Feedback control compared to common control strategies for a multi-product flow line

H. Ploegmakers

SE 420328

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, January 2003

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

January 2003

Student	H. Ploegmakers
Supervisor	Prof.dr.ir. J.E. Rooda
Advisor	Dr.ir. A.A.J. Lefeber
Start	January 2002
Finish	December 2002
Title	Hybrid control of a flow line

Subject

Several control strategies are available for controlling flow-lines existing of a number of machines, producing few products in large numbers. Examples of these strategies are Pull, Push, Conwip and Polca. These control strategies achieve different performances, depending on the system's properties. An alternative for controlling such a (Discrete Event) production system might be the use of continuous Control Theory. Research so far primarily focused on throughput, without regarding cycle time. No systematic method has been developed for this method of control yet.

Assignment

Develop a method for using Control Theory to control the described flow-line. For that purpose, observe the flow-line's behavior as a continuous flow in discrete time. Aspects of interest include:

- the development of a continuous model for the Discrete Event production system, including system dynamics caused by delay during processing.
- the design of a suitable controller for the continuous model.
- the conversion of the continuous controller output signal to the discrete event production system input.
- the conversion of the discrete event production system output to the continuous controller input.

Describe under what conditions the described approach is valid. Compare the performance of the described controller to the performance of conventional control strategies. Present the results in a report, and provide recommendations for further research.

Prof.dr.ir. J.E. Rooda

Dr.ir. A.A.J. Lefeber

Systems Engineering

Department of Mechanical Engineering

Preface

This master's thesis is a result of a final assignment at the Eindhoven University of Technology. The final assignment, a research project of one year, is the closing part of a five years curriculum to become a Master of Science in Mechanical Engineering, with a specialization in Systems Engineering.

The Systems Engineering group aims to develop methods, techniques and tools for the design of advanced industrial systems. My research project has been performed within the research theme of Optimization and Control. The research within this theme is engaged in developing controller designs and optimization tools starting from computer models of manufacturing systems and machines. The objective of this research project was to develop a method for controlling a manufacturing system using feedback control.

First of all, I would like to thank my coach, Erjen Lefeber, for his enthusiastic support. I have much appreciated the vivid discussions, and his accurate contributions. I thank professor Rooda for his interest in the research project, and my colleague-students for their daily support. Furthermore, I would like to thank my family, friends and girlfriend for patiently supporting me during the busy recent period.

Hugo Ploegmakers
Eindhoven, January 2003

Summary

The objective of this research project is to develop a method to control a manufacturing system using feedback control. The manufacturing system of interest is a three machine multi-product flow line with stochastic process times, producing eight different product types in large quantities. The performance of the feedback controlled flow line is compared to the performance of the flow line controlled by several common control methods: push, pull, conwip and POLCA control. The main performance criteria for the flow line are assumed to be the throughput of the system and inventory level. Controlling a discrete event model of a manufacturing system may be described as defining on which conditions a next controlled event should take place.

Feedback control is an exponent of control theory. The objective of feedback control is to cause a system's output variables to follow a reference signal. For discrete time feedback control, at each sample time, the system outputs are measured (sampled). The controller uses the measured output values to determine appropriate controller output values. These controller output values are applied to the system as system inputs. In general, the design of a feedback controller requires a continuous dynamic model of the system to be controlled. A continuous model is a set of difference equations, describing the relation between input and output variables.

For this research project, the system to be controlled is a discrete event model of the three machine flow line. The inputs of the discrete event model are defined as a sequence of controlled events. The outputs of the discrete event model of the manufacturing system are defined as the buffer levels and the finished goods levels. The discrete event model of the flow line, having a sequence of discrete events as input, cannot directly be connected to a (continuous) feedback controller. Furthermore, the discrete event system cannot be directly described by a continuous dynamic model. Therefore, a signal conversion algorithm is defined to convert a continuous system input into appropriate discrete events for the discrete event model. The use of a conversion algorithm enables the following approach to apply feedback control to the discrete event model of the manufacturing system:

- Define a conversion algorithm to convert a continuous input signal into discrete events.
- The system to be controlled exists of the conversion algorithm and the discrete event model. The system's input is the input of the conversion algorithm. The system's outputs are the outputs of the discrete event model.
- Construct a continuous dynamic model of the system.
- Use the continuous model to design a feedback controller.

- Interconnect the discrete event model, the feedback controller and the conversion algorithm.

The conversion algorithm is defined such that the relation between the inputs of the conversion algorithm and the outputs of the discrete event model is linear. The input variables of the conversion algorithm are not required to have a physical meaning. The input of the conversion algorithm is defined as a target throughput per product for all machines. The objective of the conversion algorithm is to generate an appropriate sequence of events for the discrete event model, so that the throughput of the machines of the discrete event model follow the target throughput determined by the controller. An algorithm is described which records the shortage of production of each machine and generates appropriate events.

Designing the feedback controller requires a continuous dynamic model of the controlled system. System Identification is a method to derive a continuous model of a system, treating the system as a black box. After a model class has been selected, the model's parameters are fitted to experimental input and output data. The suitable model class for designing an MPC controller is a discrete time linear time-invariant state space model. A discrete event model is used to determine the response of the system output to a set of training input data. For the estimation of the model, the MATLAB® System Identification toolbox is used. A state space model for all inputs and outputs contains over 4000 parameters. Because of the problem's size, first a smaller problem is observed. The structure of the smaller problem's solution is used as a structure for the larger model. The state variables are defined equal to the outputs. The System Identification toolbox is used to estimate the unknown model parameters.

The continuous model is used to design a Model Predictive Control (MPC) controller. The criterion to define the optimal controller outputs is a weighed sum of system output errors and controller output moves, for a number of samples ahead. For unconstrained MPC, the control law may be written as a linear function of predicted future errors and a constant MPC gain matrix. The predicted future errors are computed using the continuous model of the system.

Simulation experiments are used to determine the performance of both the feedback controlled system and the manufacturing system controlled by common control strategies. The described common control strategies are push, pull, conwip and POLCA. The performance of the MPC controlled system strongly depend on the system's parameter settings. For the observed case, the MPC controlled system performs well. The presence of more variability or finite buffer sizes the does not disturb the MPC controlled system.

Samenvatting (in Dutch)

Doel van dit onderzoeksproject is een methode te ontwikkelen om regeltechniek te gebruiken voor het besturen van een fabricagesysteem. Het beschouwde fabricagesysteem is een *flow line*, waar een aantal verschillende producttypes in grote aantallen wordt geproduceerd. De prestaties van de door regeltechniek bestuurde lijn kunnen vervolgens worden vergeleken met die van dezelfde lijn bestuurd door conventionele besturingsmethoden, zoals *push*, *pull*, *conwip* en *POLCA*. De doorzet en het voorraadniveau worden beschouwd als de belangrijkste criteria voor de prestatie van het systeem. Het besturen van een *discrete event*-model van een fabricagesysteem kan worden opgevat als het definiëren onder welke voorwaarden een bestuurd *event* plaats mag vinden.

Regeltechniek wordt gebruikt om een variabele van een systeem een bepaald referentiesignaal te laten volgen. Bij regeltechniek in discrete tijd wordt na regelmatige intervallen elke keer de uitgang van het te regelen systeem gemeten. Deze kennis wordt gebruikt om de uitgang van de regelaar te bepalen. De uitgang van de regelaar wordt vervolgens als ingang van het systeem gebruikt. In het algemeen is voor het ontwerp van een regelaar een continu dynamisch model van het te regelen systeem nodig. Een continu model is een set differentievergelijkingen die de relatie tussen de uitgangen en de ingangen van het systeem beschrijven.

Dit onderzoeksproject beschrijft het regelen van een *discrete event*-model (DEM) van een fabricagesysteem. Wanneer dat model wordt beschouwd als een systeem met ingangen en uitgangen, is de ingang een stroom discrete *events*. De uitgangen kunnen worden gedefiniëerd als de voorraadniveaus in de buffers en de hoeveelheid voltooide producten. Omdat de stroom van discrete *events* geen continu signaal is, kan het DEM niet direct worden aangesloten op de (continue) regelaar. Verder kan het DEM niet één-op-één worden beschreven door een continu model. Om deze problemen te ondervangen, wordt een conversiealgoritme gedefiniëerd. Dit algoritme vertaalt de continue systeemingang naar discrete *events* voor het DEM. Hierdoor wordt de volgende aanpak mogelijk:

- Definiëer het conversiealgoritme om de continue systeemingang om te zetten in discrete *events*.
- Het te besturen systeem bestaat uit het conversiealgoritme en het DEM. De ingang van het conversiealgoritme is de ingang van het systeem, en de uitgang van het DEM is de uitgang.
- Stel een continu model op van dit systeem.
- Gebruik met continue model om een regelaar te ontwerpen.

