. A reason might be that the Kailath approach is an exact approach, while it seems that the option 'min' in the function* SS *uses some approximation method. As a comparison, the* MATLAB® *approach is used which leads to the following minimal state-space model*

$$\bar{A}_M = \begin{bmatrix} -1.03e{-16} & -1.33e{-14} & -2.60e{-14} & -1.53e{-33} & +4.36e{-29} & -6.82e{-29} & -3.26e{-45} \\ +9.49e{-15} & -6.00e{+00} & -1.20e{+01} & +4.37e{-31} & -2.33e{-43} & +1.74e{-42} & +3.01e{-43} \\ -2.33e{-15} & +1.00e{+00} & -9.50e{-29} & -1.61e{-31} & -1.38e{-44} & -2.97e{-43} & -7.44e{-44} \\ -3.87e{-16} & +8.12e{-30} & -1.00e{+00} & -4.47e{-32} & -2.68e{-44} & -3.99e{-45} & -1.24e{-44} \\ +5.85e{-29} & -2.04e{-42} & -1.28e{-42} & +8.83e{-45} & -6.00e{+00} & +1.20e{+01} & +1.88e{-57} \\ -8.19e{-30} & -8.32e{-43} & -1.92e{-42} & +1.51e{-45} & -1.00e{+00} & -6.19e{-57} & -2.53e{-58} \\ -8.97e{-30} & -2.45e{-43} & -8.53e{-43} & +1.87e{-47} & +7.97e{-58} & -1.00e{+00} & -2.83e{-58} \end{bmatrix}, \tag{4.21}$$

$$\bar{B}_M = \begin{bmatrix} +7.43e{-02} & -2.17e{-15} & -5.97e{-30} \\ -3.07e{-15} & -1.00e{+00} & +9.44e{-44} \\ +7.69e{-16} & -7.22e{-30} & -1.18e{-44} \\ +1.30e{-16} & -1.63e{-32} & +1.94e{-45} \\ -2.03e{-29} & -1.26e{-43} & +1.00e{+00} \\ +2.25e{-30} & -1.58e{-43} & -4.84e{-58} \\ +2.81e{-30} & -6.86e{-44} & -1.81e{-58} \end{bmatrix}, \tag{4.22}$$

$$\bar{C}_M = \begin{bmatrix} +1.34e{+01} & +1.00e{+00} & +6.00e{+00} & -1.20e{+01} & +8.03e{-29} & +1.05e{-28} & +4.60e{-29} \\ -4.39e{-15} & -1.00e{+00} & +6.00e{+00} & +1.20e{+01} & -1.00e{+00} & +6.00e{+00} & -1.20e{+01} \\ -9.40e{-29} & +1.78e{-42} & -2.83e{-43} & +6.65e{-45} & +1.00e{+00} & +6.00e{+00} & +1.20e{+01} \end{bmatrix} \tag{4.23}$$

All insignificant terms in $\{\bar{A}_M, \bar{B}_M, \bar{C}_M\}$ can be discarded. Then, the following matrices are obtained

$$A_M = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6 & -12 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -6 & 12 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}, \quad B_M = \begin{bmatrix} 0.0743 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.24}$$

$$C_M = \begin{bmatrix} 13.45 & 1 & 6 & -12 & 0 & 0 & 0 \\ 0 & -1 & 6 & 12 & -1 & 6 & -12 \\ 0 & 0 & 0 & 0 & 1 & 6 & 12 \end{bmatrix}.$$

Comparing the minimal state-space realization $\{A_M, B_M, C_M\}$ to the realization $\{A, B, C\}$ (4.19) *found by Kailath's method, it is seen that both realizations are quite similar. However, for high order models, this similarity is lost, and it is recommended to use the Kailath approach.*

It can be concluded that the procedures described in this example lead to a minimal order state-space model in which mean process times are explicitly present. Next, the Kailath procedure—suitable for high order models—is used to form the state-space approximation model of the re-entrant flow line considered in this thesis.

Fluid model with delay of the re-entrant flow line

The previous example explains a concept to include mean process times in a state-space model. The concept uses time delays, Padé approximations, and minimal state-space realizations. Now, the same approach is used to model the re-entrant flow line considered in this thesis. The discrete event model is approximated by a continuous state-space model (4.1) with input u, output y and state x. Inputs, outputs and states are defined in a similar matter as before: let u be the rate at which material is subtracted from one buffer and inserted into another, let y be the buffer levels, and let x be equal to y. The fluid model with delay of the re-entrant flow line is illustrated in Figure 4.3. If Figure 4.3 is compared to Figure 3.3, which illustrates the fluid model without delay, the following differences are noticed. The first difference concerns the treatment of demand and the generator process. Previously, the release of new lots in the system was assumed uncontrollable: the input vector u was split in an uncontrollable part $\bar{u}_1(k)$ (release of new lots) and a controllable part $\bar{u}_2(k)$ (3.10), which led to the error dynamics (3.17). Currently, the release of new lots in the system is assumed controllable, meaning that lots arrive in the first buffers with rate u_9 and u_{10} which are determined by a controller. The generator can now be considered a buffer that delivers immediately when requested and never runs empty. As a result, demand is treated differently. Previously demand was modeled by setting the rates λ_a and λ_b, while currently demand is modeled as an increasing amount of finished goods that forms a reference trajectory traceable by a controller. In total this leads to ten controllable inputs. The second difference is that two extra outputs (states) x_9 and x_{10} are added to the model. These extra outputs represent the number of finished goods for both lot types. If x_9 and x_{10} were omitted, the time delay of the last process step would remain unnoticed in the output.

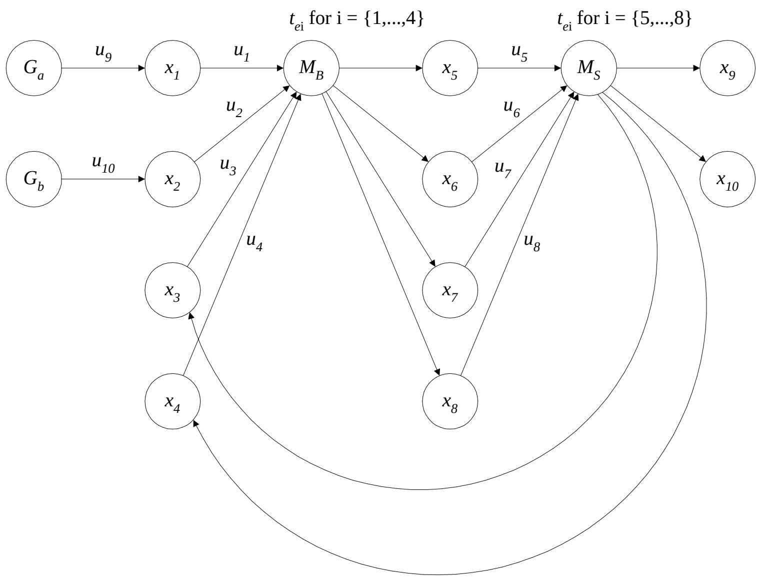


Figure 4.3: Inputs and states of the re-entrant flow line with delay

The system equations, corresponding to Figure 4.3, are obtained by connecting the in-flows and out-flows of buffers, and adding time delays. Consequently, the following equations are obtained for the buffers before the batch machine

$$\begin{aligned} \dot{x}_1(t) &= u_9(t) - u_1(t) \\ \dot{x}_2(t) &= u_{10}(t) - u_2(t) \\ \dot{x}_3(t) &= u_5(t - t_{e5}) - u_3(t) \\ \dot{x}_4(t) &= u_6(t - t_{e6}) - u_4(t) \end{aligned}, \tag{4.25}$$

and for the buffers before the single lot machine

$$\begin{aligned} \dot{x}_5(t) &= u_1(t - t_{e1}) - u_5(t) \\ \dot{x}_6(t) &= u_2(t - t_{e2}) - u_6(t) \\ \dot{x}_7(t) &= u_3(t - t_{e3}) - u_7(t) \\ \dot{x}_8(t) &= u_4(t - t_{e4}) - u_8(t) \end{aligned} , \tag{4.26}$$

and finally for the finished goods buffers

$$\begin{aligned} \dot{x}_9(t) &= u_7(t - t_{e7}) \\ \dot{x}_{10}(t) &= u_8(t - t_{e8}) \end{aligned} . \tag{4.27}$$

The outputs are equal to the states

$$y_i(t) = x_i(t) \quad \forall\, i \, . \tag{4.28}$$

In order to obtain a state-space model, the same approach (Kailath approach) is followed as in the example. At this point, a transfer function matrix $H(s)$ is constructed by taking the Laplace transformation of (4.25)—(4.28). The transformations are omitted, and $H(s)$ is given by

$$H(s) = \begin{bmatrix} -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s} & 0 \\ 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s} \\ 0 & 0 & -\frac{1}{s} & 0 & \frac{1}{s}e^{-(t_{e5}s)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{s} & 0 & \frac{1}{s}e^{-(t_{e6}s)} & 0 & 0 & 0 & 0 \\ \frac{1}{s}e^{-(t_{e1}s)} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{s}e^{-(t_{e2}s)} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{s}e^{-(t_{e3}s)} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{s}e^{-(t_{e4}s)} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s}e^{-(t_{e7}s)} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s}e^{-(t_{e8}s)} & 0 & 0 \end{bmatrix} \tag{4.29}$$

where $e^{-t_{ei}s}$ is approximated by a second order Padé approximation

$$e^{-t_{ei}s} = \frac{(t_{ei}s)^2 - 6t_{ei}s + 12}{(t_{ei}s)^2 + 6t_{ei}s + 12} \quad \text{for} \quad i = 1, \ldots, 8. \tag{4.30}$$

Given transfer function matrix (4.29), a minimal state-space model can be formed by using the Kailath approach described in Appendix A. The procedure is similar to the procedure described in Example 4.1.1. Since the matrices for the re-entrant flow line become quite large, only the A, B, C matrices that form the state-space model are given. The state-space approximation model of the re-entrant flow line with delay is given by the following expressions

$$\dot{x} = Ax + Bu \tag{4.31a}$$
$$y = Cx \, , \tag{4.31b}$$

where

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -30 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -300 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -45 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -675 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -27 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -18 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -108 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -18 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -108 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -27 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -45 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -675 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -30 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -300 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{4.32}$$

$$B = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -30 & 0 & -30 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -300 & 0 & 300 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -45 & 0 & -45 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -675 & 0 & 675 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-9 & 0 & 0 & 0 & -9 & 0 & 0 & 0 & 0 & 0 \\
27 & 0 & 0 & 0 & -27 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -18 & 0 & 0 & 0 & -18 & 0 & 0 & 0 & 0 \\
0 & 108 & 0 & 0 & 0 & -108 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & -18 & 0 & 0 & 0 & -18 & 0 & 0 & 0 \\
0 & 0 & 108 & 0 & 0 & 0 & -108 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -9 & 0 & 0 & 0 & -9 & 0 & 0 \\
0 & 0 & 0 & 27 & 0 & 0 & 0 & -27 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -45 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 675 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -30 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 300 & 0 & 0
\end{bmatrix}, \tag{4.33}$$

$$C = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}. \tag{4.34}$$

Note that the C matrix (4.34) in the above realization contains ones, while in the realization presented in Example 4.1.1 the B matrix (4.19) contained ones. The realization of the flow line example is a so called *controller* form realization, obtained from a right matrix fraction description (Appendix A.3); while the realization of the re-entrant flow line is a so called *observer* form realization, obtained from a left matrix fraction description. Both realization are interchangeable. Appendix A gives the procedure for right matrix fraction descriptions, the same procedure can also be given for left matrix fraction descriptions. The reason for presenting the state-space model in observer form instead of controller form, is that the outputs can directly be measured from the state vector, which is helpful when designing a controller.

This section is concluded by a short revision of the approach presented in this section. Similar as in Section 3.3, a state-space approximation model is designed based on modeling changes in buffers levels and linking in-flows and out-flows of buffers together. As an extension to the approach presented in Section 3.3, an new approach is described that explicitly includes the mean process times of machines in the approximation model. System equations (4.25)—(4.28) for the re-entrant flow line include the mean process times as delay. Since functions with delay cannot be included exactly—at least not by standard methods—in a state-space model, the delays are approximated by Padé approximations. This leads to a transfer function matrix (4.29) which is transformed into a state-space model using a procedure (Appendix A) derived from [Kai80]. An example is used to clarify the new approach. The example shows how a flow line, depicted in Figure 4.1, can be approximated by a state-space model that includes the mean process times of both machines. In the following section, a controller is designed for the state-space model of the re-entrant flow line by using Model Predictive Control (MPC). The section starts by given a short introduction on MPC and the possibilities of using MPC in this thesis.

4.2 Model Predictive Control

At the end of Chapter 3 two suggestions are made to improve the first approach to design the elements of the control framework considered in this thesis. The first suggestion is to include the mean process times of machines in the approximation model. A procedure to accomplish the first suggestion is presented in the previous section. The second suggestion is to design the controller by using MPC in stead of pole placement. This section explains how MPC works and shows how MPC is used to control the state-space approximation model. It is not attempted to provide an extensive presentation of the ins and outs of MPC. MPC is merely used as a tool, borrowed from control theory, applied to control the discrete event model. However, in order to apply MPC, the basic concepts must be grasped. Before continuing with the basic concepts, the reasons for adopting MPC are shortly recalled.

The reason for replacing pole placement for MPC is threefold. Since the capacity of the machines is bounded, and this information is not present in the controller nor in the approximation model, the situation could arise where the demands by the controller cannot be achieved by the discrete event model. Capacity violating input signals (referred to as *invalid* signals) were already observed during the simulations with pole placement. This occurred when the reference wip level became too low for the desired utilization. Since MPC handles inputs constraints, MPC might prevent invalid signals from occurring. Another reason for adopting MPC is for its capability to handle output (or state) constraints. By specifying output constraints it is possible to prevent input signals that would result in negative outputs (buffer levels). The third reason—besides simply being an alternative to pole placement—is that MPC naturally handles MIMO systems. MPC is an advanced control strategy which, for reasons mentioned above, is often used for industrial process control. An extensive treatment on MPC can be found in [Mac02]. Most relations and notation used in this thesis are obtained from lecture handouts of H. van Essen [Ess03]. This section continues to explain some concepts of MPC.

Model Predictive Control uses an internal model of the process to be controlled to predict the future outputs of that process. The prediction is based on a corresponding input trajectory, which is obtained by optimizing an objective function. The objective function is usually a quadratic cost function that minimizes future deviations between the system output and a reference trajectory, while preventing the input signal from changing inadmissible fast. The cost function is equipped with weighting matrices that can be used to express the significance of the elements in the cost function. A typical concept of MPC is the *receding horizon* concept. This means that MPC predicts the future outputs of the process over a number of samples ahead in time, called the *prediction horizon.* When a corresponding input signal over the prediction horizon is determined, only the *first* input sample is implemented. At the following sample time, again predictions are made over the full prediction horizon, and again only the first input is implemented. Equivalent to the prediction horizon, MPC also uses a *control horizon.* As mentioned, MPC determines an input signal over the prediction horizon that optimizes a cost function. The control horizon is the number of samples ahead for which the input signal is optimized. For the samples beyond the control horizon the input is assumed constant; therefore, the control horizon is always smaller or equal to the prediction horizon. A schematic view of the MPC implementation is presented in Figure 4.4. Basically MPC works as follows.

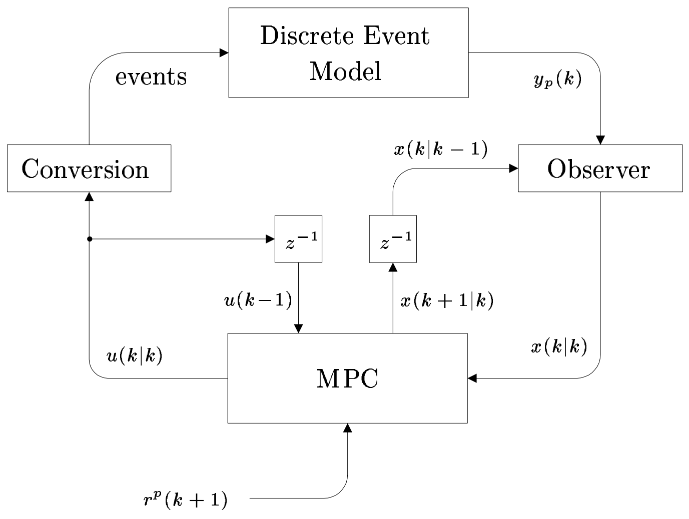


Figure 4.4: Schematic view of MPC implementation at time k

Consider the internal model of the process to be controlled—discrete event model—given by a linear discrete-time state-space model

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= Cx(k) \ , \end{aligned} \tag{4.35}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^i$, $y \in \mathbb{R}^o$. Further, Φ, Γ, and C are of appropriate size and D is zero. Let

$$\begin{aligned} p &= \text{prediction horizon,} \\ m &= \text{control horizon.} \end{aligned}$$

At time k (the present time), the output $y_p(k)$ of the discrete event model is known. Since MPC requires the full state vector $x(k|k)$ for the predictions, and often $x(k|k)$ cannot be measured directly from the process, an observer is used to reconstruct the state vector. The reconstruction is obtained from the process measurement $y_p(k)$ and an internal model of the process to be controlled. The observer used in this thesis is a standard state observer. A description of the state observer equations can be found in Appendix B.1. Next, the MPC controller predicts the response of the output over the prediction horizon. Let the notation for the predicted output be

$$y^p(k+1) = \begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+p|k) \end{bmatrix} . \tag{4.36}$$

Here $y(k+1|k)$ stands for the predicted output y at sample $k+1$ given information that is available at time k. The output prediction is based on

- current state of the process to be controlled $x(k|k)$
- past input $u(k-1)$
- reference trajectory over the prediction horizon

$$r^p(k+1) = \begin{bmatrix} r(k+1|k) \\ \vdots \\ r(k+p|k) \end{bmatrix}$$

- proposed future inputs over the control horizon

$$u^m(k) = \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+m-1|k) \end{bmatrix}$$

Determining the proposed future inputs forms the major part of the MPC controller. The future inputs over the control horizon are determined by optimizing a standard quadratic objective function [Pap00] given by

$$\min_{\Delta u^m(k)} \quad \sum_{l=1}^{p} \|y(k+l|k) - r(k+l|k)\|_Q^2 + \sum_{l=0}^{m-1} \|\Delta u(k+l)\|_R^2 \tag{4.37}$$

$$\text{subject to linear constraints} \quad A_{\text{con}} \Delta u^m(k) \leq b_{\text{con}}$$

Here the quadratic notation $\|x\|_Q^2 = x^T Q x$, with $Q > 0$. Note that the objective function uses Δu in stead of u as a design variable; Δu is called the *control move* defined by the following relations

$$\Delta u^m(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \tag{4.38}$$

$$u(k|k) = u(k-1) + \Delta u(k|k) . \tag{4.39}$$

Equation 4.37 is composed of two parts. The first part denotes the difference between the predicted outputs and the reference trajectory, weighted by matrix Q. The second part denotes the control moves, weighted by matrix R. Both weighting matrices are usually diagonal matrices, whose elements can be chosen relative to each other, depending on the significance of either part. Determining the values of Q and R is a matter of tuning: a subject briefly covered after the next paragraph.