- Verbind het DEM met de regelaar, de regelaar met het conversiealgoritme en het conversiealgoritme met het DEM.

Het conversiealgoritme moet bij voorkeur zo worden ontworpen, dat een lineaire relatie tussen de ingang van de conversie en de uitgang van het DEM ontstaat. Hoewel de ingangsvariabelen geen fysieke betekenis hoeven te hebben, is gekozen de ingang te definiëren als een doel-doorzet, per producttype per machine. Het doel van het conversiealgoritme wordt dan om de door de regelaar opgegeven doorzet zo goed mogelijk te volgen. Voor dat doel is een algoritme beschreven, dat door de productieachterstanden bij te houden de meest geschikte *events* kan genereren.

Om de regelaar te ontwerpen is een continu dynamisch model van het te regelen systeem nodig. *System Identification* is een methode om een dynamisch model van een systeem op te stellen, zonder de oorzaken achter het gedrag van het systeem te kennen. De methode schat de parameters van een bepaald model aan de hand van experimentele ingangs- en uitgangsdata. Voor het ontwerpen van een *Model Predictive Control* (MPC) regelaar is een lineair, tijds-invariant toestandsruimtemodel de aangewezen klasse. Om trainingsdata te genereren, zijn simulaties met een DEM gebruikt. Vervolgens wordt de System Identification Toolbox van MATLAB® gebruikt om het model te schatten. Het model voor alle ingangen en uitgangen bevat meer dan 4000 parameters. Vanwege dit formaat is eerst een kleiner probleem beschouwd. De structuur van de oplossing van dit kleinere probleem is gebruikt voor het volledige model. De toestandsvariabelen worden gelijk aan de uitgangsvariabelen gekozen, en de onbekende parameters van het model worden door de System Identification Toolbox geschat.

Het verkregen continue model wordt gebruikt om een *Model Predictive Control* (MPC) regelaar te ontwerpen. Een veelgebruikte doelfunctie om het optimale regelsignaal te bepalen bestraft kwadratisch afwijkingen van de systeemuitgang van het referentiesignaal, en veranderingen in de regelaaruitgang, beide voor een aantal regelacties vooruit. Wanneer geen beperkingen worden opgelegd aan de variabelen van de regelaar, kan de regelwet worden herschreven als een lineaire functie van de voorspelde toekomstige fouten en een constante MPC-versterkingsmatrix. Het continue model wordt gebruikt om de toekomstige fouten te voorspellen.

Simulatie-experimenten worden uitgevoerd met een model van het fabricagesysteem met de MPC-besturing. Dezelfde experimenten worden ook uitgevoerd voor de conventionele besturingen *push*, *pull*, *conwip* en *POLCA*. De prestaties van het systeem met MPC-besturing zijn sterk afhankelijk van de instellingen van de besturingsparameters. De beschreven flow-lijn levert met MPC-besturing goede prestaties. Het systeem is relatief ongevoelig voor variabiliteit, en introductie van eindige buffercapaciteiten stoort de MPC-besturing niet.

Contents

Table of symbols and abbreviations

α	Conversion algorithm filtering factor
δ	Throughput
δ^*	Target throughput
ϵ	Conversion algorithm backlog error
$\epsilon_{\text{threshold}}$	Conversion algorithm threshold backlog error
λ_i	Mean demand for products of product type group i
$\mu(x)$	Mean of x
$\bar{\varphi}_{Qi}$	Mean queueing time for workstation i
$\sigma^2(x)$	Variance of x
τ_i	Mean service time for product type i
τ_i	Mean service time for products of product type group$i$
φ	Flow time
ξ	Augmented state vector
$\hat{\xi}$	Reconstructed augmented state vector
c^2	Squared coefficient of variation
c_a^2	Squared coefficient of variation of inter arrival time
c_d^2	Squared coefficient of variation of inter departure time
c_e^2	Squared coefficient of variation of effective process time
e	output tracking error
e^p	System output error for the prediction horizon
k	Number of kanbans
k_d	Demand mix parameter
k_p	Service time mix parameter
m	Control horizon
n_u	Number of input variables
n_x	Number of state variables
n_y	Number of output variables
p	Prediction horizon
r	Demand fraction
r_a	Arrival rate
r_e	Effective processing rate
t_0	Natural process time
t_e	Effective process time
t_s	Sample time
u	Utilization

u_c	Controller input vector
u_s	System input vector
$\hat{u}_s^m$	System input trajectory for control horizon
$u_{s,\max}$	Maximum system input
$\bar{w}$	Mean WIP level
w_u	Input step weighing factor
w_y	Output error weighing factor
$w_{\max}$	Maximum WIP level
x	State vector
y_c	Controller output vector
y_r	Output reference vector
y_r^p	System output reference for the prediction horizon
y_s	System output vector
$\hat{y}_s^p$	Predicted system output for the prediction horizon
D_i	Mean demand for product type i
$E(x)$	Expectation of x
K_{MPC}	MPC controller gain matrix
N	Number of product types
Q	Input step weighing matrix
R	Output error weighing matrix

Chapter 1

Introduction

The performance of a manufacturing system strongly depends on the methods used to control the flow of goods and information in the manufacturing system. In the second half of the twentieth century, various new concepts for controlling the flow of inventory have been developed. Examples of these concepts are Material Requirements Planning (MRP), Just In Time (JIT) and Zero Inventories. The use of the concept MRP explosively grew in the 1970's ([Hop00]). In the 1980's, methods such as Just In Time and Zero Inventories, originating from Japan, achieved successes ([Mon83]). Currently, numerous variants of control methods have been developed, trying to combine the best aspects of both MRP and JIT.

Various tools may be used for the design and analysis of control methods for a manufacturing system. The science of queueing theory provides methods to perform computations on the queueing behavior of relatively simple systems. The rise of computer technology enabled the use of simulation experiments to test more complicated designs of control methods and manufacturing systems. For example, discrete event models may be used to predict the performance a manufacturing system and control method. Nevertheless, designing a control method mainly remains a case of experience, rules of thumb and a process of trial-and-error.

An alternative method to design a control method for a manufacturing system may be the use of control theory and feedback control. The objective of control theory is to cause a system's variable to follow a reference signal. Of a feedback controlled system, the output variables are measured, and the knowledge of the output variables is used to determine an appropriate input signal for the system. Control theory is a well-developed field of science, intensively studied since World War II. It is widely used for controlling various types of systems, such as mechanical systems or chemical process systems. Less research has been done in the use of feedback control for controlling manufacturing systems.

The objective of this research project is to develop a method for controlling the flow of goods in a manufacturing system, using feedback control. The performance of the feedback controlled manufacturing system should be compared to the performance of common control methods. As a test case to measure the performance of the control methods, an imaginary three machine multi-product flow line is defined. For this research project, the main performance criterion for a controlled manufacturing system is the relation between inventory level and throughput. A given throughput should be reached with a minimum of inventory, or, a maximum throughput should be achieved for a given amount of inventory. The high throughput level and low inventory level may be used as a reference signal for

a feedback controller. Objective of the feedback controller is to make the manufacturing system follow this reference trajectory.

The method to design a feedback controller for a manufacturing system may be divided into the following aspects:

- The feedback controller: The type of feedback controller chosen to control the manufacturing system is a Model Predictive Control (MPC) controller. As for most types of feedback controllers, designing an MPC controller requires a continuous dynamic model of the system that is to be controlled.
- The continuous dynamic model: A continuous dynamic model of a system is a set of differential (or difference) equations describing the response of the system outputs to the system inputs. In this research project, the method of System Identification is used to construct a dynamic model of the manufacturing system.
- The conversion of signals: The input and output signals of a feedback controller are generally continuous variables. A discrete event model of a manufacturing system describes the behavior of the system using discrete events. For this research project, a feedback controller is to be connected to a discrete event model. The incompatibility of the continuous and discrete signals require a conversion of the continuous controller output signal into discrete events.

Simulation experiments with discrete event models are used to measure the performance of both the feedback controlled manufacturing system and the same manufacturing system controlled by common control methods. The results of the simulation experiments are compared.