MPC can be either constrained or unconstrained. In the unconstrained case, the solution of the quadratic cost function (4.37) simply results in a linear gain matrix, which can be computed off-line. The gain matrix only changes when the weighting matrices or the model changes. Unconstrained MPC, therefore, is not computational involving. In the constrained case constraints are added to objective function (4.37), and (4.37) has to be solved every sample using a Quadratic Program (QP) solver. The MATLAB® algorithm QUADPROG can be used. Constrained MPC, therefore, is more computational involving than unconstrained MPC. This thesis distinguishes two types of constraints: constraints on the input u, i.e. saturation, and constraints on the output y (selected from the state vector x). An extensive description of the equations and constraints that form the MPC controller is given in Appendix B.2.

Tuning MPC parameters

In the MPC equations a number of parameters are present that are used for tuning the controller. These tuning parameters determine the performance of the closed loop system and should be handled carefully. The following parameters are considered MPC tuning parameters: (i) length of the prediction horizon[3], and length of the control horizon, (ii) setpoint weighting matrix Q, and input weighting matrix R (4.37), (iii) sample time h, (iv) observer gain K_x (B.8). This thesis only briefly covers the subject of tuning; an extensive coverage on tuning MPC parameters can be found in [Mac02]. A systematic approach to tuning MPC parameters is not provided in this thesis, what is provided are merely basic guidelines and general influences of the MPC parameters.

The setpoint and input weighting matrices Q and R are considered relative to each other. If the setpoints weights are increased relative to the input weights, the control signal is likely to vary more (larger Δu) and the output approaches the reference trajectory faster. This situation, where $Q/R > 1$, leads to more aggressive control signals. When $Q/R < 1$, the control signal becomes tamer, and the reference trajectory is approached slower using less control effort. During the simulations a choice should be made between aggressive, more varying corrections, resulting in faster reduction of setpoint errors, or milder, less varying corrections, resulting in a slower but calmer setpoint approach. In the controlled discrete event model the output does not remain exactly equal to the reference value, yet the output oscillates around it (e.g. Figure 3.6). Since the output is represented by natural numbers, the output varies around the reference value $\pm$ some natural number. If the output is corrected every time it is below or above the reference value, the control signal might vary unnecessarily. Also, since the reference signal used during the simulations is rather tedious, namely a constant wip level in the buffers and a constant increasing level of finished goods (constant demand), it is presumed that milder control signals lead to better results. Therefore $Q/R < 1$ is used during the simulations. A guideline for determining the observer gain matrix K_x is that the eigenvalues of the observer dynamics $(\Phi - \Phi K_x C)$ (B.5) are faster or just as fast as the system eigenvalues.

A basic rule for setting the prediction horizon is that the controller must be able to observe the result of its actions in the model predictions. Therefore, as a rule of thumb, the prediction horizon must exceed the largest time constant or dead time period in the system.

[3]provided in number of samples

Another rule of thumb is that the control horizon is chosen between 1/6 and 1/3 of the prediction horizon. A smaller control horizon leads on the one hand to less computational effort, on the other hand a smaller control horizon leads to a slower system response. For fast varying reference trajectories a larger control horizon provides better results. Since the reference trajectory is opposite from varying fast, the control horizon can be chosen small. Furthermore, the length of the prediction and control horizon are expressed in number of samples, which themselves take a certain amount of time: the sample time. Therefore, if the prediction horizon p is to exceed the largest dead time (process time) t_e in the system, and h is the sample time, it must hold that $ph > \max(t_e)$. So, the sample time should be chosen relative to the largest dead time in the system, and the prediction horizon should be chosen in respect to the sample time. Recall that the sample time is already discussed during the simulation analysis of Figure 3.11. If the sample time is chosen too small, the controller reacts on too much stochastics or noise, while if the sample time is chosen too large the controller does not react at all. Also, if the sample time is chosen too small, the assumption that the discrete event model can be approximated by a continuous model might not be valid. Both arguments plead for a large sample time. However, in order to 'observe' the delays in the approximation model, the continuous state-space model (4.1) should be sampled with a small sample time. The larger the sample time used to discretize the continuous state-space model, the less 'visible' the delays are in the system matrices of the discrete state-space model. This sample time conflict is pinpointed closer in the next paragraph.

Sample time conflict

Determining a good sample time might seem trivial at first, yet a closer look reveals there is more to the sample time than meets the eye. A conflict arises when determining which sample time to use. On the one hand a large sample time should be used to avoid the controller from reacting on too much stochastics and to ensure that the approximation model is valid; on the other hand a small sample time should be used to ensure that the delays remain 'visible' in the discrete state-space model. The term 'visible' can be clarified a bit more. If the continuous state-space model with delays is discretized with a sample time larger than the delays present in the model, the system matrix Φ (4.35) becomes close to singular. This means that if the sample time is increased, the delays present in the continuous state-space model become less present in, or harder to recover from, the discrete state-space model. When Φ becomes close to singular the condition number approaches infinity, and Φ is said to be ill-conditioned [Hea97]. Performing matrix calculations with ill-conditioned matrices leads to unreliable and bad results. A number of problems that occur are the following: (i) the controllability matrix looses full rank, making the extra states introduced by the delay less controllable, (ii) the observer uses Φ^{-1} (B.8) as a gain matrix to reconstruct the state $x(k|k)$, since Φ is almost singular, Φ^{-1} approaches infinity. Figure 4.5 indicates the problem that occurs. The figure displays the step response of the continuous state-space model compared to the step response of the discrete state-space model for a small sample time $h = 0.2$ and a large sample time $h = 2$. The delay present between input u_1 and output y_5 is given by $t_{e1} = 2/3$ and is approximated by a second order Padé approximation. The discrete state-space model is obtained by using the MATLAB® function C2D with zero order hold. It is plausible that for a sample time $h > t_{e1}$ the influence of the delay is less visible than for a sample time $h < t_{e1}$.

In order to determine how far the sample time can be increased for the condition number

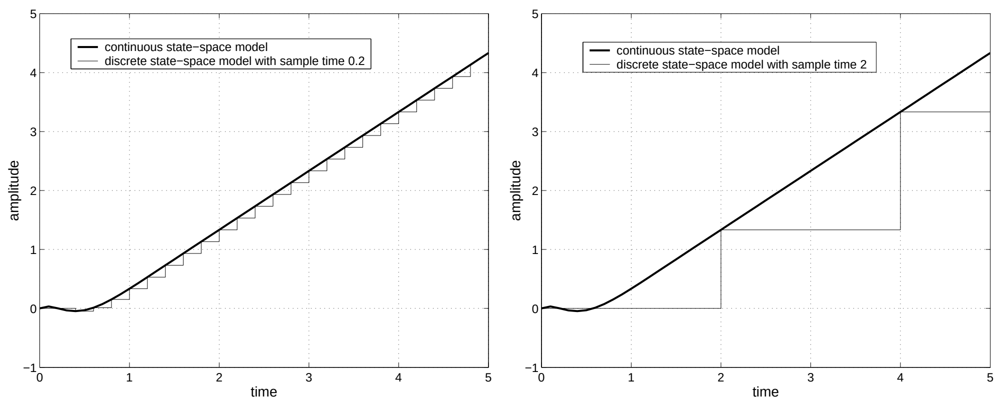


Figure 4.5: Step response of input u_1 to output y_5 of the approximation model of the re-entrant flow line with delay, where the delay between u_1 and y_5 is $t_{e1} = 2/3$, which is approximated by a second order Padé approximation. The step response of the continuous state-space model is compared to that of the discrete state-space model with sample time $h = 0.2$ and $h = 2$.

to remain acceptable, Figure 4.6 is created. Figure 4.6 displays the deterioration of the reciprocal of the condition number of Φ compared to EPS[4] plotted as a function of the sample time. For singular matrices, the condition number approaches infinity, so the reciprocal of the condition number approaches zero. MATLAB® uses a floating point relative accuracy of EPS $= 2.2204\ e^{-16}$ as default tolerance for singularity testing and rank determination. When the reciprocal of the condition number is smaller than EPS, the matrix is singular. Generally, well-conditioned matrices have a condition number of about one. Figure 4.6

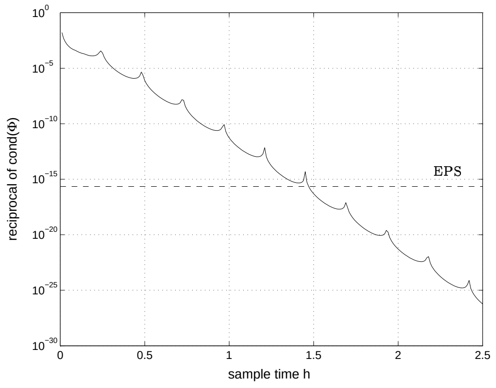


Figure 4.6: Reciprocal of the condition number of Φ (4.32) as a function of the sample time compared to EPS: the floating point relative accuracy of MATLAB® .

shows that if the continuous state-space model of the re-entrant flow line with delay is discretized with a sample time larger than approximately 3/2, the system matrix of the new discrete state-space model is singular. Therefore, during the simulations of the model with

[4]EPS is the floating point relative accuracy, a default MATLAB® tolerance for singularity testing

delay the sample time should be kept at least a considerable factor lower than 3/2. The suspicion that the approximation model might not be valid for such small sample times is kept in mind.

Section 4.2 is concluded by a short revision of the presented material in this section. A controller is designed using Model Predictive Control. The controller is designed for the approximation model that includes the mean process times of the machines. The equations to derive the MPC implementation are provided in Appendix B.2. An observer is used to reconstruct the state of the approximation model. The observer equations are provided in Appendix B.1. For the re-entrant flow line with delay, determining the sample time turns out to be conflicting. On the one hand a large sample is needed to ensure validity of the approximation model and to prevent too much stochastic effects in the controller, on the other hand a small sample time is needed to prevent the system matrix Φ from becoming singular. This conflict is further investigated during the simulations. The simulation results are discussed in the next section. First, the simulations of the MPC controller applied to the approximation model are presented. This simulation is used to investigated the influence of the tuning parameters. Next, the simulations of the discrete event model controlled by MPC are discussed, and the results obtained by MPC are compared to the results obtained by pole placement.

4.3 Simulations

Chapter 4 discusses a second approach to design the elements of the control framework. Two alterations to the first approach are presented. First, the mean process times of the machines are explicitly accounted for in the approximation model. Second, pole placement is exchanged for Model Predictive Control. Both alterations are explained, and the new approach can now be simulated and analysed. The same simulation layout is used as with pole placement, which is explained in Section 3.6. Python, MATLAB® , and χ are used to implement the elements of control framework. The discrete event model is modeled in χ, the MPC controller in MATLAB® , and Python (PYMAT) provides the interface between MATLAB® and χ.

This section first presents and discusses the simulations of the approximation model with delay controlled by MPC. It discusses the influence of the MPC tuning parameters as well as the influence of including time delays and using input- and output constraints. When these influences are clear, this section continuous by discussing the challenging case of applying the MPC controller to the discrete event model. The conversion steps used with MPC are the same as with pole placement; these steps are presented in Section 3.5. Finally, the performance of the MPC controller is compared the performance of the pole placement controller.

Approximation model with delay controlled by MPC

To test the MPC controller, first the approximation model with delay is controlled by MPC. As discussed earlier, MPC uses a number of tuning parameters. These tuning parameters influence the response of the controlled system to setpoint changes. Setting the values of these parameters can worsen or improve the reaction time, and dynamic response of the system; for some parameter settings, the system can even become unstable. Therefore, a close examination of the influence of the tuning parameters is useful. In the same sense, the influence of including time delays, input constraints, and output constraints is investigated. To explain and compare the different influences, simulation plots are used; considering the amount of parameters, hence the amount of plots, the simulation plots are placed in Appendix C to improve readability of the text. Generally, the response in buffer levels $y(k)$ and accompanying control signal $u(k)$ to a setpoint change from y_0 to y_{ref}, are compared for various parameter settings. The reference setpoint, initial buffer levels, and time delays are the following:

$$\begin{aligned}
y_{\text{ref}} &= \begin{bmatrix} y_B \\ y_S \\ y_{FG} \end{bmatrix} \\
&= \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} \end{bmatrix}^T \\
&= \begin{bmatrix} 6 & 6 & 6 & 6 & 2 & 2 & 2 & 2 & \lambda_a\, kh & \lambda_b\, kh \end{bmatrix}^T , \\
y_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T , \\
t_e &= \begin{bmatrix} 2/3 & 1/3 & 1/3 & 2/3 & 1/5 & 2/15 & 2/15 & 1/5 \end{bmatrix}^T .
\end{aligned} \tag{4.40}$$

In (4.40) the term $\lambda_a\, kh$ and $\lambda_b\, kh$ represent a constant increasing demand in the finished goods buffers. The demanded throughput for product a and b is respectively λ_a and λ_b. Furthermore, h denotes the sample time and k denotes the kth iterations.

The investigated influences are discussed one by one.

1. Influence of delay—Figure C.1

   The first influence that is investigated is the influence of including the mean process times as time delays in the approximation model. As discussed in Section 4.2, MPC uses an internal model of the process to be controlled to determine the next control action. Compare the following two situations: (i) the approximation model with delay is controlled by MPC with an internal model *with* delay (internal model = approximation model), (ii) the approximation model with delay is controlled by MPC with an internal model *without* delay (internal model $\neq$ approximation model). In this way the effect of introducing a mismatch between plant and model is investigated. The internal model without delay is equal to the discrete version of $\dot{x} = Bu, \; y = x$. MPC is considered unconstrained, all other parameters are held constant. The results are presented in Figure C.1. It is likely that the results in the case 'internal model $\neq$ approximation model' are worse than the results in the case 'internal model = approximation model'. Comparing Figure C.1(a) to Figure C.1(c) and Figure C.1(b) to Figure C.1(d) shows that a difference between the two situations is present, however this difference is very small. In the plant equal model case, control signals are a bit milder and setpoints are approached a bit calmer. Still, the difference is remarkably small; especially compared to what comes next.

2. Influence of setpoint and input weights Q and R—Figure C.2

   The MPC cost function (4.37) contains setpoint and input weighting matrices Q and R. The influence of both matrices are investigated by varying the fraction between the two. Generally, if Q is increased relative to R, setpoint errors are more penalized than variations of the inputs. Therefore, if $Q/R > 1$ (e.g. $Q = 10$,$R = 1$) setpoints are approached faster, using a more aggressive control signal, while if $Q/R < 1$ (e.g. $Q = 1$, $R = 10$) setpoints are approached smoother using a less aggressive control signal. This effect is confirmed in the simulations, which are plotted in Figure C.2. It is seen that for $Q/R = 0.1$, $y(k)$ approaches $y_{\text{ref}}(k)$ calmer and oscillations are much longer[5] than for $Q/R = 10$. Also, the absolute value as well as the deviations in $u(k)$ are considerably smaller for $Q/R = 0.1$ than for $Q/R = 10$.

3. Influence of prediction and control horizon lengths p and m, and the sample time h—Figure C.3

   If the length of the prediction or control horizon is changed, the system reacts either faster or slower to setpoint changes. The prediction horizon is determined in samples; each sample takes a certain amount of time: the sample time. The discussion on tuning parameters at the end of Section 4.2 noted that, in order for the controller to observe the effect of its actions, the prediction horizon should be larger than the largest time delay in the model. Therefore, it should hold that $ph > \max(t_e)$ or $p > 2/(3h)$. The question arises how the system responds to a setpoint change if the prediction horizon is varied while holding m, h and other parameters constant. Also, the idea that $ph > \max(t_e)$ suggests that the same system response can be achieved by either changing p or changing h. Figure C.3 in Appendix C provides an answer to these questions. Respectively Figure C.3(a), C.3(b), C.3(c), and C.3(d), show the wip levels for $p = 3$, $p = 5$, $p = 10$, and $p = 20$. The general observation is that for increasing p the output approaches setpoints faster with less overshoot, while using approximately the same—or a just a little more—control effort[6]. Remarkable is that for $p = 3$, the closed loop system is *unstable*. Unstable responses appear for prediction horizons with length $p < 2/(3h)$. When $p > 2/(3h)$ the system becomes stable again. When p is increased to $p = 5$, the unstable response disappears and long, slightly damped, oscillations occur. When p is further increased to $p = 10$, the response becomes more damped, resulting in overshoot rather than oscillations. Until for $p = 20$ even the overshoot is no longer visible. The idea that the system response has something to do with ph and the time delays in the approximation model is further supported by Figure C.3(e). Figure C.3(e) is created with $p = 10$ and $h = 0.4$, resulting in $ph = 4$, which is equal to $ph = 20 \times 0.2$ used in Figure C.3(d). Since ph is equal for both simulations, their responses should resemble each other. This resemblance is indeed present, which can be verified by comparing Figure C.3(d) and Figure C.3(e).

   Summarize the above: (i) the product of the prediction horizon and the sample time, ph, should be set considering the absolute values of the time delay in the approximation model, (ii) if the product ph is chosen to small the system can become unstable, (iii) for an increasing prediction horizon, the output approaches the setpoints faster with less overshoot.