After this introduction, the report continuous with the introduction of some important parameters of a manufacturing system, such as WIP and throughput. Analytical relations for these parameters are discussed. These relations serve to predict and understand some aspects of the behavior of a manufacturing system, such as queueing. In Chapter 3, a flow line is introduced as a test case to which feedback control and common control methods are applied. Chapter 4 describes four common control methods: push, pull, conwip and POLCA. Simulation experiments are performed to measure the performance of these control methods applied to the flow line. The results of the experiments are presented and discussed. Chapter 5 gives an introduction to feedback control and defines the input and output variables of the controlled system. Chapter 6 describes a conversion algorithm to convert the continuous controller output into discrete events for the discrete event model. In Chapter 7, the method of System Identification is introduced. This method is used to construct a dynamic model of the conversion algorithm and discrete event model. In Chapter 8, the concept of MPC is described. The dynamic model derived in Chapter 7 is used for the design of an MPC controller. Chapter 9 describes the closed loop system, consisting of a discrete event model of the manufacturing system, the MPC controller and the signal conversion algorithm. Discrete event simulation experiments are performed to measure the performance of the MPC controlled discrete event system. The results of these experiments are presented and discussed. The report ends with conclusions and recommendations for further research.

Chapter 2

Controlling a manufacturing system

Goal of this research project is to compare the performances of a manufacturing system controlled by common control strategies and a new control strategy using feedback control. This chapter defines a manufacturing system and some of its important properties. These properties may be used to describe the performance of the system. Discrete event models for manufacturing systems are introduced, and a definition of the control of discrete event models is given. The next chapter, Chapter 3, describes the case of a multi-product flow line, to which both the common control strategies and the feedback control strategies are applied in further chapters.

2.1 A manufacturing system

A manufacturing system may be viewed as an arrangement of tasks and processes, properly put together, to transform a selected group of raw materials and semifinished goods to a set of finished products ([Alt97]). The manufacturing system does not only include the flow of products, but also the required information ([Hop00]). In [Alt97], the basic functions of manufacturing are considered procurement, production and distribution. This research project focusses on the flow of products and information required for the production function of the manufacturing system.

An important class of manufacturing systems is the class of flow lines. A manufacturing system is called a flow line when its stages are arranged in series and all products produces by this system follow the same sequence of processes ([Alt97]). Flow lines may be dedicated to a particular product or a range of products and are mainly used for mass production. Important properties for describing and controlling a manufacturing system, and especially a mass production system, are throughput δ, flow time φ, inventory or work-in-process (WIP) level w and utilization u. The next part of this section is dedicated to these quantities. Section 2.4 provides analytical relations for these quantities in a manufacturing system. The throughput δ describes the speed of products passing a certain section of the manufacturing system. In this report, the smallest unit for products is a lot. Because the throughput is a

derivative or a rate, it is measured by taking an average over a certain period of time. The dimension of throughput is lots per unit time.

Note that if the throughput at one section of the system is not equal to the throughput at another section, the amount of work in process between the two sections decreases or increases. In a steady state situation, the throughput is equal for all sections of the system. The flow time φ of a lot is the period of time it takes a lot to flow through the manufacturing system or a subsystem of the manufacturing system. Both process time and queueing time are part of the flow time. Flow time may be measured for each individual lot. The dimension of flow time is units time.

The WIP level w is the amount of work-in-process in the manufacturing system or subsystem of the manufacturing system at a certain moment. It consists of all lots present in that part of the system, so both of the lots being currently processed and of the queued lots. The WIP level is measured at a certain moment for a certain (sub)system. It is represented in lots.

Quantity	Symbol	Specified for:	Unit
WIP	w	time instant, (sub)system	lots
flow time	φ	lot, (sub)system	time
throughput	δ	time interval, system section	lots/time

Table 2.1: Dimensions of w, φ and δ

Table 2.1 summarizes the dimensions of the properties described in this chapter. These dimensions are important, for example for computing averages of these properties correctly.

The performance of a manufacturing system may be measured in numerous ways. Depending on the type of manufacturing system and the objectives of the system, different criteria may be defined. For example, a system designed for mass production requires different definitions for performance than a system to manufacture custom-built products. This research project focusses on controlling a manufacturing line for mass production. Important properties for measuring the performance of a mass production system are throughput, flow time and WIP level. Because of their importance, analytical relations for these properties are given in Section 2.4.

2.2 A discrete event model

A model of a system may be defined as a simplified representation of that system. Models of manufacturing systems may be used for the design and analysis of manufacturing systems. Manufacturing systems may be described using discrete event models. The discrete event model has only a limited number of states. At certain moments, the system instantly switches from the one discrete state to the other. In this report, such a change of state is referred to as an event. Distinction may be made between controlled and uncontrolled events. Controlled events are events that can be directly controlled. An example of a controlled event is the authorization of a machine to process a lot. Uncontrolled events are events that cannot be directly controlled, because they are a reaction of the observed system. An example of an uncontrolled event is the breakdown of a machine or the end of a (stochastic) processing time. The distinction between controlled and uncontrolled events may be compared to the distinction between inputs and outputs (or disturbances) of a

dynamic system (Chapter 5). This project uses discrete event models of manufacturing systems to perform research on controlling a manufacturing system

The observed discrete event model consists of several processes. It contains a generator, buffer, machine and exit processes. Buffer and machine processes are the equivalent of buffers and machines in a manufacturing system. The buffer process receives lots through one channel and releases them through the second channel as soon as release is requested by a downstream machine or exit process. A machine process processes a lot as soon as it is authorized to do so. The generator and exit processes are responsible for releasing new lots into the manufacturing system and removing finished goods from the system. The generator process generates lots to be processed in the system. Lots may be generated uncontrolled, with a deterministic or stochastic arrival rate, or controlled, after request by a controlling process. At the end of the line, the exit process removes finished lots from the system. Other processes, e.g. a controller process, may be added to the model in order to adequately control the system.

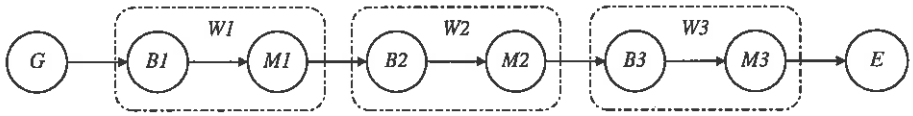

Figure 2.1: Model of the three machine manufacturing line

An example of a representation of a discrete event system is given in Figure 2.1. Each process is represented by a circle, and the flow of lots (or information) is represented by an arrow. Each workstation consists of a buffer and a machine. For the discrete event model, the properties throughput δ, WIP level w and flow time φ are defined equally as for the manufacturing system.

2.3 Controlling a discrete event model

Purpose of this research project is to compare the performance of several control methods for a manufacturing system. The manufacturing system is described by means of a discrete event model. This section provides a definition of control of a discrete event model of a manufacturing system and describes the purpose of controlling a manufacturing system. Controlling the model of the discrete event manufacturing system may be described as specifying or limiting the behavior of the processes it consists of. The control strategy defines the reactions of the manufacturing system to uncontrolled events, such as the arrival of customer demand or machine breakdown. The control strategy may be described as the set of rules defining at which conditions which controlled event should take place. These events may include for example the release of lots into the system by the controller and the authorization to take lots from a buffer to be processed at a machine. The behavior of the manufacturing system or discrete event model strongly depends on the control strategy. The purpose of the control strategy is to optimize the behavior of the discrete event model of the manufacturing system. Which control strategy to use depends both on which criteria are chosen for the performance of the manufacturing system and on the properties and characteristics of the manufacturing system.

This research project focusses on the control of a flow line. The properties of the manufacturing line under consideration are described in Chapter 3. Because this manufacturing

line is used for mass production, main targets of the control strategy are to achieve a high throughput with low flow times and WIP levels. Because throughput and flow times are considered important properties for this research project, the next section summarizes some analytical relations for flow time, throughput and WIP level.

Chapter 4 describes several commonly used control strategies and their implementation applied to the flow line described in Chapter 3. Chapters 5 to 9 describe an alternative for the common control strategies, using feedback control.

2.4 Analytical relations for a manufacturing system

The goal of this project is to compare the performance of several control strategies. Numerous performance criterions may be formulated and considered. This report focusses on the aspects of throughput δ, flow time φ and WIP level w. The relations between throughput, flow time and WIP level may be obtained using simulation experiments with discrete event models. Nevertheless, a great deal of knowledge may also be obtained by performing relatively simple analytical calculations. These calculations provide useful insight in the expected behavior of the system and help to understand and verify simulation results. Furthermore, the analytical relations may provide initial settings for the design of a control strategy.

This section provides analytical relations between WIP, flow time and throughput, as well as formulas for estimating queueing times. Furthermore, the section describes the phenomena of blocking and starvation. These phenomena cause a loss of throughput, for example in a system with a lack of storage space.