4. Influence of input constraints and output constraints—Figure C.4

   A reason for adopting MPC is its ability to handle constraints. First, the influence of input constraints are investigated. Second, output constraints are briefly mentioned. The previous simulations discussed in this section showed that the control signal initially has a large amplitude which in time becomes smaller, until the output reaches the reference value, at which point all control signals become equal to the demanded throughput. While for the approximation model, large amplitudes in the control signal are no problem, for the discrete event model these signals can be a problem, since for the discrete event model the control signal represents demanded

[5]note the longer time scale

[6]control signals are not plotted

throughput, which is physically bounded by the utilization constraint. When an input constraint $0 < u(k) < u_{\max}$ is added to the MPC controller, and $u_{\max}$ is assumed to be $3/2$, the resulting system response to a setpoint change (note the larger time scale) is presented in Figure C.4. It is seen that it takes the trajectory considerably longer to reach the reference values. This is straightforward, since smaller $u(k)$ are used. Another observation are the individual heights of $u(k)$ between time 10 and 30. Since all reference wip levels are below the desired wip levels, it should hold that $\dot{x_i} > 0 \; \forall \; i$. Then, it follows that $u_9(k) > u_1(k) > u_5(k) > u_3(k) > u_7(k)$.

If output constraints are added to the MPC controller, a problems occurs. The problem is that if the constraint $0 < y(k) <$ Inf is added to the MPC controller, the QP solution becomes infeasible, and the system unstable. Reason for this might be the Padé approximations. If the step-response for the Padé approximated model is plotted (e.g. Figure 4.2) it is seen that negative amplitudes can also occur. Then, the following conflict causes the instability: negative values for $y(k)$ are explicitly caused by the Padé approximations in the internal model, while negative values for $y(k)$ are strictly forbidden by the output constraint in the MPC controller. This problem is not resolved, however it later turns out to be not so big a problem.

Besides creating the necessary feeling for the MPC tuning parameters, the most important conclusion drawn from the above analysis is that the influence of in- or excluding time delays in the internal model of the MPC controller is *very small* in comparison to the influence of other parameters, such as the prediction horizon and weighting matrices. Influence in this context means, the influence on the response in buffer levels and control signal to a setpoint change from y_0 to y_{ref}. If during the simulations of the discrete event model controlled by MPC, the influence of delay remains small, the internal model with delay can be discarded, and the shorter, more modest model without delay can be used instead. The first paragraph of the following section shows whether delays should be considered or not. Since the MPC controller has shown to perform well on the approximation model, the controller can next be applied to the discrete event model.

Discrete event model controlled by MPC

The previous section shows that the approximation model controlled by MPC yields the desired behavior. The influence of the tuning parameters is investigated, which shows that the influence of in- or excluding time delays in the internal model is considerably lower than the influence of, for instance changing the prediction horizon, or the weighting matrices. While the previous sections main concern is the response of the buffer levels and control signal to a setpoint change (ramp-up), the main concern of this section is not the transient behavior but more the long term (steady state) performance of the closed loop system. Performance is measured in reduction of flow time of lots through the system. Generally, a system has a better performance when for a constant throughput and constant wip-level a lower flow time is achieved. The reference wip level y_{ref} enforced on the system is given in (4.40). The main concern of this section is the influence of the MPC parameters, time delays, and constraints on the steady state performance of the MPC controlled discrete event model. This section starts by showing whether the influence of time delays remains small in case the MPC controller is used to control the discrete event model. Next, an appropriate prediction horizon is determined. Furthermore, two different settings of the setpoint versus input weights are discussed. The periodic signals, observed with pole placement are again observed with MPC. Naturally, the performance of the MPC controller is compared to the performance of the pole placement controller. As an appetizer, the demand—increasing wip level in the finished goods buffers—is changed from a constant demand, to a time-varying signal.

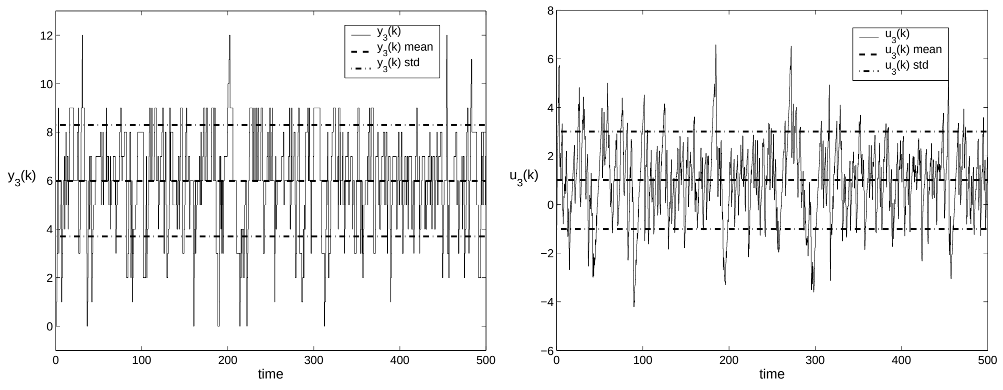
First, the influence of time delays in the internal model is tested. Consider again the two situations where (i) MPC uses an internal model with delay, and (ii) MPC uses an internal model without delay. The following table shows the, in Section 2.2 discussed performance measures, flow time φ, throughput δ, mean wip levels w, for a simulation of 5.000 lots where the first 1.000 lots are ignored.

	utilization 2/3		utilization 13/15	
	with delay	without delay	with delay	without delay
φ [unit time]	17.44	17.47	14.63	14.45
δ [lots per unit time]	1.00	1.00	1.30	1.30
w [lots]	17.33	17.45	18.89	18.56

Table 4.1: Difference between in- or excluding time delays in the internal model, on the steady state performance of the discrete event model controlled by MPC for a utilization of 2/3 and 13/15. Other parameters: $h = 0.2$, $p = 10$, $m = 2$, $Q = R = 1$, unconstrained, batchsize of three

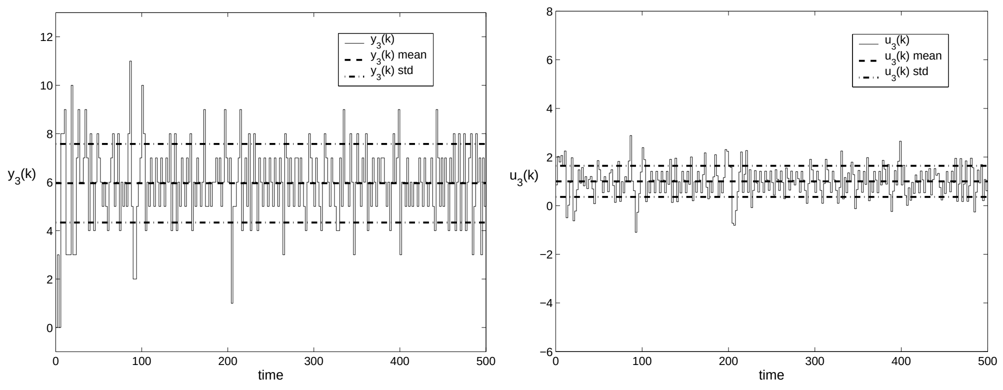
The table shows that the influence of the time delays in the internal model on the flow time, throughput, and wip level is not recognizable. When using an MPC controller, the explicit influence of the time delays on the steady state performance is unrecognizable and overshadowed by the influence of the MPC tuning parameters. For this reason, and the sample time conflict pointed out in Section 4.2, the time delays are discarded from the approximation model. Let the sample time conflict be explained a bit more. In order to prevent the system matrix Φ from becoming singular, the continuous state-space model needs to be sampled with a small sample time. While using a small sample time might not be a problem with the approximation model controlled by MPC, it is indeed a problem with the discrete event model controlled by MPC. Namely, if the sample is chosen too small, the stochastics in the discrete event model directly effect the control action, which means that the controller reacts too much to stochastic effects in the discrete event model. One could say that the controller overcompensates. For a constant demand in the steady state situation, the system does not need that much control; probably an open loop controller where the machines simply produce with a rate equal to the desired throughput would achieve reasonable results. Another complication with using a small sample rate is the noted validity issue of the approximation model.

Compare Figure 4.7 with Figure 4.8 to verify that a small sample time leads to a control signal with a large standard deviation, while for a larger sample time the standard deviation is smaller. The larger the sample time, the more the control signal approaches the reference throughput. If the steady state performance of the two sample times ($h = 0.2$ and $h = 2$) are compared, both sample times result in the same performance. This is understandable since y_{ref} fixes two of three variables in Little's equation. If the controller is able to achieve y_{ref}, the flow time follows automatically; so, simply comparing Little is not sufficient. Nevertheless, if the same performance is obtained with a larger sample time, and a larger sample time leads to less computational effort, it is better to use a larger sample time. It can be concluded that for an y_{ref} given by (4.40), using a small sample time is dubious and definitely not better than using a larger sample time.



(a) Buffer level (b) Control signal

Figure 4.7: A small sample time ($h = 0.2$) leads to a large standard deviation in the control signal. Other parameters: utilization is 2/3, $p = 10$, $m = 2$, $Q = R = 1$, unconstrained, internal model without delay, batchsize of three.



(a) Buffer level (b) Control signal

Figure 4.8: A large sample time ($h = 2$) leads to a small standard deviation in the control signal. Other parameters: utilization is 2/3, $p = 10$, $m = 2$, $Q = R = 1$, unconstrained, internal model without delay, batchsize of three.

From this point on, the internal model with delay is exchanged for the internal model without delay. Now, it is possible to choose an arbitrary sample time without the concern of singular matrices. This brings us back to setting the sample time in the situation of an approximation model without delay. In that case, a sample time of 10 was used; the same sample time is also used for the MPC controlled discrete event model. Discarding the delays t_e from the approximation model also negates the discussion of setting the prediction horizon according to $ph > \max(t_e)$. In case of an internal model without delay and an y_{ref} given by (4.40), the prediction horizon can be set arbitrarily. Therefore, a prediction horizon of length one might just as well be used; consequently, the length of the control horizon also becomes one: $p = 1$ and $m = 1$. In order to investigate the effect of using a short and a long prediction and control horizon on the steady state performance, Figure 4.9 is created. Again, the simulation layout is used where the utilization is increased by increasing the reference throughput λ from 0.8 with steps of 0.05 until 1.3, and from 1.3 with steps of 0.02 until 1.46. To ensure that the mean values are reliable, the simulation is repeated at least 30 times for every value of λ. During every simulation 20.000 lots are processed, of which the first 2.000 lots are ignored. Figure 4.9 shows that only for high utilizations, the difference

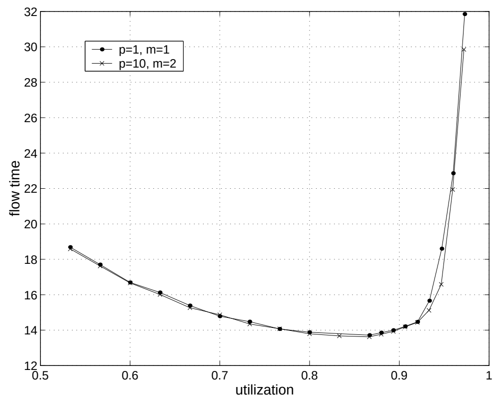


Figure 4.9: Compare the performance of two settings for p and m. Other parameters: $h = 10$, $Q = R = 1/10$, unconstrained, internal model without delay, batchsize of three.

between the short and the long horizons are visible. The reason for this could be that for longer prediction horizons, the control signal generally varies less. When the utilization becomes too high for the setpoints to be maintained, a less variating control signal tends to excite the system less, and the performance is maintained a little longer.

It is next investigated, whether changing the fraction between the setpoint weights Q and the input weights R has any effect on the steady state performance of the discrete event model controlled by MPC. In order to do so recall that generally a faster response using a more aggressive control action is achieved when more weight is put on Q than on R. Also recall that the buffer levels never remain exactly equal to the reference values, but rather oscillate around them $\pm$ some natural number. Since these oscillations occur due to the inherent discrete nature of the manufacturing system, less effort should be used to correct the buffer levels toward the setpoints than with the approximation model controlled by MPC. Figure 4.10 compares the steady state performance for two settings of Q/R. The figure shows that overall no difference is present. However, for utilizations larger than 0.9 $Q/R = 1/10$ performs slightly better. The overall lack of difference is again caused by the fact that Little's equation is completely determined by y_{ref}. For utilizations larger than 0.9, the reference levels can no longer be reached, and therefore become unrealistic. In the case of unrealistic reference levels, the situation that attempts to reach the unrealistic levels less ($Q/R = 1/10$), results in a slightly better performance.

Until now MPC is considered to be unconstrained. One of the reasons for adopting MPC is its ability to handle *input constraints*. The original reason for using input constraints is that machines have a maximal physical capacity which is not present in approximation model nor controller. Therefore, it might be possible that the controller demands a capacity

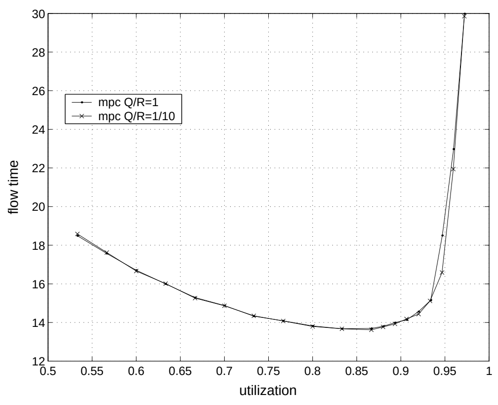
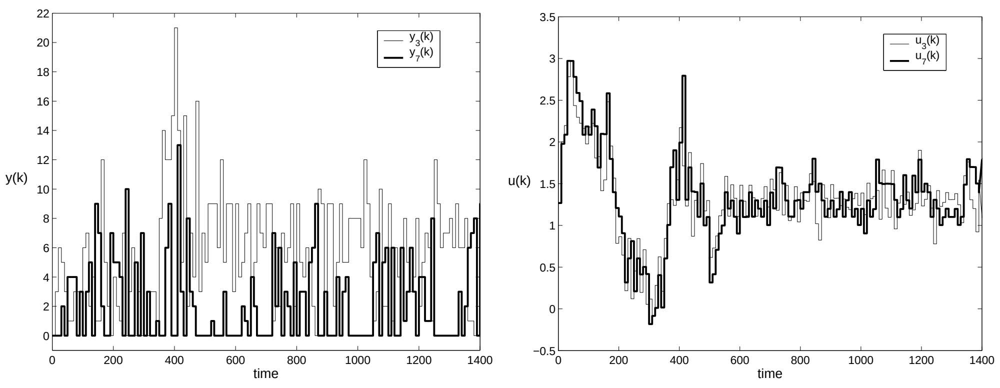


Figure 4.10: Compare the performance of two settings for Q/R. Other parameters: $h = 10$, $p = 10$, $m = 2$, unconstrained, internal model without delay, batchsize of three.

violating throughput. Is it actually necessary to use input constraints? Clearly, for low utilization the controlled system performs excellent without input constraints. Even for high utilizations, the deteriorating controller is not primarily caused by the unrealistic control signals. These unrealistic signal are merely a response to the fact that the reference wip levels are too low for the desired throughput (demand). As long as the demand in the finished goods buffer is realistic, surely the controller never demands an unrealistic throughput from the machines. Occasionally, due to stochastic or transient effects in the discrete event model, the controller indeed demands more throughput than can be processed, however, this excess demand is corrected during the following samples. Figure 4.11 shows that indeed the situation occurs where the controller first asks too much and later corrects the excess demand. In the end, if the reference wip levels in the buffers are realistic, the average demanded throughput by the controller never exceeds the demanded throughput in the finished goods buffers. So why use input constraints?

The MPC controller can now be compared to the pole placement controller. The first aspect that is compared are the steady state performance measures: flow time and wip versus utilization. Again, the well known simulation layout is used where the utilization is increased by increasing the reference throughput λ. The same parameters are used as before (4.40), except for the varying λ. Figure 3.9 compares the MPC controller to the pole placement controller.

It is seen that overall the MPC controller has a slightly better performance. For low utilizations this difference is not that interesting, since both controllers are able to perform according to y_{ref}, which completely fixes Little's equation. Only for high utilizations the difference becomes apparent. Actually, for a prediction and control horizon of one, unconstrained MPC is not that different from pole placement. In the unconstrained case, both controllers reduce to a matrix multiplication. For MPC, the gain matrix is the solution of a cost function with weighting matrices, while for pole placement, the gain matrix is the result of placing the poles at certain locations. The other aspects of the control framework are mostly identical. The observer is an identity matrix in the case of an approximation model without delay. Further, the same conversion algorithm is used with MPC as with pole placement. Of course the same discrete event model is used. Except for the gain matrices, the only difference is the way demand is modeled. In the case of pole placement, lots are inserted into the line at an uncontrollable constant rate; demand is represented by the height



(a) Buffer level (b) Control signal

Figure 4.11: Excess demand in $u(k)$ is later corrected by the controller. Other parameters: utilization is 13/15, $h = 10$, $p = 1$, $m = 1$, $Q = R = 1$, unconstrained, internal model without delay, batchsize of three.

of this rate. In the case of MPC, the insertion of lots in the line is a controllable input, and demand is represented by a constant increasing level of finished goods. Both approaches realize the same demanded throughput, only MPC uses a controllable generator process, while pole placement uses an uncontrollable one. The fact that MPC overall performs slightly different could, though unlikely, be caused by this difference in modeling demand. A more likely reason that MPC performs better, especially for high utilizations, could be that MPC compromises between following the setpoints and using a variating input signal. When the setpoints can no longer be reached, this effect is less amplified by MPC than by pole placement. Actually, the important question that should be asked here is whether the difference between pole placement and MPC can be visualized at all by comparing the *steady state* performance such as flow time and wip. Are flow time and wip the relevant quantities to compare? When controller performance is compared, is it not the transient behavior that makes the difference?