Little's law

Little's law describes the relation between the mean amount of WIP ($\bar{w}$) in a system in steady state, the throughput (δ) and mean flow time ($\bar{\varphi}$):

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

Note that Little's law only holds in steady state situations.

Queueing equations

For simple queueing systems, consisting of one workstation of buffer and a single machine, the following formula provides the waiting time φ_q in the upstream queue in front of a machine:

$$\varphi_q = \frac{c_a^2 + c_e^2}{2} \cdot \frac{u}{1-u} \cdot t_e. \tag{2.2}$$

This formula requires the squared coefficients of variation c^2 of both arrival (c_a^2) and processing (c_e^2), the mean effective processing time t_e and the utilization u. The squared coefficient of variation defined as the ratio of variance σ^2 and squared mean μ^2:

$$c^2 = \frac{\sigma^2}{\mu^2}. \tag{2.3}$$

The utilization of a one machine workstation may be defined as the ratio of the effective machine capacity r_e and the arrival rate r_a:

$$u = \frac{r_a}{r_e}, \tag{2.4}$$

with r_e being defined using the effective process time t_e:

$$r_e = \frac{1}{t_e}. \tag{2.5}$$

Note that u is required to be strictly less than one for stochastic systems. At a utilization level of one or higher, the workstation cannot process the arriving flow of work. In steady-state, the queue length grows to infinity and no steady-state situation is reached. The effective process time consists of natural process time t_0 and if necessary several disturbances such as random outages and setups (see [Hop00]).

For a so-called M/G/1-system ([Ken51]), the results of queueing equation (2.2) are exact ([Buz93]. An M/G/1-system is a queueing system with a memoryless inter arrival time distribution and general process time distribution and one (parallel) machine. For other systems, the results are only an approximation.

Equation 2.2 may be used for a series of machines. Lots departing from the first workstation arrive at the next workstation. Therefore, c_a^2 of a downstream buffer equals c_d^2 of the upstream machine. The relation between both coefficients of variation and the utilization u are given by the linking equation ([Buz93]):

$$c_d^2 = (1 - u^2) \cdot c_a^2 + u^2 \cdot c_e^2. \tag{2.6}$$

Note that (2.6) clearly shows that at high utilization, the departure coefficient is primarily determined by process time distribution, and by the arrival coefficient at low utilization.

The results of linking formula (Formula 2.6) are only exact in case both processing times and inter arrival times are distributed exponentially (M/M/1). In other cases, the results are approximations only.

An example of an M/M/1-system is a push controlled system with exponentially distributed order inter arrival times and exponentially distributed service times. For each workstation, the average queueing time φ_q can be computed. The push controlled system and its performance are described in Chapter 4. Note that controlling the authorization of machines or the release of lots in many cases introduces a dependence between inter arrival times of sequential lots. Queueing equation (2.2) are only valid in case of memoryless inter arrival time distribution.

Blocking and starvation

In the previous section, the effective process time t_e and effective capacity r_e have been introduced because the queueing equation (2.2) requires the utilization u of the workstation. For a stand-alone workstation with infinite buffer capacity, all arrival rates r_a strictly smaller than the effective capacity r_e are allowed. In that case, the throughput equals the arrival rate. In some situations, the arrival rate is subject to more strict conditions. In this section, the phenomena of blocking and starvation are defined. After that, an example is given of a small system where blocking and starvation cause more strict constraints to the arrival

Figure 2.2: Two machine flow line: no intermediate buffer

rate. This example is meant to explain the phenomenon that a limited WIP level limits the maximum throughput that a system can achieve. The maximum system throughput is not only limited by the effective capacity of the slowest workstation, but also by the configuration of the system.

Blocking is the phenomenon of stoppages at work stations because of a lack of storage space or excessive accumulation in downstream stages ([Alt97]). Starvation is the phenomenon of a lack of jobs in the upstream stages, causing a loss of throughput as well. For a simple case, the magnitude of the influence of blocking and starvation on the maximum throughput of the system is derived.

Example 2.4.1 *Consider a simple flow line consisting of two machines in series. In front of the first machine is a buffer always containing sufficient lots. Finished lots always immediately leave the second machine. This may be represented by an exit process E. No buffer is available between both machines. Figure 2.2 shows the described system. Equivalent to the*

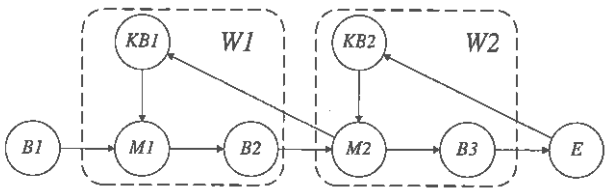

Figure 2.3: Two machine flow line as a kanban system

two machine system in Figure 2.2 is a pull system with only one kanban per workstation, as shown in Figure 2.3. For explanation about kanban controlled systems such as pull control is referred to Chapter 4.

The maximum throughput of this system is not only bounded by the machine capacity r_e. Assume that both machines are processing a lot. First, machine 1 finishes processing the lot. The lot can only leave the machine after machine 2 has also finished processing. A loss of throughput because of blocking occurs. Now assume that machine 2 finishes processing first. The machine cannot start processing the next lot until processing has been finished at machine 1. A loss of throughput because of starvation occurs. Because of these losses, the system's processing rate is not determined by processing times of single machines, but by the maximum of both processing times.

Let X_1 and X_2 denote the (stochastic) processing times of machines 1 and 2. Let Z denote the maximum of stochastic functions X_1 and X_2:

$$Z = max(X_1, X_2). \tag{2.7}$$

To determine the maximum allowed throughput, not only the effective process time t_e needs to be considered, but the sum of effective process time t_e and time lost due to blocking and

starvation t_b. This sum is denoted by t_{eb}:

$$t_{eb} = t_e + t_b. \tag{2.8}$$

For each product, the sum t_{eb} is determined by the slowest of the two machines. The mean of t_{eb} equals the expectation of the stochastic variable Z:

$$\bar{t}_{eb} = E(Z). \tag{2.9}$$

When at least one of the processing times is stochastic, the value of $\bar{t}_{eb}$ is larger than the maximum of $\bar{t}_{e1}$ and $\bar{t}_{e2}$.

As an example, $\bar{t}_{eb}$ is derived for assuming equal mean service times, but deterministic for one machine and exponentially distributed for the other. For two machines with stochastic process times, the approach is similar, but the calculations are more complicated.

As an example, $\bar{t}_{eb}$ is derived for a simple case. Assume that both machines in Figures 2.2 and 2.3 have equal effective process times, but for one machine, the service times t_{e1} are deterministic, while for the other machine the effective process times $\bar{t}_{e2}$ are distributed exponentially:

$$t_{e1} = \bar{t}_{e2} = 1/a. \tag{2.10}$$

To compute the expectation of the stochastic function Z of (2.7), the density function of Z needs to be determined. Let the process time for machine 1 be described by X_1 and the process time of machine 2 by X_2. The density function of the exponentially distributed function equals:

$$f(t) = a\, e^{-at}, \tag{2.11}$$

and the distribution function equals

$$F(t) = 1 - e^{-at}. \tag{2.12}$$

The integral of the distribution function of Z must equal one. Using $F(1/a) = 1 + 1/e$, the density function of Z equals:

$$\begin{cases} f(t) & = a\, e^{-at} & \text{for } t > 1/a, \\ \int f(t)dt & = 1 + 1/e & \text{for } t = 1/a, \\ f(t) & = 0 & \text{for } t < 1/a. \end{cases} \tag{2.13}$$

(3.16) *can be used to determine the expectation for Z and therefore (see (2.9)) $\bar{t}_{eb}$:*

$$\bar{t}_{eb} = \int_{-\infty}^{\infty} t\, f(t)\, dt \tag{2.14}$$

$$= \int_{\frac{1}{a}}^{\infty} t\, f(t)\, dt \tag{2.15}$$

$$= \frac{1}{a} \cdot \frac{e-1}{e} + (\frac{-1}{a} - t)\, e^{-at}]_{\frac{1}{a}}^{\infty} \tag{2.16}$$

$$= \frac{e+1}{a\, e}. \tag{2.17}$$

This result shows that the system can only process $\frac{a\, e}{e+1}$ lots during one unit time. Compared to the theoretical process rate r_e of a lots per unit time, $1/e$ (approximately 37%) of throughput is lost.

The example shows that for a certain configuration of manufacturing system and control method, a loss of throughput occurs with respect to the effective process rate, due to blocking and starvation. A similar loss of throughput may occur in other manufacturing systems using other control strategies when the storage space or the amount of WIP is limited. Chapter 4 shows similar results from experiments.

It might be possible as well to define the effective process time t_e in such a way that the losses because of blocking and starvation are included in t_e. The occurrence of blocking and starvation depend on the configuration of the manufacturing system and the control method. For each control method, a different t_e would have to be computed. Furthermore, detailed effective process time computations are beyond the scope of this research project. For more details about the effects of blocking on t_e is referred to [Wul02].

This chapter introduces the concept of a manufacturing system and some of its important properties, discrete event models of a manufacturing systems and control of discrete event models of manufacturing systems. Furthermore, analytical relations for important quantities in manufacturing systems are provided. The next chapter describes one specific manufacturing system. This system is used as a test case. Chapter 4 to 9 describe control methods. These control methods are all applied to the case first described in Chapter 3.

Chapter 3

Case: a flow line

Goal of this research project is to develop a method for using feedback control for controlling a manufacturing system. The performance of that control strategy may be compared to the performance of more common control methods. To measure and compare the performance of the control strategies, a case has to be developed. The control strategies are all applied to this case. On the one hand, the case has to be challenging, requiring a good control strategy. On the other hand, the case must be general, in order to avoid that the results can only be applied specifically to that case and not to other cases. The case of a multi-product flow line studied by Krishnamurthy and Suri in [Kri00] meets the conditions. The case shows that —contrary to the common idea— in some cases Push controlled systems outperform Pull systems. Both a specific and a more general variant of this case are described in the next section. In the following chapters, both common control strategies and feedback control are applied to the case.

3.1 The multi-product flow line

This section describes the basic manufacturing system used as a case to test the performance of several control methods. Subject of this study is a manufacturing system consisting of three workstation. Each workstation consists of a buffer and a single machine. The workstations are oriented in series as a flow line. A number of N different product types is produced in large quantities. For this study, the number of product types N is set to 8.

The machines are assumed not to have setup times and not to fail. The machines are assumed to be single lot machines. Processing of lots never fails and lots do not re-enter the system. Initially, infinite buffer capacities are assumed. Process times are modelled as Poisson processes. Demand inter arrival times are modelled as Poisson processes, unless

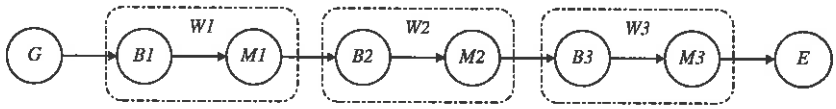

Figure 3.1: The three machine multi-product flow line

indicated otherwise. Initially, the demand is equally divided over the eight different product types. Assuming Poisson processes for inter arrival times and process times yield an M/M/1-system ([Ken51]), simulating an environment with high variability ([Kri00]). For M/M/1-systems, the queuing relations described in Section 2.4 can be used. Initially, the mean service time for all machines equals 1 [units time], independent from the type of product that is processed. This may be denoted as:

$$t_{e_{ij}} = 1 \text{ for all product types } i \text{ and machines } j. \tag{3.1}$$

Thus, if no variability would be present, the maximum throughput of the system would have been 1 [lots/unit time], and the minimum flow time 3 [units time]. Because the service times are assumed to be exponentially distributed, the variability causes a lower throughput and a longer flow time. Because of the equal machines, none of the machines is a specific bottleneck.

The situation with equal process times and demands for each product type, is referred to as the *equal product types* case. The next section describes the case of eight product types divided into two groups with different demand and service times.

3.2 Non-equal product types

The previous section described the situation of all product types having equal properties. This section introduces a situation with more variability due to differences between the product types. This situation may simulate a disturbance in demand mix and in process times, which are likely to occur in practice ([Kri00]). The control strategy must also be able to react adequately to this more variable situation.

Assume that the 8 product types, which initially had equal properties, are divided into two product type groups with different properties. These two product type groups are assumed to have different (mean) demands and mean process times. This situation is referred to as *non-equal product types*.

The mean demand for product i is denoted by D_i. The mean demand of the first four product types does not equal the mean demand of the last four:

$$D_i = \lambda_1 \text{ for } i = 1, \ldots, 4 \tag{3.2}$$

$$D_i = \lambda_2 \text{ for } i = 5, \ldots, 8. \tag{3.3}$$

The two product type groups also have different mean service times. The mean effective processing time of product i on machine j ($t_{e_{ij}}$) is given by:

$$t_{e_{ij}} = \tau_1 \text{ for } i = 1, \ldots, 4 \tag{3.4}$$

$$t_{e_{ij}} = \tau_2 \text{ for } i = 5, \ldots, 8. \tag{3.5}$$

Two parameters, k_d and k_p are defined as the ratio between the demands and process times for the two product type groups:

$$k_d = \frac{\lambda_1}{\lambda_2} \text{ and } k_p = \frac{\tau_1}{\tau_2}. \tag{3.6}$$

Note that for $k_d = k_p = 1$ both product type groups have equal properties: the equal product types case. To be able to compare the performance of the non-equal product types system with the equal product types system, the values for λ and τ may be chosen such that the mean process time and mean demand over all product types equals these of the equal product types case. This is ensured by (3.7) and (3.8). The mean demand of the two product type groups equals D:

$$N \cdot \frac{(\lambda_1 + \lambda_2)}{2} = D. \tag{3.7}$$

The mean process time over all product types equals the fraction of demands multiplied by the corresponding process time τ. The mean process time for all products in the equal product types case is 1.

$$\frac{N}{2} \cdot (\lambda_1 \tau_1 + \lambda_2 \tau_2) = 1. \tag{3.8}$$

The mean demands λ_1 and λ_2 should be expressed as a function of D, N, k_d and k_p. For that purpose, (3.6) is substituted in (3.7):

$$(\lambda_1 + \frac{\lambda_1}{k_d}) = \frac{2D}{N}. \tag{3.9}$$

Equation (3.9) can be rewritten into the following two expressions for λ_1 and λ_2.

$$\lambda_1 = \frac{k_d \cdot 2D}{N(k_d + 1)}, \tag{3.10}$$

$$\lambda_2 = \frac{2D}{N(k_d + 1)}, \tag{3.11}$$

Substitute the expression for k_p in (3.6), and (3.10) and (3.11) into (3.8) and divide both numerator and denominator by $\frac{D}{k_d+1}$ to obtain:

$$k_d \tau_1 + \frac{\tau_1}{k_p} = k_d + 1. \tag{3.12}$$

Equation (3.12) can be rewritten into the following two expressions for τ_1 and τ_2:

$$\tau_1 = \frac{(k_d + 1) \cdot k_p}{k_d \cdot k_p + 1}, \tag{3.13}$$

$$\tau_2 = \frac{k_d + 1}{k_d \cdot k_p + 1}. \tag{3.14}$$

This chapter describes the properties of a case to be used in further chapters to measure the performance of different control methods. To variant of a multi-product flow line are described: the equal product type case of all eight product types having equal properties, and the non-equal product types case of two different product type groups, having different mean service times and demands. Section 2.4 provided relations from queueing theory for queueing times in for example manufacturing systems. These relations may directly be applied to the equal product type case. For the non-equal product type case, few additional derivations have to be made. These derivations are presented in the following section.

3.3 Coefficients of variation for Non-equal product types

Section 2.4 provides sufficient tools for performing calculations on the equal product types case. In this section, additional formulas are given, required to perform equal calculations to the system with non-equal product types.

Queueing and linking equations (2.2) and (2.6) require a value for the squared coefficients of variation of inter arrival times and of process time c_a^2 and c_e^2. For non-equal product types, each product group has its own coefficient of variation. To be able to use (2.2) and (2.6), an effective value has to be determined for both c_a^2 and c_e^2.