For now it is concluded that both pole placement and MPC are able to let the systems output follow a reference signal given by (4.40). As an appetizer, let the demand for once, instead of constant, be given by some sinusoidal function. Then, Figure 4.13 displays the responding buffer levels and control signal. During the simulation 10.000 lots are processed. The reference output and initial condition are given by

$$\begin{aligned}
y_{\text{ref}} &= \begin{bmatrix} y_B \\ y_S \\ y_{FG} \end{bmatrix} \\
&= \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} \end{bmatrix}^T \\
&= \begin{bmatrix} 6 & 6 & 6 & 6 & 2 & 2 & 2 & 2 & \frac{1}{2a}(\sin(akh) + akh) & \frac{1}{2a}(\sin(akh) + akh) \end{bmatrix}^T , \qquad (4.41) \\
\text{with} \quad a &= \frac{1}{400}, \quad k \text{ is the } k\text{th sample, } h \text{ is the sample time, and} \\
y_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T .
\end{aligned}$$

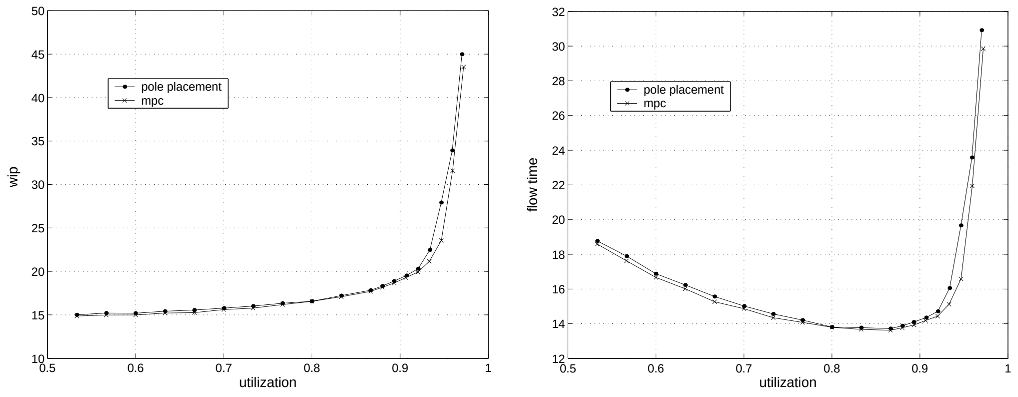
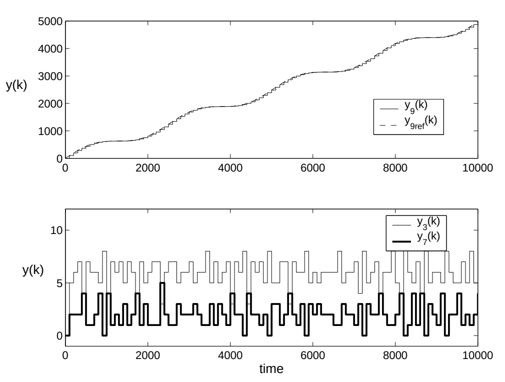


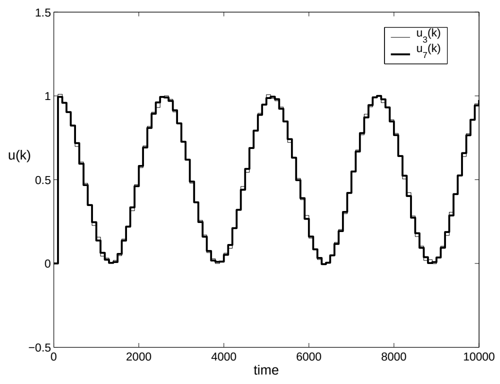
Figure 4.12: Mean wip and mean flow time of lot type a as a function of the utilization. Other parameters: $h = 10$, $p = 10$, $m = 2$, $Q/R = 1/10$, unconstrained, internal model without delay, batchsize of three

Figure 4.13 shows that the MPC controller can also let the discrete event model follow a sinus shaped demand while maintaining buffer levels equal to six and two. The idea that the sample time should be set with respect to the frequency of the demand is encouraged by this simulation. During this simulation a sample time of a 100 is used. It is interesting to see, that even with a prediction and control horizon of one, unconstrained MPC still manages to follow the setpoints. Figure 4.13 arises questions whether similar results would be obtained using pole placement. An extensive study into time varying demand, choosing an adequate controller, and setting the sample time are topics for future research.

The simulations of the discrete event model controlled by MPC are completed, and can be concluded with a short revision of the most important observations. The simulations show that MPC can be used to control the discrete event model. The influence of the MPC tuning parameters is clearly visible when the approximation model is controlled by MPC, but if the discrete event model is controlled by MPC, the influence is no longer distinguishable in a similar matter. While for the controlled approximation model unstable situations occur, unstable situations are never observed for the controlled discrete event model. Since time delays in the approximation model are shown to have an indistinguishable effect on the steady state performance, and the control of the approximation model with time delays gave rise to the sample time conflict, the time delays are removed from the approximation model. The necessity for input constraints is questionable, since the controller on average never demands more than the demanded throughput. Output constraints seem to have little effect in the investigated manufacturing system, however it is suspected that outputs constraints do effect transient behavior in long flow lines. The question is raised whether the controller's concern should be transient behavior or steady state behavior. It is suspected that the actual strength of using control theory to control discrete event models, lies in the fact that transient behavior can be manipulated, e.g. ramping a factory up or down. Further, the unconstrained MPC controller with a prediction horizon and control horizon of one, is not very different from the pole placement controller. Finally, it is shown that MPC can also handle time-varying demand.



(a) Buffer level



(b) Control signal

Figure 4.13: Buffer levels and controls signal for a sinus shaped demand in the finished goods buffers. Other parameters: $p = 1$, $m = 1$, $h = 100$, $Q = \text{diag}[1 \cdots 1\ 10\ 10]$, $R = \text{diag}[10 \cdots 10]$, unconstrained, internal model without delay, batchsize of three.

4.4 Discussion

Chapter 4 presents a second approach to control the discrete event model by means of the control framework discussed in Chapter 1. Two main alterations are suggested to the first approach, discussed in Chapter 3: (i) the mean process times of the machines are included in the approximation model as time delays, and (ii) MPC is used in stead of pole placement.

The discussion at the end of the previous chapter, Section 3.7 states that by adding more realistic elements to the model or controller a better performance is expected. Also, one only want to add those elements that contribute significantly to the performance of the system. Section 3.7 does not state what is meant by performance. The MPC simulations of the discrete event model show that for the steady state performance, as discussed in Section 2.2, the two alterations do not contribute significantly: the model without delay predicts just as well as the model with delay, and since on average the controller never requests more throughput than the desired demand—provided that the reference wip levels are realistic—input constraints remain questionable. However, if instead of steady state behavior the controller's concern is transient behavior, the difference and individual strengths of the controllers might be far greater. Also, the fact that the reference trajectory is, in all but one situation, chosen rather tedious—a fixed wip in the buffers and a constant increasing demand in the finished goods buffers—does not contribute to distinguishing differences between controllers. Under the given circumstances, both MPC and pole placement perform adequate. Therefore, it can be concluded that if the approximation model is as simplistic as it is, and the system is excited with a constant demand, pole placement can just as well be used as MPC.

The remainder of this thesis consists of a conclusion and discussion of the total thesis, followed by recommendations for future research.

Chapter 5

Conclusion, Discussion, and Future Research

This thesis investigates a control framework that opens up the possibility to control manufacturing systems using control theory. The control framework forms an alternative to heuristic control methods, such as conwip, push, or pull, that are currently often used to control manufacturing systems. The manufacturing system under consideration is a four step re-entrant manufacturing line with batching, which is based on a model by [Dia00]. In this thesis feedback control, such as pole placement and model predictive control (MPC), is used to control a discrete event model of the re-entrant manufacturing line. In the Systems Engineering Group[1] manufacturing systems are represented by discrete event models. Since discrete event models are essential different models than continuous or discrete time models used by control theory, a direct coupling between control theory and discrete event models is not possible. The essential difference is that discrete event models are driven by events, while most models for control theory are described by differential or difference equations. A possibility to work around these differences is provided by the control framework as explained in Section 1.2.

The control framework states the following. Derive a model for a manufacturing system—referred to as the approximation model—to which standard control theory can be applied. Next, design a controller for the approximation model. The controller and the approximation model are uncoupled, and the controller is coupled to the manufacturing system—represented by a discrete event model—via two conversion steps. The first conversion step translates the controller output into events, the second conversion step translates the output of the discrete event model into an input for the controller. Summarizing, the control framework consists of the following elements: (i) discrete event model (manufacturing system), (ii) approximation model, (iii) controller, and (iv) two conversion steps.

Two possible approaches to design the elements of the control framework described above are investigated in this thesis. Both approaches use the same discrete event model, and the same conversion steps, but use different approximation models (fluid models) and different controllers. The first approach uses a fluid model in the state-space form, and a controller designed by pole placement. The second approach uses a similar fluid model only with

[1]http://se.wtb.tue.nl

approximated time delays, and a controller designed by MPC instead of pole placement. The conclusions and most important observations from investigating these approaches are discussed next. Recommendations for future research are presented at the end of this chapter.

Approach I: fluid model controlled by pole placement

As a first approach all elements of the control framework are chosen as modest as possible. The approximation model is a fluid model in state-space form, the controller is designed using pole placement, and conversion is only performed on one side of the framework: between controller output and discrete event model input. This paragraph summarizes the most important issues of each element of the control framework. Generally, a fluid model is based on the assumption that material flows through the system can be considered as continuous flows. A state-space model of the form $\dot{x} = Bu$ can be used to describe the material flows by simply stating that the change in buffer contents equals the inflow rate minus the outflow rate. The system equations are obtained by using the property that buffers are fed from upstream buffers and feed downstream buffers. For the controller pole placement is used; all poles are placed in the origin (deadbeat controller), which results in a matrix multiplication with a highly transparent gain matrix. The reference trajectory is a setpoint step change, and a desired throughput. In order to connect the controller to the discrete event model, a conversion algorithm is used. The conversion algorithm translates a continuous control signal into events for the discrete event model, while the intentions of the controller are guaranteed. The control signal is considered a desired throughput over the following sample time; integrating the desired throughput results in a desired production. By answering questions such as, should a machine remain idle or start processing, and which lot should be processed next, the conversion algorithm equalizes the desired production of the controller and the actual production of the discrete event model. The results of the described approach are discussed next.

When the above described approach is simulated, the results are encouraging. First of all, pole placement, through the suggested control framework, can indeed be used to control the output of the four step re-entrant flow line towards a setpoint step change. Initially the trajectories are driven toward the setpoints, once the setpoints are reached, the mean control signals become equal to the desired throughput. Since the buffer levels are naturals, buffer levels never remain exactly equal to the setpoints but rather oscillate around them. A remarkable observation is that the buffer levels and control signals seem to display periodic patterns which are effected by the utilization. For low utilizations $0.5 < \rho < 0.8$ the periodic patterns (often period three) are clearly present. When the utilization is increased $0.8 < \rho < 1.0$ these periodic patterns gradually disappear. A cause for the disappearing periodic patterns can be that stochastic effects are more present in highly utilized systems; meaning, that for an increasing utilization the probability to observe high buffer levels at the sample times increases.

The boundaries of the current approach are investigated. As the utilization is pushed further towards one, the reference wip levels for the discrete event model become too low for the increasing utilization. Consequently, the controller observes an error between the reference wip and actual wip. To correct the error, the controller demands more throughput from the discrete event model. Nevertheless, the error remains, since the reference wip levels are too low. While the controller keeps observing an error, and keeps demanding throughput that would remove the error, the integrator in the conversion algorithm keeps integrating. As long as the error remains, the integrator causes the difference between the desired production and actual production to remain increasing. This process is called integrator windup, and is caused by actuator saturation. Avoiding integrator windup, by using an anti-windup scheme, is a point where the conversion algorithm, or maybe the controller, can be improved.

The final effect investigated in the first approach is the effect of increasing the sample time. It is seen that when the sample time is increased, the flow time decreases. This seems to be a good result, a lower flow time and less wip indicates a better performance. However, while the performance of the discrete event model increases, the performance of the controller decays because the reference wip levels are no longer reached. It remains a point of investigation, why the controller with a larger sample time achieves, of all possible flow times, precisely lower ones. Determining an appropriate sample time is further investigated during the second approach.

The first approach successfully showed that through the control framework it is possible to control discrete event models using control theory. As a comparison to the first approach, a second approach is investigated. The most important observations of the second approach are summarized next.

Approach II: fluid model with delay controlled by MPC

During the second approach two elements of the control framework are modified. First, the mean process times of lots are included in the approximation model as time delays. Second, the controller is designed using model predictive control instead of pole placement. The reason for including mean process times in the approximation model is, that since process times are present in the discrete event model and not in the approximation model, the approximation model assumes lots to move instantaneously from one buffer to another, while in the discrete event model lots move from one buffer to another across a machine that processes them for a certain amount of time. The reason for replacing pole placement by MPC is—besides simply being an alternative—that MPC can handle constraints on inputs and outputs. Input constraints could be used to avoid control signals that exceed the machine capacity. Output constraints could be used to prevent control signals that result in negative buffer levels. Also, MPC naturally handles multiple input multiple output (MIMO) systems.

A state-space approximation model is derived by including the mean process times in the system equations as time delay or dead time. Since, these delays are present in the system equations, a state-space model cannot be derived as easily as before. In this thesis the delays are approximated using 2nd order Padé approximations. Once the delays are approximated, a state-space model can be derived. State-space models of systems with a large number of inputs and outputs can become quite large and unpleasant to work with. Techniques from Kailath [Kai80] or MATLAB® can be used to derive a minimal state-space realization, which means that the state-space model has the smallest possible size. The Kailath approach, presented in Appendix A, provides a method to derive a minimal state-space model from a given transfer function matrix. The MATLAB® approach uses the function SS with the option 'min'. The Kailath approach in comparison to the MATLAB® approach creates matrices with a more transparent structure, especially for high order models (i.e. higher than 10) the function SS option 'min' destroys any recognizable patterns in the realization matrices, and it is recommended to use the Kailath approach.

To discuss the results obtained with MPC, recall that MPC uses a large number of tuning parameters that should be handled carefully. Setting the values of these parameters can worsen or improve the response of the closed loop system; for some parameter settings the system can even become unstable. The following tuning parameters are investigated: (i) length of prediction horizon p, and length of control horizon m, (ii) setpoint weighting matrix Q, and input weighting matrix R, (iii) sample time h. The effect of the individual tuning parameters on the response to a setpoint step change for the approximation model controlled by MPC is investigated, and it is seen that the transient behavior is indeed effected by these parameters in the expected way. If on the other hand the discrete event model is controlled by MPC, the parameters do not effect the transient behavior in a similar matter. For some sample time setpoint combinations, the discrete event model controlled by MPC even displays *unstable* behavior, which is not observed when the approximation model is controlled by MPC. The discrete event model controlled by MPC indeed reacts differently to setpoint changes than the approximation model controlled by MPC. More insights are needed in stability issues for the MPC controlled discrete event model for changing parameter settings and sample times.

The above showed that the transient behavior of the approximation model as well as the discrete event model is effected by the MPC tuning parameters. If instead of the transient behavior the steady state behavior is investigated, it is seen that for the discrete event model controlled by MPC the tuning parameters only effect the steady state behavior for high utilizations ($\rho > 0.8$). Steady state behavior, in this case means, steady state performance measured in Little's variables flow time and wip. Generally, for larger prediction horizons, and for a smaller fraction between setpoint weights and input weights Q/R, the performance is maintained a little longer—lower flow times are achieved—when the utilization is increased. A reason for this could be that for the larger prediction horizon and a smaller fraction Q/R, the control signal generally varies less, causing the system to be less excited which causes the performance to be maintained a little longer.

A conflict involving the sample time arises when the approximation model with delays is discretized. On the one hand, a large sample time is needed to ensure validity of the approximation model, prevent the controller from reacting too much to stochastic effects, and in some cases prevent unstable behavior, on the other hand, a small sample time is needed to prevent the system matrix Φ (4.35) from becoming singular. Due to the sample time conflict, and the fact that the time delays do not influence the steady state performance of the discrete event model controlled by MPC, the time delays are removed from the approximation model. Therefore, the same fluid model is used as during the first approach.

The use of input constraints remains questionable. Input constraints could be used to prevent constraint violating control signals. However, for realistic reference wip levels, the controller on average, never demands more throughput than the desired throughput. The constraint violating control signals, that appear for high utilizations, are merely a response to the fact that the reference wip levels are too low for the desired utilization. More logical would be to attribute the consequences of unrealistic reference wip levels to the approximation model, since the approximation model does not model the asymptotic effect that an increasing utilization has on the buffer levels. To accurately model the asymptotic effect between utilization and buffer levels is a topic for future research.

MPC can, besides a constant demand, also handle a time-varying demand, e.g. some sinusoidal reference signal in the finished goods buffers. Time varying reference signals can also be enforced onto any other output of the discrete event model. The sample time can now be set according to the frequency of the reference signal.

MPC compared to pole placement

If the discrete event model controlled by pole placement is compared to the discrete event model controlled by MPC, it is seen that both pole placement and MPC are able to drive the system output from y_0 to y_{ref} and keep it there. In both cases, control theory has been used to control the discrete event model towards a setpoint step change. Comparing the steady state performance measures, flow time and wip, shows that for low utilizations both controllers perform equally. The reason is that for low utilizations, both controllers are able to perform according to y_{ref}, and y_{ref} fixes two of three variables in Little's equation. For high utilizations MPC performs better than pole placement. This difference is caused by the fact MPC optimizes an objective function to determine the control action. Through tuning parameters and weighting matrices, MPC possesses the subtlety to compromise between following setpoints and using a fluctuating input signal. For larger prediction and control horizons, MPC is able to use a more gentle control signal. Pole placement on the other hand does not possess this subtlety, and attempts to correct every error via an aggressive control action. For high utilizations, the subtlety of MPC is necessary to prevent the conversion algorithm from winding up too fast which causes the performance to decay since the intentions of the controller are inadequately translated into events. When the setpoints are no longer attainable, the error between reference wip and actual wip is less amplified by MPC than by pole placement.

It is suspected that a clearer distinction between the performance of both controllers is obtained when instead of steady state behavior, transient behavior is compared. While current heuristic control policies are mostly concerned with optimizing steady state performance—obtaining low flow times at high utilizations—, the proposed control framework is applicable in a broader area. The control framework can possibly be very useful for obtaining controllers that optimize transient behavior. Consider for instance the situation where a factory needs to be switched from one production scheme to another, while losing the least amount of production. Or consider the situation where the factory is ramped up or down, and the new production levels should be reached as fast as possible with little overshoot. In these situations transient behavior is important, and the control framework probably shows its strength. However, in order to observe these effects, it might be necessary to adopt an approximation model, that includes the asymptotic relation between buffer levels and utilization.

The conclusions, discussed in this chapter, point out the boundaries of the investigated approaches. By exploring these boundaries areas for future research are brought to the attention. These areas form recommendations for future research, which are summarized in the following paragraph.

Future research

The following points are considered to be interesting topics for future research.

- Design approximation models that include the asymptotic effect that increasing the utilization in a stochastic manufacturing system has on the buffer contents (wip) and consequently the flow time.
- Improve the conversion algorithm for highly utilized systems, by e.g. including an anti-windup scheme. Literature covering anti-windup such as [Kap98], [Kot94], or [Che01] could be interesting.