Coefficient c_e^2

Goal of this subsection is to find a relation for an effective c_e^2 for two product groups with different demand and service times. Recall that the definition of c^2 in (2.3) uses the mean μ and the variance σ^2. The mean process time is known from (3.8): 1. For derivation of the variance, the following equations are used:

$$\sigma^2 = Ex^2 - (Ex)^2. \tag{3.15}$$

The expected value of x, $E(x)$ is the mean. The expected value of x^2, $E(x^2)$ is obtained by solving the following integral, containing the probability density function f:

$$E(x^2) = \int_{-\infty}^{\infty} x^2 f(x)\, dx. \tag{3.16}$$

Both the service time for product group one and for product group two are distributed exponentially. Their density functions f_1 and f_2 are:

$$f_1(t) = \frac{1}{\tau_1} e^{-\frac{1}{\tau_1}t} \quad \text{and} \quad f_2(t) = \frac{1}{\tau_2} e^{-\frac{1}{\tau_2}t}. \tag{3.17}$$

The probability that a lot is drawn from the first product type is proportional to the fraction r of demand for that product type:

$$r = \frac{\lambda_1}{\lambda_1 + \lambda_2}. \tag{3.18}$$

Now the effective density function f_e for process time of either the one or the other product type group equals:

$$f_e(t) = r(\frac{1}{\tau_1} e^{-\frac{1}{\tau_1}t}) + (1-r)\frac{1}{\tau_2} e^{-\frac{1}{\tau_2}t}. \tag{3.19}$$

Solve integral (3.16) for density function f_e from (3.19) to obtain the following expression for the variance σ^2:

$$\sigma^2 = (2r - r^2)\tau_1^2 + (1 - r^2)\tau_2^2 - (2r - r^2)\tau_1\tau_2. \tag{3.20}$$

Use (2.3) to compute c_e^2. For this particular case, with $\mu = 1$, the squared coefficient of variation equals the variance:

$$c_e^2 = (2r - r^2)\tau_1^2 + (1 - r^2)\tau_2^2 - (2r - r^2)\tau_1\tau_2 \quad \text{for } \mu = 1. \tag{3.21}$$

Note that the variance only equals 1 in case $r = 1$, which corresponds to $\lambda_1 = \lambda_2$. For situations other than $r = 1$, the variance does not equal the mean, so service times are not distributed exponentially.

Coefficient c_a^2

In the previous subsection, an expression has been derived for an effective c_e^2. This subsection describes the derivation of a relation for determining the effective squared coefficient of variation of inter arrival times, c_a^2, for two different product type groups having different inter arrival time distributions.

The inter arrival time distribution of the products 1 to 4 (product type 1, see (3.2)) have the following density function:

$$f_i(t) = \frac{1}{\lambda_1} \cdot e^{-\frac{1}{\lambda_1}t} \text{ for } i = 1, \ldots, 4. \tag{3.22}$$

For the four products of product type 2, the density function of the inter arrival time distribution equals:

$$f_i(t) = \frac{1}{\lambda_2} \cdot e^{-\frac{1}{\lambda_2}t} \text{ for } i = 5, \ldots, 8. \tag{3.23}$$

An exponential distribution of inter arrival times implies a Poisson distribution for the number of arrivals. The following equations give the probability of k arrivals of product type i in the interval from 0 to t:

$$P_i(k) = \frac{(\frac{1}{\lambda_1}t)^k}{k!} \cdot e^{-\frac{1}{\lambda_1}t} \text{ for } i = 1, \ldots, 4, \tag{3.24}$$

$$P_i(k) = \frac{(\frac{1}{\lambda_2}t)^k}{k!} \cdot e^{-\frac{1}{\lambda_2}t} \text{ for } i = 5, \ldots, 8. \tag{3.25}$$

To obtain the probability P_0 that no arrivals occur in the interval from 0 to t, use (3.24) and 3.25 and substitute k with 0. For eight independent processes, the probability that no arrivals occur for any of these processes, equals the product of the individual probabilities. This probability $P_{0,\text{total}}$, is:

$$P_{0,\text{total}}(t) = e^{-(\frac{4}{\lambda_1}+\frac{4}{\lambda_2})t}. \tag{3.26}$$

The probability of one or more arrivals occurring in the interval equals $1 - P_{0,\text{total}}$. The probability density function is obtained by differentiating $1 - P_{0,\text{total}}$:

$$f(x) = (\frac{4}{\lambda_1} + \frac{4}{\lambda_2})e^{-(\frac{4}{\lambda_1}+\frac{4}{\lambda_2})t}. \tag{3.27}$$

Equation (3.27) shows that the inter arrival times remain exponentially distributed regardless of mix parameters k_d and k_p. The parameter of the effective exponential distribution is $(\frac{4}{\lambda_1} + \frac{4}{\lambda_2})$. The squared coefficient of variation of an exponential distribution equals 1 (see (2.3)). The squared coefficient of variation of arrival for the non-equal product types variant equals 1 regardless of mix parameter k_d. Because of that, c_a^2 equals 1 in case demand arrival pattern As mentioned in Section 3.1, for the equal product types case, a workstation is an M/M/1-system. The non-equal product types case is only an M/M/1-system for $k_p = 1$. For $k_p \neq 1$, the system is M/G/1. Note that c_a^2 is not a function of k_d. This implies that the performance of a Push system (see Chapter 4) must be equal for all values of k_d as long as $k_p = 1$. This fact is in contradiction to the simulation results shown in [Kri00].

In this chapter, two variants of a multi-product flow line are described. In the next chapters, several control methods are described and applied to this flow line to measure and compare the performance of these control methods. The next chapter describes commonly used control methods. In that chapter, the analytical methods from Section 2.4 are used to verify and interpret the results of simulations with these control methods. The chapters after that, Chapter 5 to 9 describe an alternative control method, using feedback control.

Chapter 4

Common control strategies

Subject of this chapter are several common control methods for manufacturing systems. Section 2.3 described the control strategy as the set of rules defining which controlled events should take place at which conditions. These sets of rules may be decomposed into several components, each responsible for a part of the control strategy for the manufacturing system. In this chapter, first some of these components for control strategies are given. After that, these components are used to describe four more or less commonly used control methods: *push, pull, conwip* and *POLCA*. At last, these four control methods are all applied to the test case of the multi-product flow line that is described in Chapter 3. Simulation experiments are used to measure the performance of the control methods. Each control method has its dedicated section, in which the implementation of the simulation experiments is described and the results are evaluated.

4.1 Components of control strategies

In Section 2.3 the suggestion is made to define a control strategy as the set of rules defining and restricting the controlled events as reaction to the uncontrolled events. Several control methods are more or less commonly known in different variants. For this report, the sets of rules of these control methods are decomposed into several components. Each component corresponds to a concept used in common control methods. Designing a control strategy may now be regarded as choosing how to use these components. To illustrate this concept, Section 4.2 describes the four commonly known control strategies of *push, pull, conwip* and *POLCA* as combinations of the properties introduced in this section. When designing control methods, one is not restricted to these commonly known methods: alternative methods may control methods may be designed combining these components as well. First, the components are described.

MRP release times

In [Hop00] is described how in the 1970's the use of Material Requirements Planning (MRP) grew explosively. MRP replaced older manufacturing control systems that mainly used some variant of statistical reorder points. The basic idea of MRP is to use the knowledge about

planned final product demand to plan the production of its components. This subsection describes the use of MRP release dates in controlling the discrete event model.

In some cases, processing lots too early should be avoided. A lot that is finished too early only claims storing capacity until the due date. Furthermore, machine capacity is claimed, which might better be used by other orders. The use of MRP release dates avoids processing lots early. The MRP schedule contains the times and dates that jobs should be processed in order to be available right in time. Lots are only allowed to be processed when the corresponding release time has passed. This causes the (seemingly illogical) phenomenon of lots queueing in front of an idle machine.

Basic lot release methods

The generator process and the machines need a trigger when to release or process which type of lot. This section describes the most basic method for releasing these triggers.

The most basic method for authorizing a machine is sending a trigger for that machine to process a lot as soon as the machine is idle and a lot is available in the upstream buffer.

Although slightly more complicated, a similar release method may be used for the generator process. The manufacturing system is restricted by a utilization constraint: the throughput of the manufacturing system cannot exceed the throughput of the system bottleneck. Therefore, the generator process should release lots at a lower rate than the maximum throughput of the bottleneck. In case lots are released into the system at a rate equal to or exceeding the throughput of the bottleneck, the utilization of the bottleneck approaches one, yielding an unstable system. The amount of WIP does not reach steady state, but rises to infinity. The generator process is allowed to release lots at a rate strictly lower than the bottleneck's throughput. The lots may be released at a fixed rate, or at a variable rate. The latter option may be used to model a system using MRP for the release of orders. The fixed lead times are subtracted from the (stochastic) arrival times of the orders.

Work In Process level control

This section describes a more sophisticated method for controlling the authorization of machines and the generator process, limiting the amount of WIP in (part of) the system.

The WIP level is an important quantity in controlling a manufacturing system. The amount of WIP is physically limited by the limited storage space for inventory (modelled by finite buffer capacities). Reaching this limit may cause the phenomenon called blocking: a finished product cannot leave the machine because it cannot be stored and the machine is blocked for next products. Furthermore, Little's law (see Section 2.4) shows the relation between WIP and flow time: an increase of WIP also causes an increase of flow time. In controlling manufacturing systems, an aim may be to reduce flow time, for example to be able to react to customer demand more quickly. These two reasons justify limiting the amount of WIP in the manufacturing system.