- Investigate and manipulate transient behavior of manufacturing systems by using the current control framework. Somehow include the specific effect machines have on the transient behavior in the control framework.
- Determine stability conditions for the discrete event model controlled by control theory. The situation may very well occur that the controlled approximation model is stable—or at least appears to be—, while the controlled discrete event model displays unstable behavior. It is suspected that the sample time has something to do with these issues.
- Add more realistic elements to the discrete event model, such as finite buffer levels and failure behavior, and deal with these elements either in the approximation model, the controller or the conversion steps.

Bibliography

[Ban97] J. Banks and J.G. Dai. Simulation studies of multiclass queueing networks. *IIE Transactions*, 29(3):213–219, 1997.

[Bee00] D.A. van Beek and J.E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of chi. *Control Engineering Practice*, 8(1):81–91, 2000.

[Che01] M. Cherukuri and M. Nikolaou. Connection between model predictive control and anti-windup control schemes. *Automatica*, re-submitted (2001). http://www.chee.uh.edu/faculty/nikolaou.

[Dai96] J.G. Dai and G. Weiss. Stability and instability of fluid models for re-entrant lines. *Mathematics of Operations Research*, 21:115–135, 1996.

[Dia00] I. Diaz-Rivera, D. Armbruster, and T. Taylor. Periodic orbits in a class of re-entrant manufacturing systems. *Mathematics of Operations Research*, 25(4):708–725, November 2000.

[Ess03] H. van Essen. Homepage, April 2003. http://www.wfw.wtb.tue.nl/control/harm1.html, http://www.industrialdesign.tue.nl/.

[Fra94] G.F. Franklin, J.D. Powell, and A. Emami-Naeini. *Feedback control of dynamic systems.* Addison-Wesley, 3rd edition, 1994.

[Gro96] M.P. Groover. *Fundamentals of modern manufacturing: materials, processes, and systems.* Prentice Hall, Englewood Cliffs, 1996.

[Hea97] M.T. Heath. *Scientific Computing: an Introductory Survey.* The McGraw-Hill Companies, 1997.

[Hof01] A.T. Hofkamp. Python from χ. Technical report, Eindhoven University of Technology, Department of Mechanical Engineering, Systems Engineering Group, 2001. http://se.wtb.tue.nl/documentation.

[Hop00] W.J. Hopp and M.L. Spearman. *Factory Physics: Foundations of Manufacturing Management.* Irwin/McGraw-Hill, New York, 2nd edition, 2000.

[Int03] Intersil. Lexicon of semiconductor terms, February 2003. http://rel.intersil.com/docs/lexicon/index.html.

[Kai80] T. Kailath. *Linear Systems.* Prentice-Hall, Englewood Cliffs, N.J. 07632, 1980.

[Kap98] N. Kapoor, A.R. Teel, and P. Daoutidias. An anti-windup design for linear systems with input saturation. *Automatica*, 34(5):559–574, May 1998.

[Kle01] J.J.T. Kleijn and J.E. Rooda. χ *Manual.* Eindhoven University of Technology, Department of Mechanical Engineering, Systems Engineering Group, 2001. http://se.wtb.tue.nl/documentation.

[Kot94] M.V. Kothare, P.J. Campo, M. Morari, and C.N. Nett. A unified framework for the study of anti-windup designs. *Automatica*, 30(12):1869–1883, December 1994.

[Kum93] P.R. Kumar. Re-entrant lines. *Queueing Systems*, 13:87–110, 1993.

[Kum01] S. Kumar and P.R. Kumar. Queueing network models in the design and analysis of semiconductor wafer fabs. *IEEE Transactions on Robotics and Automation*, pages 1–14, May 2001.

[Lit61] J.D.C. Little. A proof for the queueing formula $l = \lambda w$. *Operations Research*, 9:383–387, 1961.

[Mac02] J.M. Maciejowski. *Predictive Control with Constraints.* Pearson Education - Prentice Hall, 2002.

[Pap00] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design.* Cambridge University Press, 2000.

[Pol98] J.W. Polderman and J.C. Willems. *Introduction to Mathematical Systems Theory: A Behavioral Approach.* Springer-Verlag, New York, 1998.

[Rem03] B. Rem and D. Armbruster. Control and synchronization in switched arrival systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 13(1):128–137, March 2003.

[Roo00] J.E. Rooda. Modelling industrial systems. Lecture notes, Eindhoven University of Technology, Department of Mechanical Engineering, Systems Engineering Group, 2000.

[Sem03] Sematech. Sematech dictionary, February 2003. http://www.sematech.org/, http://www.sematech.org/public/publications/dict/index.htm.

[Ste99] A. Sterian. Pymat — python to matlab interface. http://claymore.engineer.gvsu.edu/~steriana/Python/pymat.html, 1999.

[Wei88] L.M. Wein. Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 1(3):115–130, August 1988.

[Zei00] B.P. Zeigler, H. Praehofer, and T. Gon Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems.* Academic Press, London, 2000.

Appendix A

State-Space Realizations of Minimal Order

To achieve a system that is suitable for numerical computations and controller design, a state-space formulation rather than an input-output formulation of a transfer function matrix is used. A transfer function matrix can be transformed into state-space equations. This is called making a state-space realization of a transfer function matrix, or shortly making a realization. Realizations can be derived in various ways; most realizations do not necessarily require a minimal number of states. Realizations that do require a minimal number of states will be called state-space realizations of a minimal order.

When dealing with systems with large numbers of inputs and outputs, realizations can be far from minimal and therefore unpleasant to work with. This leads to the question what the minimal possible order of a state-space realization of a given transfer function matrix is. While minimal realizations for SISO systems are classical classroom examples, minimal realizations of MIMO systems deserve a bit more attention. A well developed theory for deriving various canonical forms and minimal realizations for MIMO systems is described in Chapter 6 of [Kai80].

This section deals with deriving the minimal order of a MIMO state-space realization, and, given this order, forming a controllable and observable realization following steps from [Kai80]. This procedure uses a few concepts from the field of polynomial methods and control that require a short introduction. First, the terms *transfer function matrix* and *Matrix Fraction Description* are introduced. Next, the *Smith* and *Smith-McMillan form* of a polynomial matrix are explained. The Smith-McMillan degree is the minimal degree of any state-space realization. Section A.6 deals with the *Hermite form* of a matrix. After these introductions, the procedure to derive the minimal realization is described in Section A.7. Finally, Section A.8 contains an example where this technique is applied to the Padé approximation model of the discrete event model described in Section 4.1.

The realization techniques presented in this appendix assume a general continuous-time state-space model of the following form.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \, , \end{aligned} \tag{A.1}$$

where $\dot{x}$ denotes the derivative of x, x is an $(n \times 1)$ state vector, u is an $(m \times 1)$ input vector, and y is a $(p \times 1)$ output vector. Consequently, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. Here n denotes the order of the state-space model. Also,

$$\begin{aligned} n &= \text{number of states,} \\ m &= \text{number of inputs,} \\ p &= \text{number of outputs.} \end{aligned}$$

A.1 Transfer Function Matrix

For a MIMO system with n states, m inputs, and p outputs, the transfer function matrix is given by

$$H(s) = \begin{bmatrix} H_{11}(s) & H_{12}(s) & \dots & H_{1m}(s) \\ H_{21}(s) & H_{22}(s) & \dots & H_{2m}(s) \\ \vdots & \vdots & \ddots & \vdots \\ H_{p1}(s) & H_{p2}(s) & \dots & H_{pm}(s) \end{bmatrix} \tag{A.2}$$

where $H_{ij}(s)$ is the transfer function between input j and output i. Note that in the following formula a, b, n also depend on i, j, but for notational reasons the subscripts i, j are omitted.

$$H_{ij}(s) = \frac{b_1 s^{n-1} + b_2 s^{n-2} + \dots + b_n}{s^n + a_1 s^{n-1} + \dots + a_n} \tag{A.3}$$

A minimal realization of a given transfer function $H(s)$ is one that has the smallest size A-matrix for all $\{A, B, C\}$ that satisfy

$$H(s) = C(sI - A)^{-1}B. \tag{A.4}$$

In terms of controllability and observability the following important property holds:

A realization $\{A, B, C\}$ is minimal if and only if it is controllable and observable.

A.2 Controllability observability

The controllability matrix $\mathcal{C}_N(B, A)$ and the observability matrix $\mathcal{O}_N(C, A)$ are given by

$$\mathcal{C}_N(B, A) = \begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix}, \tag{A.5a}$$

$$\mathcal{O}_N(C, A) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix}. \tag{A.5b}$$

A linear system (A.1) of order n is said to be controllable if and only if $\mathcal{C}_n(A, B)$ has rank n. When a system is controllable this means that, given an initial state of the system, it is always possible to design an input signal that leads the system, in finite time, to the desired final state. A linear system (A.1) of order n is said to be observable if and only if $\mathcal{O}_n(C, A)$ has rank n. When a system is observable it is always possible to reconstruct, in finite time, the initial state of the system by observing the output.

A.3 Matrix Fraction Description

Write $H(s)$ as follows

$$H(s) = \frac{N(s)}{d(s)} \tag{A.6}$$

where, $d(s)$ equals the monic[1] least common multiple of the denominators of the entries of $H(s)$,

$$d(s) = s^r + d_1 s^{r-1} + \cdots + d_r \tag{A.7}$$

and $N(s)$ equals a polynomial matrix that contains the remainder after extracting $d(s)$.

A Matrix Fraction Description (MFD) of $H(s)$ is be given by:

$$H(s) = N(s)D^{-1}(s), \quad D(s) = d(s)I_m \tag{A.8}$$

where I_m equals an $m \times m$ identity matrix and $D(s)$ is the right denominator matrix. The degree of the MFD is defined to be degree of the denominator matrix:

$$\text{degree of the MFD} = \text{deg } D(s) = \text{deg det } D(s). \tag{A.9}$$

A.4 Smith Form

The Smith form $\Lambda(s)$ of any $p \times m$ polynomial matrix $P(s)$ can be obtained by elementary row and column operations. These operations result in corresponding unimodular[2] matrices $\{U_1(s), U_2(s)\}$. The Smith form of $P(s)$ is given by $\Lambda(s)$ as follows

$$P(s) = U_1(s)\Lambda(s)U_2(s) \tag{A.10}$$

where

$$\Lambda(s) = \text{diag } \{\lambda_i(s), \ldots, \lambda_r(s), 0, \ldots, 0\} \tag{A.11}$$

and

$$r = \text{rank of } P(s).$$

Further $\lambda_i(s)$ are monic polynomials that obey the following division property

$$\lambda_i(s)|\lambda_{i+1}(s), \qquad i = 1, \ldots, r-1. \tag{A.12}$$

The above notation means that $\lambda_i(s)$ divides $\lambda_{i+1}(s)$. Or equivalently, the polynomial $\lambda_i(s)$ is a factor, without remainder, of the polynomial $\lambda_{i+1}(s)$. A detailed description on how to form $\Lambda(s)$ can be found in Paragraph 6.3.3 of [Kai80]. Note that there the Smith form is defined as $\Lambda(s) = U(s)P(s)V(s)$, which is equivalent to (A.10) using inverse unimodular matrices $U(s) = U_1^{-1}(s)$ and $V(s) = U_2^{-1}(s)$. This can freely be done since for polynomial unimodular matrices the property holds that the inverse of a polynomial unimodular matrix is again a polynomial unimodular matrix.

[1] A polynomial is said to be monic if the highest power of s has coefficient one

[2] A matrix $U(s)$ is said to be unimodular if and only if det $U(s)$ = a nonzero constant independent of s

A.5 Smith-McMillan Form and Minimal Order

Recall that a transfer function matrix (A.2) can be written as (A.6) and therefore

$$d(s)H(s) = N(s) = U_1(s)\Lambda(s)U_2(s), \tag{A.13}$$

where $\Lambda(s)$ is the Smith form of $N(s)$ and $\{U_1(s), U_2(s)\}$ are unimodular matrices. To derive the Smith-McMillan form write

$$U_1^{-1}(s)H(s)U_2^{-1}(s) = \frac{\Lambda(s)}{d(s)} = \operatorname{diag}\left\{\frac{\lambda_i(s)}{d(s)}\right\}, \tag{A.14}$$

and reduce common factors to obtain

$$\frac{\lambda_i(s)}{d(s)} = \frac{\epsilon_i(s)}{\psi_i(s)}, \tag{A.15}$$

where $\{\epsilon_i(s), \psi_i(s)\}$ have to be monic and coprime (i.e. having no common roots). Also $\{\epsilon_i(s), \psi_i(s)\}$ will obey the division property as explained in the previous section

$$\begin{matrix}\epsilon_i(s)|\epsilon_{i+1}(s)\\ \psi_{i+1}(s)|\psi_i(s)\end{matrix}, \qquad i = 1, \ldots, r-1 \tag{A.16}$$

with

$$\psi_1(s) = d(s) \tag{A.17}$$

and

$$r = \text{rank of } H(s). \tag{A.18}$$

Then, the Smith-McMillan form of $H(s)$ is given by $M(s)$ as follows

$$H(s) = U_1(s)M(s)U_2(s) \tag{A.19}$$

where

$$M(s) = \begin{bmatrix}\operatorname{diag}\{\epsilon_i(s)/\psi_i(s)\} & 0\\ 0 & 0\end{bmatrix}. \tag{A.20}$$

Then, the Smith-McMillan degree, often called the McMillan degree, which gives the minimal degree of any state-space realization, is defined to be

$$n_{\min} = \sum \deg \psi_i(s). \tag{A.21}$$

A.6 Hermite Form

An $(m \times m)$ nonsingular polynomial matrix $P(s)$ can be written in its Hermite form $P_H(s)$ by elementary row or column operations. If only row operations are used, which is equivalent to premultiplication by an unimodular matrix, $P_H(s)$ will be so called *column-reduced.* If, on the other hand, only column operations are used, which is equivalent to postmultiplication by an unimodular matrix, $P_H(s)$ will be so called *row-reduced.* It also holds that if $P_H(s)$ is column-reduced, then $P_H^T(s)$ is row-reduced. The above properties are illustrated by

$$P_H(s) = U_H(s)P(s), \quad \text{where } P_H(s) \text{ is column-reduced} \tag{A.22a}$$
$$P_H^T(s) = P^T(s)U_H^T(s), \quad \text{where } P_H^T(s) \text{ is row-reduced.} \tag{A.22b}$$

The structure of a row-reduced Hermite form $P_H(s)$ is the following: $P_H(s)$ is upper/lower triangular, with the diagonal elements monic and of higher degree than any other element in that row. If the diagonal element is a constant than all other elements in that row are zero. For the column-reduced Hermite form the same holds, only the word *row* should be replaced by *column*. The actual construction of the Hermite form is described in the proof of Theorem 6.3.2 on page 375 of [Kai80].

A.7 Procedure to Obtain Minimal State-Space Realizations

The previous sections have introduced the necessary means to form a procedure that obtains a minimal state-space realization of a given transfer function matrix. This procedure will be presented here, and follows along the lines of [Kai80] Paragraph 6.4.4. The algorithm consists of the following steps

1. Start with a transfer function matrix $H(s)$ as in(A.2).
2. Extract the least common multiples of $H(s)$ and write according to (A.6)
$$H(s) = \frac{N(s)}{d(s)}. \tag{A.23}$$
3. Construct the Smith form $\Lambda(s)$ of the numerator matrix $N(s)$ and obtain unimodular matrices U_1 and U_2 such that
$$N(s) = U_1(s)\Lambda(s)U_2(s). \tag{A.24}$$
4. Construct the matrix $D_0(s)$ that, together with $N_0(s)$, forms the MFD $H(s) = N_0(s)D_0^{-1}(s)$ according to
$$N_0(s) = U_1(s)E(s), \qquad D_0(s) = U_2^{-1}(s)\Psi_R(s) \tag{A.25}$$
where
$$E(s) = \left[\begin{array}{c|c} \text{diag}\{\epsilon_i(s)\} & 0 \\ \hline 0 & 0 \end{array}\right], \qquad p \times m \tag{A.26}$$
$$\Psi_R(s) = \left[\begin{array}{c|c} \text{diag}\{\psi_i(s)\} & 0 \\ \hline 0 & I_{m-r} \end{array}\right], \qquad m \times m \tag{A.27}$$
$$r = \text{rank of } H(s),$$
and I denotes the identity matrix. Also $\{\epsilon_i(s), \psi_i(s)\}$ are the numerator and denominator polynomials of the Smith-McMillan form of $H(s)$. These polynomials were defined in (A.15).

5. Write $D_0(s)$ in its row-reduced Hermite form $D_H(s)$ as follows

$$D_H(s) = D_0(s)U_H(s), \tag{A.28}$$

and define accompanying the numerator matrix

$$N_H(s) = N_0(s)U_H(s), \quad \text{or} \quad N_H(s) = H(s)D_H(s) \tag{A.29}$$

Note that, since $U_H(s)$ is unimodular the degree of $D_H(s)$ remains equal to the degree of $D_0(s)$.

6. Split $D_H(s)$ in two parts

$$D_H(s) = S(s)D_{hr} + P(s)D_{lr} \tag{A.30}$$

where

$$S(s) = \text{diag}\{s^{l_i}, i = 1, \dots, m\} \tag{A.31}$$

$$P(s) = \text{blkdiag}\{[s^{l_i-1}, \dots, s, 1], \quad i = 1, \dots, m\} \tag{A.32}$$

and

$$l_i = \text{degree of the } i\text{th row of } D(s) \tag{A.33}$$

D_{hr} has size $(m \times m)$ and is called the highest-row-degree coefficient matrix of $D_H(s)$. D_{lr} has size $(\sum l_i \times m)$ and is called the lower-row-degree coefficient matrix of $D_H(s)$.