A method to limit the amount of WIP is the use of authorization cards. These cards are also called *kanbans*, after the Japanese word for card [Sur00]. Within (part of) the system, a limited number of cards is available, and the amount of WIP is not allowed to exceed the number of cards. This is achieved by only allowing a lot to enter that part of the system

when a (corresponding) kanban is available. The kanban is then attached to the lot and goes with the lot through the manufacturing system. Only at a certain stage of the line, at the end of the kanban loop, the kanban leaves the lot and becomes available for next lots at the beginning of the loop again. For modelling purposes, in this report available kanbans are assumed to wait for a next lot in a kanban buffer (KB). Figure 4.1 shows the

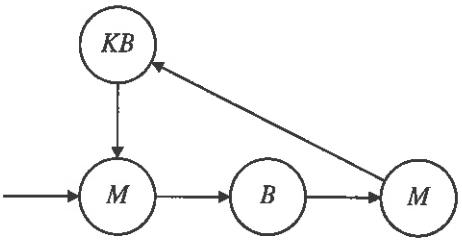

Figure 4.1: Example of a small kanban loop

smallest possible kanban loop. Lots flow through the system from left to right. Machine M is only allowed to process a lot when a (corresponding) kanban is available in kanban-buffer KB. The kanban goes with the lot, also after processing on M has finished, and returns to kanban-buffer KB only after the lot has left buffer B. Figure 4.2 shows an example of a larger

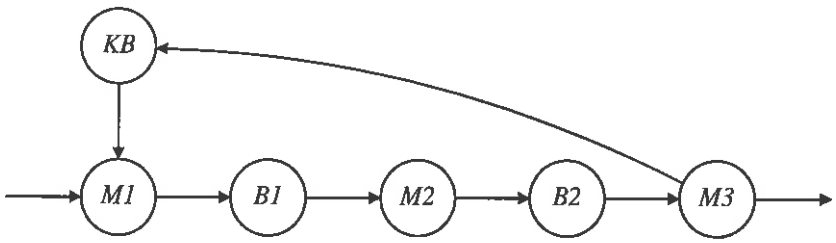

Figure 4.2: Example of a larger kanban loop

kanban loop: the first machine is only allowed to process a next lot when a (corresponding) kanban is available in kanban-buffer KB. The kanban goes with the lot and only returns to the kanban-buffer after the lot has left the last buffer. Note that it is also possible for kanban loops to overlap each other, or to have one loop within the other loop.

When kanbans are used to control a manufacturing system producing different types of products, distinction may be made between the use of product-specific kanbans and generic kanbans. In the first case, the type of kanban determines which lot is to be processed next. In the second case, with non-specific cards, the kanban does not determine which type of product is to be produced. For a buffer, FIFO may be used, or the order of products in a backlog order list may determine the type of product to be started.

The number of kanbans determines the maximum utilization of the kanban controlled part of the system. For a description of the mechanism causing this maximum utilization is referred to Section 2.4.

Scheduling of buffered lots

In case more than one lots are present in a buffer, a selection has to be made which lot to process first. This subsection describes some possible sorting methods.

In a system with WIP level control, a lot is only allowed to be processed in case a (corresponding) kanban is available. The availability of kanbans determines which lots are authorized by WIP control. Furthermore, lots have to be available in the upstream buffer. When MRP release times are used, only lots for which the release time has passed are allowed to be processed. Only from the lots obeying these three conditions, a lot may be selected to be processed. The order of lots may for example be sorted based on the time that the lots have been waiting in the buffer, or time that the kanban has spent in the buffer, or the MRP due dates for that process.

Now that some components of control methods for manufacturing systems have been described, control methods may be created by combining these components. The next sections describe several common control methods. These methods are all applied to the case of the flow line introduced in Chapter 3. Each of these control methods has a section dedicated to it. For each control method, a simulation model of the controlled flow line is made, experiments are performed and the results of the experiments are discussed. The chapter ends with a comparison of the performances of the control methods applied to the flow line.

4.2 Commonly used control methods

This section describes several commonly used control methods for manufacturing systems. These methods are described using the components from Section 4.1. Models of these control methods are made and applied to the flow line case. After that, experiments are performed to measure and compare the performance of the control methods. Purpose of these experiments is to compare the results to the results of a control method using feedback control, as is described in Chapters 5 to 9.

Each manufacturing system requires its control strategy. The previous chapter described components of control strategies. In this chapter, these components are combined. Four widely used control strategies may also be described as a combination of choices for these parameters. For these control methods, the method for WIP level control (kanban loops), material release and scheduling of buffer lots are summarized.

Push control

The most basic control strategy is called push control. No WIP level control method is used and new lots are just 'pushed' into the system, at moments corresponding to the MRP due date minus the lead time. During processing, the MRP release times are not considered anymore. The buffers use a first-in-first-out (FIFO) schedule. The demanded throughput determines the release rate of lots. The amount of inventory, or Work In Process (WIP), is a result of the chosen release rate.

For each material release rate strictly lower than the theoretical maximum throughput, a dynamic equilibrium occurs. The amount of work in the buffers varies. As long as one

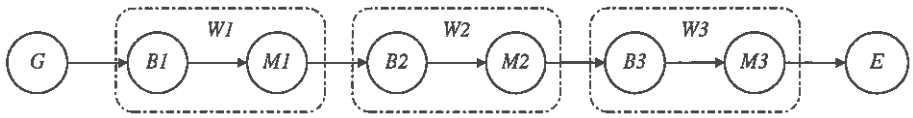

Figure 4.3: Flow line with Push control

or more lots are available in the buffer, the machine is not idle. As soon as no lots are available, the machine is idle, causing a loss of throughput. The higher the average WIP level, the smaller is the probability that no lots are available. This results in less idle time and a higher throughput.

Pull control

A simple method using the advantages of maximum WIP level control by kanbans is pull control. In a pull system, all machines are authorized by small kanban loops, with product specific kanbans. The number of kanbans in each kanban loop (consisting of machine, buffer

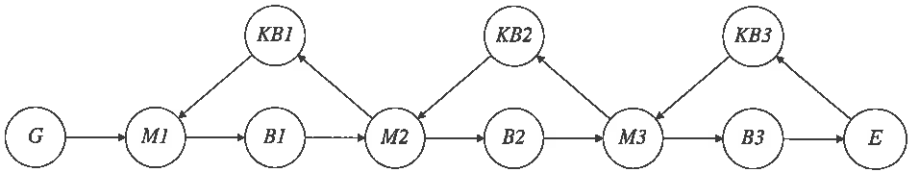

Figure 4.4: Flow line with Pull control

and kanban buffer) determines the maximum amount of WIP in that part of the system. A next lot is only allowed to enter the subsystem after another lot has left the subsystem. MRP release times are not used.

Conwip control

Conwip control is an example of what [Kri00] refers to as *hybrid control*, because it uses both properties from push and from pull control. Conwip control uses one long kanban loop, stretching from the first machine to the last buffer. In this way, the total amount of WIP in

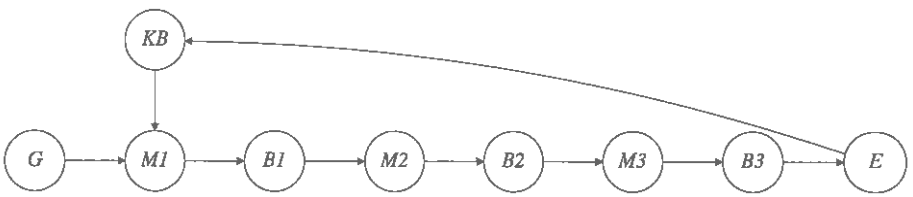

Figure 4.5: Flow line with Conwip control

the system is limited to the number of kanbans in the system. When the maximum number of lots are in the system, the next lot is only allowed to be released into the system after

another one has left it. The first machine is authorized by kanbans as in a pull system, the other machines do not use kanban authorization, as in a push system.

In case different product types are being processed at the same system, two different types of conwip control may be distinguished: *pull* conwip and *hybrid* conwip [Kri00]. The difference is the method used for scheduling the lots for the first machine. Pull conwip uses product specific kanbans. Therefore, the type of kanban determines which type of lot is set up. Hybrid conwip uses generic kanbans and a list of backlog orders is used to determine which type of product is set up next.

MRP due dates are not used for authorizing machines.