7. Let A be given by

$$A = \tilde{\mathbb{I}} A_o \tilde{\mathbb{I}} \tag{A.34}$$

where

$$\tilde{\mathbb{I}} = \text{blkdiag}\{\tilde{I}_{l_i}, \ i = 1, \dots, m\} \tag{A.35}$$

$$\tilde{I}_{l_i} = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & \iddots & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}, \quad l_i \times l_i \tag{A.36}$$

and

$$A_o = A_o^0 - D_{lr}D_{hr}^{-1}C_o^0 \tag{A.37}$$

where

$$A_o^0 = \text{blkdiag}\left\{ \begin{bmatrix} 0 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & 0 \end{bmatrix}, \quad l_i \times l_i, \quad i = 1, \dots, m \right\} \tag{A.38}$$

$$C_o^0 = \text{blkdiag}\left\{ \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}, \quad 1 \times l_i, \quad i = 1, \dots, m \right\}. \tag{A.39}$$

8. Construct B as follows

$$B^T = C_o^0. \tag{A.40}$$

9. In order to construct C, define

$$\bar{N}(s) = N_H(s)D_{hr}^{-1}C_o^0\tilde{\mathbb{I}} \tag{A.41}$$

and write $\bar{N}(s)$ in terms of powers of s.

$$\bar{N}(s) = \bar{N}_0 s^n + \bar{N}_1 s^{n-1} + \cdots + \bar{N}_n \tag{A.42}$$

Then C is obtained by

$$C = \bar{N}(A) \tag{A.43}$$

where the notation $\bar{N}(A)$ stands for the evaluation of $\bar{N}(s)$ at $s = A$, which is given by

$$\bar{N}(A) = \bar{N}_0 A^n + \bar{N}_1 A^{n-1} + \cdots + \bar{N}_n. \tag{A.44}$$

A detailed argumentation for the choice of $\bar{N}(s)$ in (A.41) can be found on page 420 of [Kai80]. The argumentation uses the polynomial division theorem for the special case that $D_H(s) = sI - A$.

After following the above steps, a minimal state-space realization $\{A, B, C\}$ with $D = 0$ has been constructed. This realization is controllable as well as observable and satisfies $H(s) = C(sI - A)B$.

A.8 Example of a Minimal Realization

To illustrate the procedure described in the previous section an example will be given. The example derives a minimal state-space realization of the dynamic model with Padé approximations as described in Section 4.1. In order to prevent the matrices in the following example from becoming to large, all processing times of the machines are chosen equal to one: $t_{ei} = 1 \; \forall \; i$. This leads to the following transfer function matrix

$$H(s) = \begin{bmatrix} -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s} & 0 \\ 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{s} \\ 0 & 0 & -\frac{1}{s} & 0 & \frac{1}{s}e^{-s} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{s} & 0 & \frac{1}{s}e^{-s} & 0 & 0 & 0 & 0 \\ \frac{1}{s}e^{-s} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{s}e^{-s} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{s}e^{-s} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{s}e^{-s} & 0 & 0 & 0 & -\frac{1}{s} & 0 & 0 \end{bmatrix} \tag{A.45}$$

where

$$e^{-s} = \frac{s^2 - 6s + 12}{s^2 + 6s + 12}. \tag{A.46}$$

The least common multiple $d(s)$ given by (A.6) is found to be

$$d(s) = s(s^2 + 6s + 12). \tag{A.47}$$

The Smith form of $N(s)$ is given by

$$\Lambda(s) = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&s^2+6\,s+12&0&0&0\\ 0&0&0&0&0&0&0&s^2+6\,s+12&0&0 \end{bmatrix}. \tag{A.48}$$

The denominator matrix $D_0(s)$ is irrelevant and to large to display here. The relevant matrix $\Psi_R(s)$ is given by

$$\Psi_R(s) = \begin{bmatrix} (s^2+6\,s+12)\,s,0,0,0,0,0,0,0,0,0 \\ 0,(s^2+6\,s+12)\,s,0,0,0,0,0,0,0,0 \\ 0,0,(s^2+6\,s+12)\,s,0,0,0,0,0,0,0 \\ 0,0,0,(s^2+6\,s+12)\,s,0,0,0,0,0,0 \\ 0,0,0,0,(s^2+6\,s+12)\,s,0,0,0,0,0 \\ 0,0,0,0,0,(s^2+6\,s+12)\,s,0,0,0,0 \\ 0,0,0,0,0,0,s,0,0,0 \\ 0,0,0,0,0,0,0,s,0,0 \\ 0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,1 \end{bmatrix} \tag{A.49}$$

Therefore, the minimal degree of the system (McMillan degree) is equal to

$$\begin{aligned} n_{\min} &= \sum \deg \psi_i(s) \\ &= 6 * 3 + 2 = 20. \end{aligned} \tag{A.50}$$

The Hermite form of $D_0(s)$ is given by

$$D_H(s) = \begin{bmatrix} s^2+6\,s+12&0&0&0&0&0&0&0&0&0\\ 0&s^2+6\,s+12&0&0&0&0&0&0&0&0\\ s^2+6\,s+12&0&s^3+6\,s^2+12\,s&0&0&0&0&0&0&0\\ 0&s^2+6\,s+12&0&s^3+6\,s^2+12\,s&0&0&0&0&0&0\\ s^2+6\,s+12&0&0&0&s^3+6\,s^2+12\,s&0&0&0&0&0\\ 0&s^2+6\,s+12&0&0&0&s^3+6\,s^2+12\,s&0&0&0&0\\ 12&0&0&0&0&0&s&0&0&0\\ 0&12&0&0&0&0&0&s&0&0\\ 12&0&0&0&0&0&0&0&s&0\\ 0&12&0&0&0&0&0&0&0&s \end{bmatrix}. \tag{A.51}$$

$D_H(s)$ is split in two parts: D_{hr} and D_{lr}. Since D_{hr} represents the coefficients of the highest order of s, and the diagonal elements are monic, it follows that D_{hr} is a unity matrix of size

$(m \times m)$. The other part D_{lr} is given by

$$D_{lr}^T = \begin{bmatrix} 6 & 12 & 0 & 0 & 1 & 6 & 12 & 0 & 0 & 0 & 1 & 6 & 12 & 0 & 0 & 0 & 12 & 0 & 12 & 0 \\ 0 & 0 & 6 & 12 & 0 & 0 & 0 & 1 & 6 & 12 & 0 & 0 & 0 & 1 & 6 & 12 & 0 & 12 & 0 & 12 \\ 0 & 0 & 0 & 0 & 6 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 12 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.52}$$

and the vector l_i is found to be

$$l_i = \begin{bmatrix} 2 & 2 & 3 & 3 & 3 & 3 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{A.53}$$

The matrix $\bar{N}(s)$ from (A.41) is given by

$$\bar{N}(s) = \begin{bmatrix} 0 & -s-6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -s-6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -12 & 0 & 0 & 0 & 0 & -s^2-6\,s-12 & 0 & 0 & 0 & 0 & 0 & s^2-6\,s+12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & -s^2-6\,s-12 & 0 & 0 & 0 & 0 & 0 & s^2-6\,s+12 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -s^2-6\,s-12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -s^2-6\,s-12 & 0 & 0 & 0 & 0 \\ 0 & s-6 & 0 & 0 & 0 & 0 & s^2-6\,s+12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & s-6 & 0 & 0 & 0 & 0 & 0 & s^2-6\,s+12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{A.54}$$

When all the steps are correctly executed this leads to the following realization $\{A, B, C\}$ where $D = 0$

$$A = \begin{bmatrix} 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 1 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 & 1 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A.55}$$

$$B = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1 \end{bmatrix} \tag{A.56}$$

and

$$C = \begin{bmatrix} -1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&-1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&-1&0&0&0&0&0&1&-12&72&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&-1&0&0&0&0&0&1&-12&72&0&0&0&0 \\ 1&-12&0&0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&-12&0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0&0 \\ 0&0&0&0&1&-12&72&0&0&0&0&0&0&0&0&0&-1&0&0&0 \\ 0&0&0&0&0&0&0&1&-12&72&0&0&0&0&0&0&0&-1&0&0 \end{bmatrix}. \tag{A.57}$$

Now a state-space model consisting of A (A.55), B (A.56) and C (A.57) exists which is suitable for controller design.

Appendix B

MPC implementation

A schematic view of the MPC implementation is presented in Figure 4.4. This appendix provides the equations for the state observer and the MPC controller.

B.1 State observer

Generally a state observer is used to reconstruct the state of the process to be controlled, often referred to as the plant. The state is reconstructed using process measurements $y_p(k)$, process inputs $u(k)$, and an internal model of the plant:

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y_p(k) &= C x(k) \ , \end{aligned} \tag{B.1}$$

The observer equations used in the MPC implementation are:

$$x(k|k) = x(k|k-1) + K_x[y_p(k) - y(k-1|k)] \tag{B.2}$$

$$x(k+1|k) = \Phi x(k|k) + \Gamma u(k) \tag{B.3}$$

$$y(k|k-1) = C x(k|k-1) \tag{B.4}$$

Here the notation $x(k|k-1)$ denotes the estimated value of x at time k based on information available at time $k-1$. Substituting (B.4) into (B.2) and eliminating $x(k|k)$ from (B.3) gives

$$x(k+1|k) = (\Phi - \Phi K_x C) x(k|k-1) + \Phi K_x y_p(k) + \Gamma u(k) \ , \tag{B.5}$$

which is a stable system if the eigenvalues of $(\Phi - \Phi K_x C)$ lie within the unit disc.

If the state estimate error is defined $e(k) = x(k) - x(k|k-1)$ it follows that

$$e(k+1) = (\Phi - \Phi K_x C) e(k) \ , \tag{B.6}$$

showing that the state estimate error converges to zero if (B.5) is stable. The convergence rate is determined by the location of the eigenvalues.

If the pair (Φ, Γ) is observable (Appendix A.2), then a gain matrix K_x exists that places the observer eigenvalues at stable locations in the complex plain. Finding such a gain matrix is dual to finding a state-feedback pole placement matrix and can be solved using the PLACE algorithm of MATLAB® . Let $K = \Phi K_x$, then

$$e(k+1) = (\Phi - KC)e(k) \tag{B.7}$$

and K can be found by placing the poles of $\Phi - KC$ at a desired location. Then, if Φ is invertable, inverse of Φ is used to obtain

$$K_x = \Phi^{-1}K \tag{B.8}$$

B.2 MPC controller

This section of the appendix explains more closely how the MPC controller is implemented in this thesis. Model Predictive Control uses an internal model of the process to be controlled to predict the future outputs of that process over the prediction horizon. Let the internal model be given by the following linear discrete-time state-space model

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= Cx(k) \ , \end{aligned} \tag{B.9}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^i$, $y \in \mathbb{R}^o$. Further, Φ, Γ, and C are of appropriate size and D is zero. Let

$$\begin{aligned} p &= \text{prediction horizon}, \\ m &= \text{control horizon}. \end{aligned}$$

The output prediction is based on a corresponding input trajectory, which is determined by minimizing a quadratic cost function with or without constraints (4.37). The following notation is introduced for the states, outputs, and inputs over the appropriate horizons. The notation and relations presented in this appendix are obtain from [Mac02] and lecture handouts from Harm van Essen [Ess03].

$$x^p(k+1) = \begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+p|k) \end{bmatrix}, \ y^p(k+1) = \begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+p|k) \end{bmatrix}, \ \Delta u^m(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix} \tag{B.10}$$

In the above notation, $x^p(k+1)$ is an $n \times p$ column containing the state vector of length n for all samples of the prediction horizon p, $y^p(k+1)$ is an $o \times p$ column containing the output vector of length o for all samples of the prediction horizon p, and $\Delta u^m(k)$ is an $i \times m$ column containing the input vector (control moves) for all samples of the control horizon m. It is assumed that the input determined at time k can also be applied at time k; computational delay is therefore neglected.

The evolution of the state vector over the prediction horizon $x^p(k+1)$ can be split in a part determined by a constant input: $\Delta u = 0$, called the *free response*; and a part determined by future input changes: Δu, called the *forced response*.

$$x^p(k+1) = x^p_{\Delta u=0}(k+1) + x^p_{\Delta u}(k+1) \tag{B.11}$$

The state vector $x^p(k+1)$ is determined by iterating the internal model $x(k+1|k) = \Phi x(k|k) + \Gamma u(k)$ over the prediction horizon, and using the expression for the control move

$$u(k) = u(k-1) + \Delta u(k|k) \; .$$

Then,

$$\begin{bmatrix} x(k+1|k) \\ x(k+2|k) \\ \vdots \\ x(k+m|k) \\ x(k+m+1|k) \\ \vdots \\ x(k+p|k) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^m \\ \Phi^{m+1} \\ \vdots \\ \Phi^p \end{bmatrix} x(k|k) + \begin{bmatrix} \Gamma \\ \Phi\Gamma+\Gamma \\ \vdots \\ \sum_{i=0}^{m-1} \Phi^i\Gamma \\ \sum_{i=0}^{m} \Phi^i\Gamma \\ \vdots \\ \sum_{i=0}^{p-1} \Phi^i\Gamma \end{bmatrix} u(k-1)}_{\text{free response}} + \underbrace{\begin{bmatrix} \Gamma & 0 & \cdots & 0 \\ \Phi\Gamma+\Gamma & \Gamma & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \sum_{i=0}^{m-1} \Phi^i\Gamma & \dots & \Phi\Gamma+\Gamma & \Gamma \\ \sum_{i=0}^{m} \Phi^i\Gamma & \dots & \ddots & \Phi\Gamma+\Gamma \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=0}^{p-1} \Phi^i\Gamma & \dots & \dots & \sum_{i=0}^{p-m} \Phi^i\Gamma \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix}}_{\text{forced response } \; A\Delta u^m(k)} \tag{B.12}$$

The forced response matrix is an important matrix, since this matrix defines the relation between the proposed future inputs and the states. Denote this matrix with A. Two steps remain: first the output relation is presented, second the proposed future inputs are determined by solving the quadratic cost function. Let, the output equation be given by

$$y^p(k+1) = \overline{C} x^p(k+1) \; , \tag{B.13}$$

where

$$\overline{C} = \begin{bmatrix} C & & 0 \\ & C & \\ & & \ddots & \\ 0 & & & C \end{bmatrix} \tag{B.14}$$

Just like the state equation, also the output equation can be split in a free and forced response part

$$y^p(k+1) = y^p_{\Delta u=0}(k+1) + y^p_{\Delta u}(k+1) \; . \tag{B.15}$$

By using (B.13) it follows that

$$\begin{aligned} y^p(k+1) &= \overline{C} \left[x^p_{\Delta u=0}(k+1) + x^p_{\Delta u}(k+1) \right] \\ &= y^p_{\Delta u=0}(k+1) + \overline{C} A \Delta u^m(k) \\ &= y^p_{\Delta u=0}(k+1) + Y \Delta u^m(k) \end{aligned} \; . \tag{B.16}$$

Here,

$$Y = \overline{C} A \; , \tag{B.17}$$

of size $(o * p \times i * m)$, is called the *prediction matrix*, since this matrix defines the relation between the proposed future inputs—determined by the quadratic cost function—and the predicted future outputs. In order to determined the proposed future inputs over the prediction horizon, define the following tracking error

$$e^p(k+1) = y^p(k+1) - r^p(k+1) \; , \tag{B.18}$$

which has a free response part

$$e^p_{\Delta u=0}(k+1) = y^p_{\Delta u=0}(k+1) - r^p(k+1) \; , \tag{B.19}$$

where $r^p(k+1)$ is the output reference trajectory over the prediction horizon. The quadratic cost function (4.37) is given by[1]

$$\min_{\Delta u^m} \; (y^p - r^p)^T \, Q \, (y^p - r^p) + (\Delta u^m)^T \, R \, (\Delta u^m) \; , \tag{B.20}$$

where Q is the setpoint weighting matrix of size $(p*o \times p*o)$, and R is the input weighting matrix of size $(m*i \times m*i)$. By using (B.16) and (B.19) is follows that

$$\min_{\Delta u^m} \; (e^p_{\Delta u=0} + Y\Delta u^m)^T \, Q \, (e^p_{\Delta u=0} + Y\Delta u^m) + (\Delta u^m)^T \, R \, (\Delta u^m) \; . \tag{B.21}$$

Now (B.21) can be written as a standard QP problem

$$\min_{\Delta u^m} \; \frac{1}{2} (\Delta u^m)^T \, [Y^T Q Y + R] \, (\Delta u^m) + [Y^T Q e^p_{\Delta u=0}] \, \Delta u^m \; , \tag{B.22}$$

which can be solved with of without constraints. These two approaches are discussed next.

- In the unconstrained case, the solution of the quadratic cost function (B.22) is a linear gain matrix K_{MPC} that can be computed off-line:

  $$\Delta u^m = -\left[Y^T Q Y + R\right]^{-1} Y^T Q \quad e^p_{\Delta u=0} \tag{B.23}$$

  and the control law becomes a matrix multiplication

  $$\Delta u^m = K_{\text{MPC}} \;\; e^p_{\Delta u=0} \; , \tag{B.24}$$

  where K_{MPC} is of size $(m*i \times p*o)$. The vector $e^p_{\Delta u=0}$ of size $(p \times o)$ contains the deviations between the predicted outputs and the reference values over the full prediction horizon when the inputs are kept constant $\Delta u = 0$. The gain matrix K_{MPC} only changes when the weighting matrices or the model changes. Therefore, unconstrained MPC is not computational involving

- In the constrained case the quadratic cost function (B.22) has to be solved every sample time using a Quadratic Program (QP) solver. Constrained MPC, therefore, is more computational involving than unconstrained MPC. (B.22) is extended with linear constraints of the form

  $$A_{\text{con}} \Delta u^m \leq b_{\text{con}} . \tag{B.25}$$

  Two types of constraints are considered: input constraints, and output constraints.

[1]the arguments k and $k+1$ are omitted

The *input constraints* are written in format (B.25) by considering the following evaluation of $u^m \leq u_{\max}$

$$\begin{aligned}
u(k+j) &\leq u_{\max} \\
u(k+j-1) + \Delta u(k+j) &\leq u_{\max} \\
u(k+j-2) + \Delta u(k+j-1) + \Delta u(k+j) &\leq u_{\max} \, . \\
&\vdots \\
u(k-1) + \Delta u(k) + \cdots + \Delta u(k+j-2) + \Delta u(k+j-1) + \Delta u(k+j) &\leq u_{\max}
\end{aligned} \tag{B.26}$$

Perform the above evaluation for $j = \{0, 1, \ldots, m\}$ and stop as soon as the term

$$u(k-1) + \Delta u(k)$$

appears in the expression. An equivalent evaluation can be made for $u^m \geq u_{\min}$. If the final expression of every evaluation are put in a vector, the following constraint formulation is obtained

$$\begin{aligned}
J\Delta u^m &\leq u^m_{\max} - u^m \\
-J\Delta u^m &\leq u^m - u^m_{\min}
\end{aligned} \, , \tag{B.27}$$

where J has size $(i * m \times i * m)$ and contains unit matrices:

$$J = \left[\begin{array}{ccc|ccc}
1 & & & & & \\
 & \ddots & & & & \\
 & & 1 & & & \\
\hline
1 & & & 1 & & \\
 & \ddots & & & \ddots & \\
 & & 1 & & & 1
\end{array}\right] \, . \tag{B.28}$$

The *outputs constraints* are written in format (B.25) by using (B.17). This leads to the constraint formulation

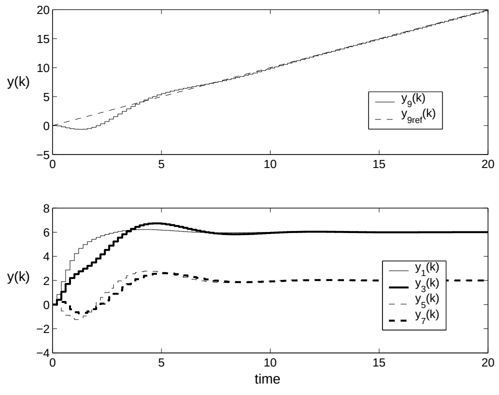
$$\begin{aligned}
Y\Delta u^m &\leq y^p_{max} - y^p_{\Delta u=0} \\
-Y\Delta u^m &\leq y^p_{\Delta u=0} - y^p_{\min}
\end{aligned} \, , \tag{B.29}$$

where Y is the prediction matrix (B.16). In order to solve the objective function (B.22) subject to (B.25), the MATLAB® function QUADPROG can be used. Type `help quadprog` on the MATLAB® command line to find out the specific QP formulation used by QUADPROG.

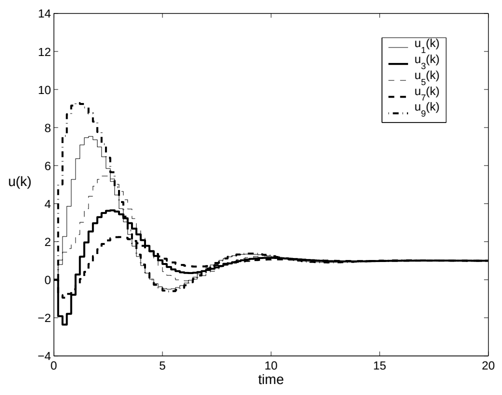
Appendix C

Simulation plots of fluid model with delay controlled by MPC

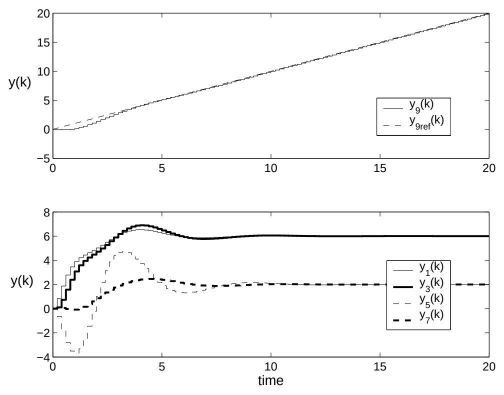
This appendix contains the simulation plots used to investigate the influence of the MPC tuning parameters, the influence of including delay in the approximation model, and the influence of input constraints. The figures presented in this appendix accompany the discussion in Section 4.3. Concerning the setpoint and input weighting matrices Q and R note the following: if all individual elements in Q and R are kept equal, Q and R are addressed as single numbers. In the figures presented in this section, the fraction Q/R is varied. Note that Q and R are actually square diagonal matrices of size $(p*o \times p*o)$ and $(m*i \times m*i)$, where p and m are the prediction and control horizon, and o and i are the number of outputs and inputs. Further, in the figures $y(k)$ always represents the *buffer levels*, and $u(k)$ always represents the accompanying *control signal.*



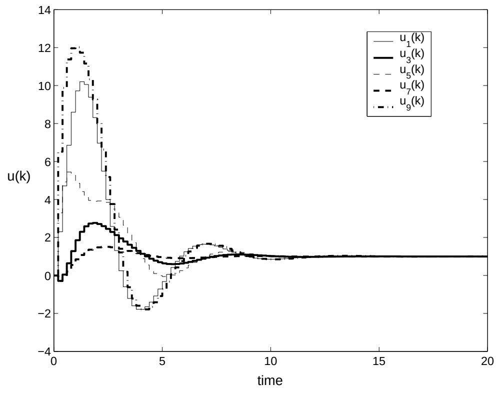
(a) internal model = approximation model



(b) internal model = approximation model

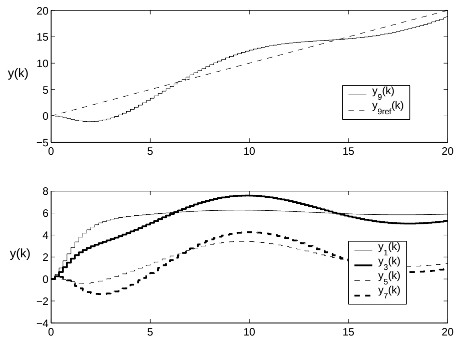


(c) internal model $\neq$ approximation model

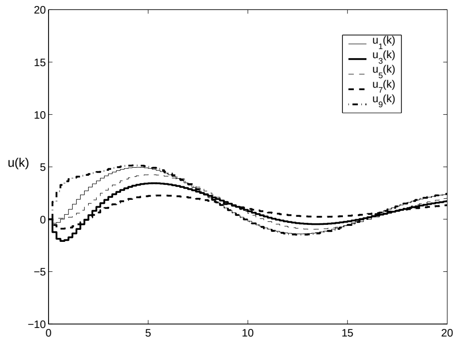


(d) internal model $\neq$ approximation model

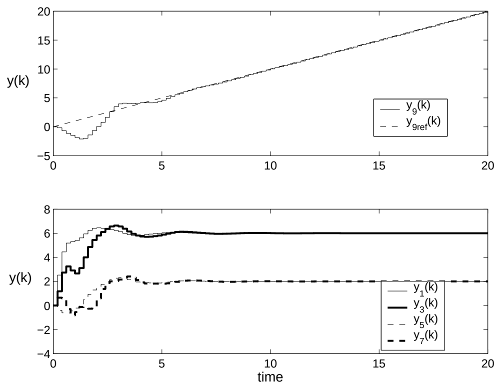
Figure C.1: Influence of including time delays in the internal model used by MPC on the buffer levels and control signal. Compare two situations: (i) approximation model with delay controlled by MPC with an internal model *with* delay: internal model = approximation model, (ii) approximation model with delay controlled by MPC with an internal model *without* delay. Further, $Q = R = 1$, $p = 10$, $m = 2$, $h = 0.2$, unconstrained MPC.



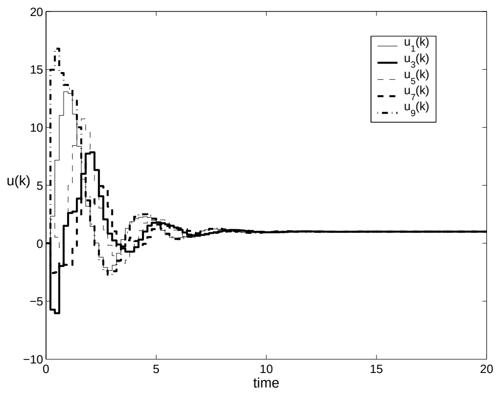
(a) $Q = 1,\ R = 10$



(b) $Q = 1,\ R = 10$

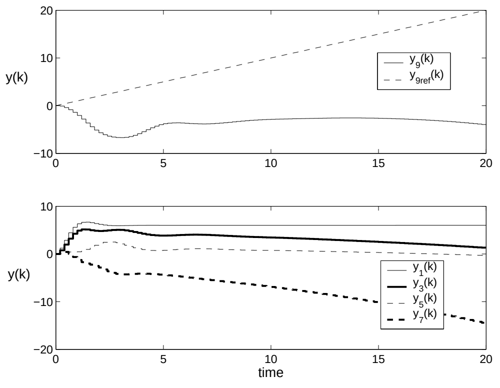


(c) $Q = 10,\ R = 1$

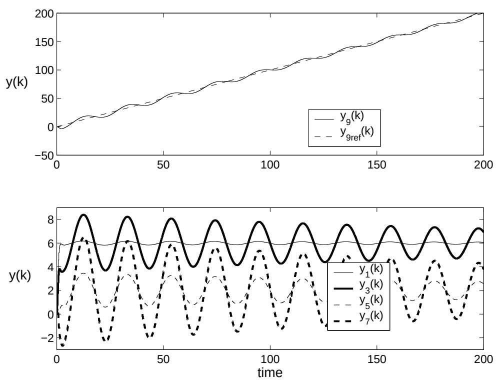


(d) $Q = 10,\ R = 1$

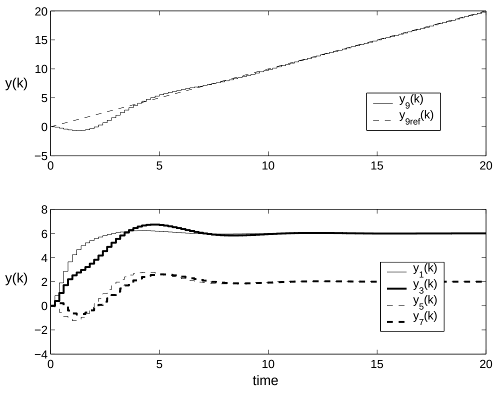
Figure C.2: Influence of setpoint weights Q and input weights R on the approximation model with delay responding to a setpoint change. Compare two situations: (i) input weights R ten times larger than setpoint weights Q, (ii) input weights R ten times smaller than setpoint weights Q. Further, internal model = approximation model, $p = 10$, $m = 2$, $h = 0.2$, unconstrained MPC.



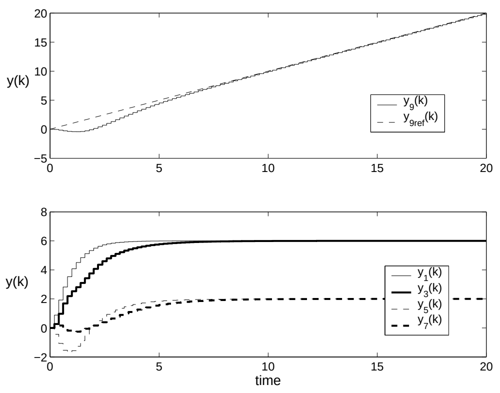
(a) $p = 3$, $m = 2$, $h = 0.2$



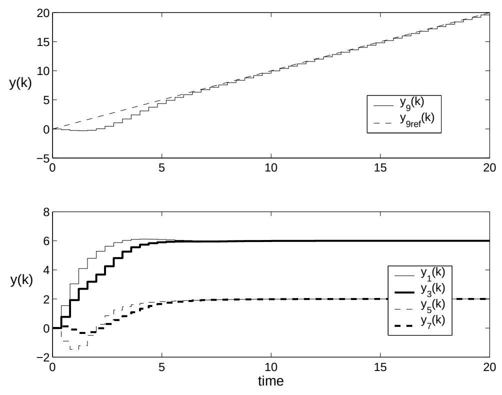
(b) $p = 5$, $m = 2$, $h = 0.2$



(c) $p = 10$, $m = 2$, $h = 0.2$

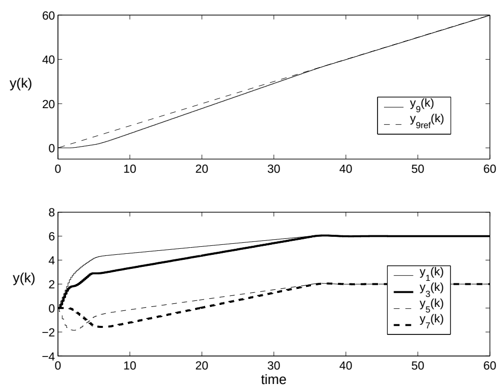


(d) $p = 20$, $m = 2$, $h = 0.2$

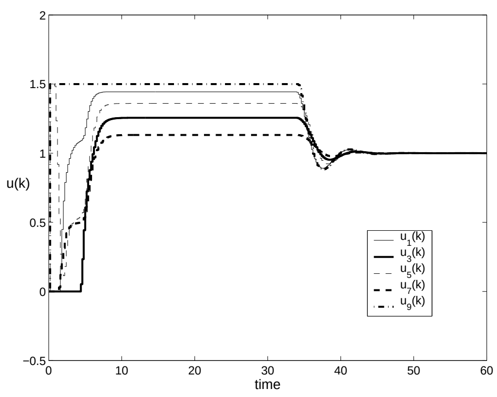


(e) $p = 10$, $m = 2$, $h = 0.4$

Figure C.3: Influence of the prediction horizon p on the approximation model with delay responding to a setpoint change. Buffer levels $y(k)$ are plotted for an increasing prediction horizon from $p = 3$ Figure C.3(a) until $p = 20$ Figure C.3(d). The system is stable if $ph > \max(t_e)$ or equivalently $p > 2/(3h)$. The response remains similar if the product ph is kept equal; Figure C.3(d) and Figure C.3(e) show two situations where ph is equal. Further, internal model = approximation model, $Q = R = 1$, unconstrained MPC.



(a) buffer levels



(b) constrained control signal: $0 \leq u(k) \leq 3/2$

Figure C.4: Influence of including input constraints in the MPC controller on the response of the approximation model with delay to a setpoint change. Negative inputs and inputs larger than the machine capacity are avoided: $0 \leq u(k) \leq 3/2$. Further, internal model = approximation model, $p = 10$, $m = 2$, $h = 0.2$.

Appendix D

Simulation scripts

This chapter of the appendix contains the code used to simulate the discrete event model and the communication between the MATLAB® controllers. The code for the MPC controller and the pole placement controller are not included in the appendix, if so desired these codes and others can be provided.

D.1 χ simulation diagram

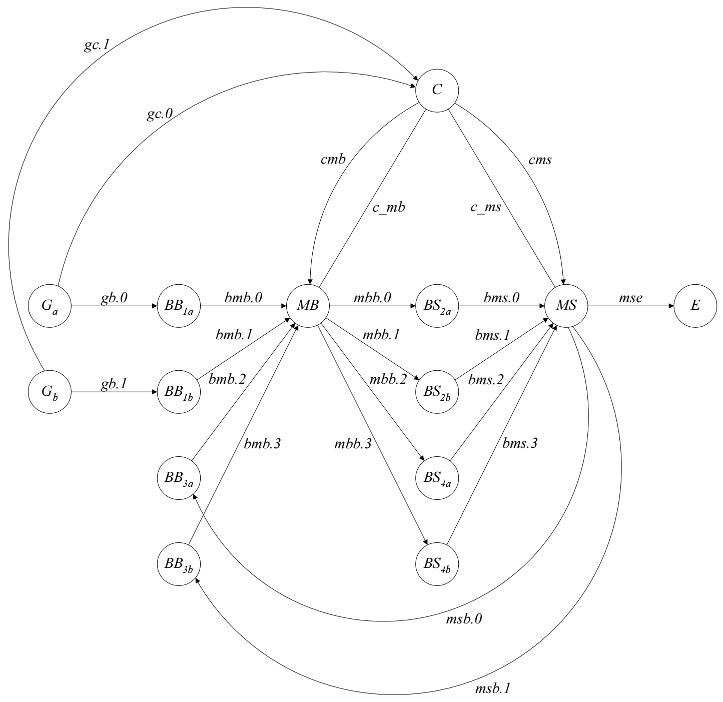


Figure D.1: χ simulation diagram accompanying the code in the following section

D.2 χ simulation code

```
// #### Four step re-entrant manufacturing line with batching
// #### Re-entrant manufacturing systems, composed of a batch machine
// #### and a single lot machine. Two products undergo four
// #### production steps.

type
  prodType = nat // 0, 1
, step     = nat // 1, 2, 3, 4
, lot      = prodType # real // sojourn time

const N: nat = 10 // number of inputs

func prodDesired(p,u: real^N, T: real) -> real^N =
|[ i: nat
 | i:= 0; *[ i < N -> p.i:= p.i + u.i*T; i:= i + 1 ]
 ; ret p
]|


func nextprod(p, pbar: real^N, batch: bool, x: nat^4, b: nat) -> bool # nat =
|[ i, j, next: nat, Dp: real^4
 | i:= 0; j:= 3; Dp:= <0.0, 0.0, 0.0, 0.0>
 ; *[ i < 4 -> [     batch and x.i >= b -> Dp.i:= p.i - pbar.i
               | not batch and x.i >= b -> Dp.i:= p.(i+4) - pbar.(i+4)
               |              x.i <  b -> skip
               ]
               ; i:= i + 1
    ]
 ; i:= 0
 ; *[ i /= j -> [ Dp.i >  Dp.j -> next:= i; j:= j - 1
                | Dp.i <= Dp.j -> next:= j; i:= i + 1
                ]
    | i  = j -> [ Dp.next <= 0.0 -> next:= 101; ret <false,next>
                | Dp.next >  0.0 -> ret <true,next>
                ]
    ]
]|


proc G (a: ?nat, b: !lot, d: ?int, Ts: real)=
|[ nr: nat, w: real, n: int
 | d?n; w:= Ts/n
 ; *[ n > 0 -> a?nr; b!<nr,time>; delta w; n:= n - 1
    | n = 0 -> d?n; w:= Ts/n
    ]
]|


proc Bbatch(a: ?lot, b: !lot*)=
|[ x: lot, xs: lot*, bs: nat
 | xs:= []; bs:= GetNat("batchsize")
 ; *[ true         ; a?x           -> xs:= xs ++ [x]
    | len(xs) >= bs; b!take(xs,bs) -> xs:= drop(xs,bs)
    ]
]|


proc Bsetup(a: ?lot*, b: !lot)=
|[ xs, ys: lot*
 | xs:= []
 ; *[ true       ; a?ys     -> xs:= xs ++ ys
    | len(xs) > 0; b!hd(xs) -> xs:= tl(xs)
    ]
]|


// xb: <1a,1b,3a,3b>, xs: <2a,2b,4a,4b>,
// next: 101 -> initialization value

proc C(a: (!nat)^2, b, c: !nat, d, e: ~void, f: (!int)^2, Ts: real, lambda: real^2)=
|[ nextb, nexts: bool # nat, bs, k, j: nat, xb, xs: nat^4, xf: nat^2, u, p, pbar, pcor: real^N, cp: int^2
```

```
 | xb:= <0,0,0,0>; xs:= <0,0,0,0>; xf:= <0,0>; nextb:= <false,101>; nexts:= <false,101>
 ; bs:= GetNat("batchsize"); k:= 1; j:= 0
 ; *[ j < N -> p.j:= 0.0; j:= j + 1 ]; pbar:= p; pcor:= p; j:= 0
 ; u:= mpcinit(xb,xs,xf,Ts,lambda); p:= prodDesired(p,u,Ts)
 ; *[ j < 2 -> cp.j:= ceil(p.(j+8)-pbar.(j+8)); j:= j + 1 ]; j:= 0
 ; *[ j < 2 and cp.j > 0 -> f.j!cp.j; j:= j + 1 ]; j:= 0
 ; [ resetlog() -> skip ]; [ writelog(p,pbar,u,pcor,xb,xs,xf,time) -> skip ]
 ; *[ i: nat <- 0..2: true; a.i!i     -> xb.i:= xb.i + 1; pbar.(i+8):= pbar.(i+8) + 1
                                        ; [ nextb.0 -> skip | not nextb.0 -> nextb:= nextprod(p,pbar,true,xb,bs) ]
    | nextb.0;      b!(nextb.1)       -> xb.(nextb.1):= xb.(nextb.1) - bs
    | nexts.0;      c!(nexts.1)       -> xs.(nexts.1):= xs.(nexts.1) - 1
    | true;         d~                -> xs.(nextb.1):= xs.(nextb.1) + bs
                                        ; pbar.(nextb.1):= pbar.(nextb.1) + bs
                                        ; [ nexts.0 -> skip | not nexts.0 -> nexts:= nextprod(p,pbar,false,xs,1) ]
                                        ; nextb:= nextprod(p,pbar,true,xb,bs)
    | true;         e~                -> [ nexts.1 <  2 -> xb.(nexts.1+2):= xb.(nexts.1+2) + 1
                                         | nexts.1 >= 2 -> xf.(nexts.1-2):= xf.(nexts.1-2) + 1
                                         ]
                                        ; pbar.(nexts.1+4):= pbar.(nexts.1+4) + 1
                                        ; [ nextb.0 -> skip | not nextb.0 -> nextb:= nextprod(p,pbar,true,xb,bs) ]
                                        ; nexts:= nextprod(p,pbar,false,xs,1)
    | true; delta k*Ts-time            -> u:= mpcnewu(xb,xs,xf,time)
                                        ; [ writelog(p,pbar,u,pcor,xb,xs,xf,time) -> skip ]
                                        ; p:= prodDesired(p,u,Ts); j:= 0
                                        ; [ nextb.0 -> skip | not nextb.0 -> nextb:= nextprod(p,pbar,true,xb,bs) ]
                                        ; [ nexts.0 -> skip | not nexts.0 -> nexts:= nextprod(p,pbar,false,xs,1) ]
                                        ; *[ j < 2 -> cp.j:= ceil(p.(j+8)-pbar.(j+8)); j:= j + 1 ]; j:= 0
                                        ; *[ j < 2 and cp.j > 0 -> f.j!cp.j; j:= j + 1 ]; j:= 0
                                        ; k:= k + 1
  ]
]|


proc Mbatch(a: (?lot*)^4, b: (!lot*)^4, c: ?nat, d: ~void)=
|[ zs: lot*, dis:(-> real)^4, nx, i: nat, mu_batch: real^4
 | mu_batch:= GetReal4("mu_batch"); i:= 0
 ; *[ i < 4 -> dis.i:= exponential(1/mu_batch.i); i:= i + 1 ]
 ; *[ c?nx; a.nx?zs; delta sample dis.nx; b.nx!zs; d~ ]
]|


proc Msetup(a: (?lot)^4, b: (!lot)^2, c: !lot, d: ?nat, e: ~void)=
|[ x: lot, dis:(-> real)^4, nx, i: nat, mu_setup: real^4
 | mu_setup:= GetReal4("mu_setup"); i:= 0
 ; *[ i < 4 -> dis.i:= exponential(1/mu_setup.i); i:= i + 1 ]
 ; *[ d?nx; a.nx?x; delta sample dis.nx; [ nx < 2 -> b.nx!x | nx >= 2 -> c!x ]; e~ ]
]|


proc E (a: ?lot)=
|[ x: lot, beginsim, endsim, n: nat, data0, data1: real*, t: real
 | beginsim:= GetNat("beginsim"); endsim:= GetNat("endsim"); data0:= []; data1:= []
 ; *[ n <  beginsim               -> a?x; n:= n + 1
    | n >= beginsim and n < endsim -> a?x; [ n = beginsim -> t:= time | n > beginsim -> skip ]
                                    ; n:= n + 1
                                    ; [ x.0 = 0 -> data0:= data0 ++ [time-x.1]
                                      | x.0 = 1 -> data1:= data1 ++ [time-x.1]
                                      ]
    | n >= endsim                  -> !mean(data0), tab(), mean(data1), tab()
                                    ; [ littledata(t) -> skip ]
                                    ; !getlittle("tha"), tab(), getlittle("thb"),  tab()
                                    ; !getlittle("wipa"), tab(), getlittle("wipb"), tab()
                                    ; !getlittle("ut_batch"), tab(), getlittle("ut_setup"),nl()
                                    ; [ savelog() -> terminate ]
    ]
]|


syst GBMBME (Ts: real, lambda: real^2) =
|[ cmb, cms: -nat
 , gc: (-nat)^2
 , cg: (-int)^2
 , c_mb, c_ms: -void
 , mse: -lot
 , gb, msb: (-lot)^2
 , bms: (-lot)^4
 , bmb, mbb: (-lot*)^4
 |  i:nat <- 0..2: G(gc.i, gb.i, cg.i, Ts)
 || i:nat <- 0..2: Bbatch(gb.i,  bmb.i)
 || i:nat <- 0..2: Bbatch(msb.i, bmb.(i+2))
 || i:nat <- 0..4: Bsetup(mbb.i, bms.i)
 || Mbatch(bmb, mbb, cmb, c_mb)
 || Msetup(bms, msb, mse, cms, c_ms)
 || C(gc, cmb, cms, c_mb, c_ms, cg, Ts, lambda)
 || E(mse)
]|


xper(Ts: real, lambda: real^2) = |[ GBMBME (Ts, lambda) ]|
```

D.3 I/O function declaration

The I/O file contains the type declarations of the function arguments and results used in the χ simulation. These functions are calculated outside χ, after which the result is given back to χ. This file must be compiled together with the $*$.chi file. The file "data.py D.4 contains the body of the functions.

```
// io.ext
language "python"
file "data"

ext mpcinit(xb, xs: nat^4, xf: nat^2, Ts: real, lambda: real^2) -> real^10
ext mpcnewu(xb, xs: nat^4, xf: nat^2, t: real) -> real^10
ext newu(xb: nat^4, xs: nat^4) -> real^8
ext mean(n: real*) -> real
ext resetlog() -> bool
ext writelog(p, pbar, u, pcor: real^10, xb, xs: nat^4, xf: nat ^2, t: real) -> bool
ext savelog() -> bool
ext GetNat(s: string) -> nat    = "Get"
ext GetReal(s: string) -> real   = "Get"
ext GetReal2(s: string) -> real^2 = "Get"
ext GetReal4(s: string) -> real^4 = "Get"
ext writelogu(u: real^10, t:real) -> bool
ext littledata(t: real) -> bool
ext getlittle(dat: string) -> real
```

D.4 Main Python file

The main Python file (data.py) is used to evaluate the functions declared in the I/O file (io.ext) and used in the χ simulation. Data imports two objects: first the *pymat* module, used as an interface between Python and Matlab, second the simulation parameter file (paraas.py Appendix D.5).

```
# data.py
# Version 2 with Matlab inerfacing
#
from Numeric import *
import pymat
import paraasshort

globalH=pymat.open('matlab -nosplash')
globalG=pymat.open('matlab -nosplash')
```

```
def getH():
        global globalH
        pymat.eval(globalH,'clear all')
        return globalH

def getG():
        global globalG
        pymat.eval(globalG,'clear all')
        return globalG

def mpcinit(xb,xs,xf,Ts,lm):
        H=getH()
        y=xb+xs+xf
        pymat.put(H,'dt',[Ts])
pymat.put(H,'lambda', lm)
        pymat.put(H,'ypk', y)
        pymat.put(H,'xinit', Get('xinit'))
        pymat.put(H,'uinit', Get('uinit'))
        pymat.put(H,'Rdinputs', Get('R'))
        pymat.put(H,'Qoutputs', Get('Q'))
        pymat.put(H,'cdu', Get('cdu'))
        pymat.put(H,'highu', Get('highu'))
        pymat.put(H,'lowu', Get('lowu'))
        pymat.put(H,'Qmeas', Get('Qmeas'))
        pymat.put(H,'Yconstraints', Get('Yconstraints'))
        pymat.put(H,'Ylow', Get('Ylow'))
        pymat.put(H,'Yhigh', Get('Yhigh'))
        pymat.put(H,'gain_d', [Get('Kd')])
        pymat.put(H,'compdelay', [Get('compdelay')])
        pymat.put(H,'prediction', Get('model'))
        pymat.put(H,'setpoint', Get('setpoint'))
        pymat.put(H,'ap', [Get('p')])
        pymat.put(H,'am', [Get('m')])
        pymat.put(H,'yref', Get('yref'))
        pymat.eval(H,'mpcinit3')
        u=pymat.get(H,'upk')
        return (u[0],u[1],u[2],u[3],u[4],u[5],u[6],u[7],u[8],u[9])

def mpcnewu(xb,xs,xf,t):
        y=xb+xs+xf
        pymat.put(globalH,'ypk',y)
        pymat.put(globalH,'kt',[t])
        pymat.eval(globalH,'upk = mpcrun3(ypk,kt)')
        u=pymat.get(globalH,'upk')
        return (u[0],u[1],u[2],u[3],u[4],u[5],u[6],u[7],u[8],u[9])

def resetlog():
        G=getG()
        pymat.eval(G,'P=[]; Pbar=[]; U=[]; U2=[]; T2=[];')
        pymat.eval(G,'Pcor=[]; T=[]; Xb=[]; Xs=[]; Xf=[]; T2=[];')
        pymat.eval(G,'Xbs=[]; Xss=[]; Twip=[]; Xfs=[]')
        return 1

def writelog(p,pbar,u,pcor,xb,xs,xf,t):
       pymat.put(globalG,'p',p)
        pymat.put(globalG,'pbar',pbar)
        pymat.put(globalG,'u',u)
        pymat.put(globalG,'pcor',pcor)
        pymat.put(globalG,'xb',xb)
        pymat.put(globalG,'xs',xs)
        pymat.put(globalG,'xf',xf)
        pymat.put(globalG,'t',[t])
        pymat.eval(globalG,'P=[P p]; Pbar=[Pbar pbar]; U=[U u];')
        pymat.eval(globalG,'Pcor=[Pcor pcor]; Xb=[Xb xb]; Xs=[Xs xs]; Xf=[Xf xf]; T=[T t];')
        return 1

def littledata(tstart):
        pymat.put(globalH,'tstart',[tstart])
        pymat.put(globalH,'b',[Get('batchsize')])
        pymat.eval(globalH,'ts=max(find(ttot<=tstart))')
        pymat.eval(globalH,'tha=(yptot(end, 9)-yptot(ts, 9))/(ttot(end)-ttot(ts))')
        pymat.eval(globalH,'thb=(yptot(end,10)-yptot(ts,10))/(ttot(end)-ttot(ts))')
        pymat.eval(globalH,"[ut_batch,ut_setup,wipa,wipb]=utilization([tha,thb],yptot(:,1:4)',yptot(:,5:8)',b,ts)")
        return 1

def getlittle(dat):
        return pymat.get(globalH, dat)[0]

def savelog():
        pymat.eval(globalH,'save proddataH') ## yptot ycontot uptot ttot rptot cptot')
        pymat.eval(globalG,'save proddata') ## P Pbar U Pcor Xb Xs Xf T Xss Xbs Twip')
        pymat.close(globalG)
        pymat.close(globalH)
        return 1

def mean(a):
        pymat.put(globalH,'x',a)
        pymat.eval(globalH,'y=mean(x)')
        y=pymat.get(globalH,'y')
        return y[0]

dict=paraas.par
def Get(s):
        return dict[s]
```

D.5 Simulation parameter file

The simulation parameters used in the χ simulation are stored in the file paraas.py. They can be retrieved from this file by the function "Get". For instance: Get("batchsize") retrieves the batchsize.

```
#paraas.py

Inf    = 1e50
zeros  = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
ones   = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
Infset = (Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf)
mInfset= (-Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf)

par = { 'mu_batch': (1.5, 3.0, 3.0, 1.5)
      , 'mu_setup': (5.0, 7.5, 7.5, 5.0)
      , 'beginsim': 1000
      , 'endsim': 10000
      , 'batchsize' : 3
      , 'yref':  (6,6,6,6,2,2,2,2)

############# MPC Initialisation
      , 'xinit': zeros
      , 'uinit': zeros
      , 'R': (10, 10, 10, 10, 10, 10, 10, 10, 10, 10)  ## weighting factor input changes
      , 'Q': (1, 1, 1, 1, 1, 1, 1, 1, 10, 10)          ## weighting factor output - ref
      , 'cdu': Infset   ## maximal step size per sample of each input
      , 'highu': Infset ## maximal absolute value u of each input
      , 'lowu': mInfset ## minimal absolute value u of each input
      , 'Qmeas': ones
      , 'Yconstraints': zeros
      , 'Ylow': zeros
      , 'Yhigh': Infset
      , 'compdelay': 0
      , 'Kd': 0.0       ## d(k+1) = d(k) + Kd ( yp(k) - ypkm(k) ) filtergain output disturbance filter
      , 'model': 'prodline'
      , 'setpoint': 'prodlinesetpoint'
      , 'p': 10         ## prediction horizon in samples
      , 'm': 2          ## control horizon in samples
      }

## mu_batch(1): product a step 1
## mu_batch(2): product b step 1
## mu_batch(3): product a step 3
## mu_batch(4): product b step 3

## mu_setup(1): product a step 2
## mu_setup(2): product b step 2
## mu_setup(3): product a step 4
## mu_setup(4): product b step 4

## yref: <1a,1b,3a,3b,2a,2b,4a,4b>
## yref: < 1, 2, 3, 4, 5, 6, 7, 8>
```

D.6 tfmlcm

```
function [d,N,num] = tfmlcm(sys,m,n)
%TFMLCM converts transfer function matrix into data for TFM2SS
%
% [d,N,NUM] = TFMLCM(SYS,m,n) computes the data necessary for the MIMO
% state-space realization: TFM2SS. TFMLCM needs the symbolic toolbox.
%
% SYS must contain a transfer function matrix Hij(s) with m inputs and n outputs.
%
%     Hij(s) = 1/d(s)  Nij(s)
%
%                      |N11(s) N12(s) ... N1m(s)|
%                      |N21(s) N22(s) ........  |
%            = 1/d(s)  |  .           .      .  |
%                      |  .              .   .  |
%                      |Nn1(s) ......... Nnm(s)|
%
%     d(s)   = d0*s^r + d1*s^r-1 + ... + dr
%              Least Common Multiple of the denominator of Hij(s)
%     d      = [d0 d1 ... dr]
%
%     Nij(s) = N0*s^r + N1*s^r-1 + ... + Nr
%     N      = [N0 N1 ... Nr], matrix of size (n x m*r)
%
%     Nk     = |c11_k c12_k ... c1m_k|  for k = 0:r
%              |c21_k c22_k ...    . |  where cij_k are the coefficients of
%              | .           .     . |  the k-th power of s in Nij(s)
%              | .              .  . |
%              |cn1_k ........ cnm_k|
%
%     NUM    = [N11 ... Nn1 ... N1m ...Nnm]', with Nij = [cij_0 cij_1 ... cij_r]
%              matrix of size (n*m x r)

x=1;
for i=1:n
    for j=1:m
       [nm dn]=tfdata(sys(i,j),'v');
       p(i,j)=poly2sym(dn,'s');
       x=maple('lcm',x,p(i,j)); %use associativity lcm(lcm(a,b),c)=lcm(a,lcm(b,c))
    end
end

d=sym2poly(x);
r=length(d)-1; %degree of the polynomial d(s)


% contruct the remaider of sys
num=[];
for j=1:m
    for i=1:n
        [a,b]=tfdata(sys(i,j),'v');
        num=[num; deconv(conv(a,d),b)];
    end
end


% remove zero column off highest factors of s
[a,b]=size(num);
```

```
while sum(num(:,1)==0)==m*n & b > r
    num=[num(:,2:end)];
    [a,b]=size(num);
end

% rearrange columns off num into N
N=[];
for j=1:r
    for i=1:m
        N=[N num(n*(i-1)+1:n*i,j)];
    end
end
```

D.7 Simulink model

The Simulink model used to test the pole placement controller, together with the approximation model are shown below.

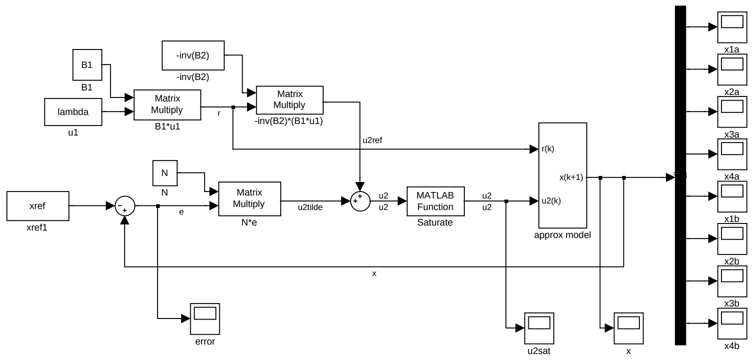


Figure D.2: Simulink model of controller and approximation model

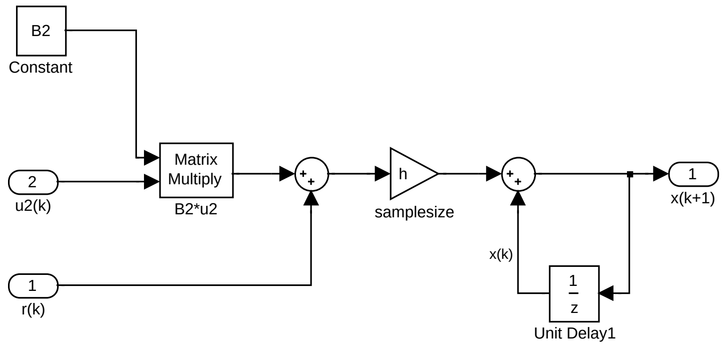


Figure D.3: Simulink model of approximation model