Conwip control uses kanbans to limit the level of WIP in the system. Conwip uses only one (large) kanban loop for the entire system, instead of a small loop for each machine (Pull control). This causes a lower minimum number of kanbans: the lowest number of kanbans equals one kanban for each product type (for pull conwip) or even just one kanban (for hybrid conwip). For this reason, the resident WIP remains far less than using Pull control.

Pull Conwip control uses product specific kanbans. The type of product that has to be released at the start of the line is determined by the type of kanban that is available. For that reason, customer demand has to be modelled by the Exit process that determines which type of lot is taken out of the last buffer. This model of customer order arrivals is equal to the model used for Pull control.

POLCA control

POLCA control is another hybrid control strategy. The name is an abbreviation for Paired cell Overlapping Loops of Cards with Authorization. The strategy uses both MRP due dates and generic kanbans to determine whether a machine is authorized to produce a certain lot. For determining MRP due dates, the relations for queueing times described in Section 2.4 are used. The kanban loops extend over two cells and the loops are overlapping: one cell is both the second cell of the first loop and the first cell of the next loop. The lots in the buffers

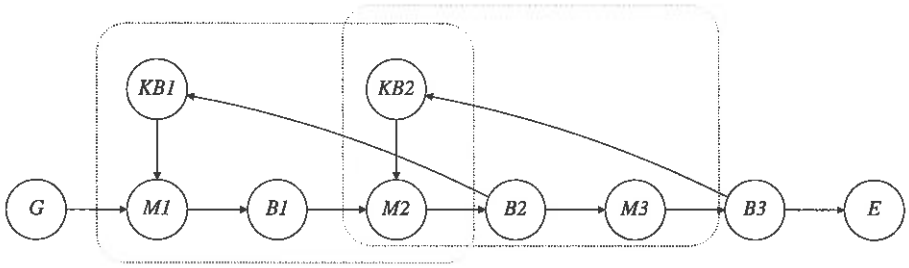

Figure 4.6: Flow line with POLCA control

are sorted on MRP due date. The system is specifically designed for a certain throughput: the number of kanbans corresponds to the mean WIP level of a pure push system.

4.3 Simulation experiments

In this section, the control methods introduced in the previous section are applied to the multi-product flow line described in Chapter 3. Simulation experiments are used to measure the performance of the control methods. For this research project, the main criteria for performance is the average WIP level and the throughput of the system. For that reason, experiments are performed for each control method to determine the amount of WIP required to achieve a throughput level. The throughput levels correspond to a workstation utilization of 50% to 90%. The experiments are performed for both the equal product types variant of the case ($k_d = k_p = 1$, see Chapter 3) and an unequal product types case with mix parameters $k_d = 0.2$ and $k_p = 5.0$. These mix parameters correspond to values of $1/6$ for r (3.18), and 3 and $3/5$ for service times τ_1 and τ_2 for product type 1 and 2. The squared coefficient of variation of the service time distribution is then $c_e^2 = 51/20 = 2.55$ (using (3.21)). Note that service times are subject to far more variability than in the equal product type case, with $c_e^2 = 1$.

For each described control method, first the model and the experiment are described. After that, the simulation results are presented and discussed.

The push control experiment

A model of the push controlled flow line has been made. A graphical representation of the push controlled flow line is given in Figure 4.3. The arrival of orders is modelled by the generator generating lots with exponentially distributed inter arrival times. The mean inter arrival times are computed using (3.10) and (3.11). The Exit process is always able to receive finished goods. The buffers are modelled as first-in-first-out (FIFO) buffers of infinite capacity. The machines have exponentially distributed service times. The mean service times for both product type groups are computed using Equations (3.13) and (3.14). For the χ implementation code is referred to Appendix B.

Experiments are performed for five throughput levels, corresponding to a utilization varying from 50 to 90%. The experiment starts with an empty manufacturing line. Only after a certain number of lots, the system has reached steady state. For that reason, an offset has to be determined. The mean flow time or WIP are computed using only the data of the lots after the offset. Furthermore, the number of lots for an experiment has to be determined. For estimating the required offset and number of lots, an experiment is performed at the highest observed utilization rate of 90%. Figure 4.7 shows the flow time per lot, the mean flow time computed over all previously finished lots and the mean flow time computed over all previously finished lots, neglecting the first 3000 lots. The figure shows that the mean flow time should not be computed for a too small number of lots. For all experiments, an offset of 3000 lots is chosen. The mean flow time is measured over the next 7000 lots. For each throughput level, the experiment is repeated 30 times. The mean WIP level $\bar{w}$ is computed using the measured mean flow time φ and throughput δ, and Little's law (2.1). For higher utilization levels, a higher WIP level is expected in the simulation experiments. The WIP level for the non-equal product types case is expected to be higher than the equal product types case, because the distribution of service times has a higher coefficient of variation.

As mentioned in Section 3.1, the flow line of the equal product types variant with push control consists of M/M/1 workstations. For M/M/1-systems, the queueing times

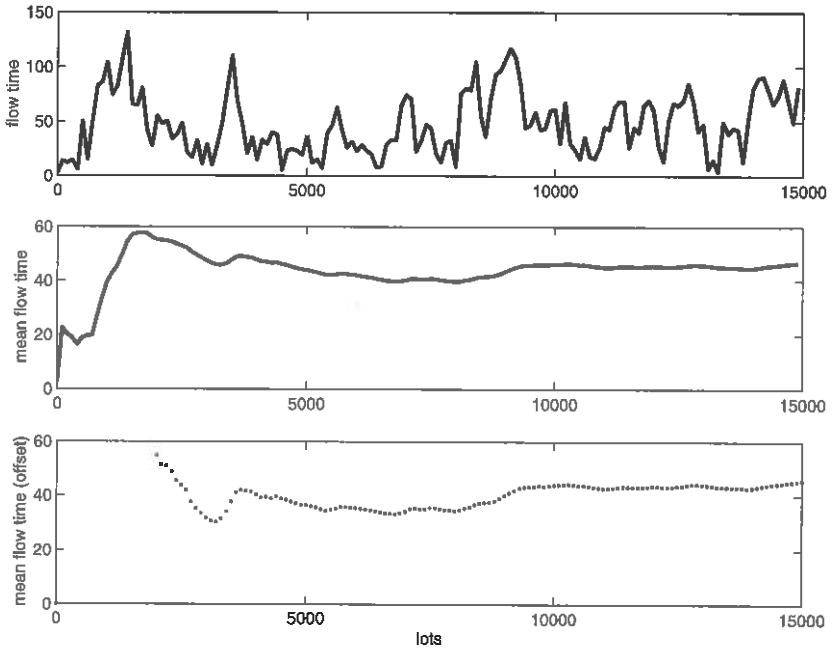

Figure 4.7: Push experiment: (mean) flow times ($u = 0.9$)

can be computed analytically using the relations presented in Section 2.4. Table 4.1 gives a summary of the results of the analytical relations applied to the multi-product flow line. These results should correspond to the results of the simulation experiments. The waiting

u	t_e	$\bar{\varphi}_{Q1}$	$\bar{\varphi}_{Q2}$	$\bar{\varphi}_{Q3}$	$\bar{\varphi}_{\text{total}}$	$\bar{w}$
0.5	1.0	1.0	1.0	1.0	6.0	3.0
0.6	1.0	1.5	1.5	1.5	7.5	4.5
0.7	1.0	2.3	2.3	2.3	10	7.0
0.8	1.0	4.0	4.0	4.0	15	12.0
0.9	1.0	9.0	9.0	9.0	30	27.0

Table 4.1: Calculated WIP for equal product types case

time in the queue in front of workstation i is denoted as φ_{Qi}. For $k_d = k_p = 1$, the waiting times are equal in front of each workstation. This is caused by the fact that the first term of Formula 2.2 equals 1 as long as process times and inter arrival times are distributed exponentially.

For the non-equal product types case with $k_p \neq 1$, the effective distribution of service times is not exponentially (see Section 3.3), yielding an M/G/1-system. In case of an M/G/1-system, the results of (2.6) are only an approximation. Because of that, only an approximation of the queueing times can be computed. Table 4.2 gives a summary of the results of the methods presented in Section 2.4 to the flow line.

Table 4.2 shows that the queueing times are the shortest for the first workstation, and the longest for the last workstation. This is expected, because the inter arrival times for the first machine are exponentially distributed ($c_a^2 = 1$). For the machines further in line,

u	t_e	$\bar{\varphi}_{Q1}$	$\bar{\varphi}_{Q2}$	$\bar{\varphi}_{Q3}$	$\bar{\varphi}_{\text{total}}$	$\bar{w}$
0.5	1.0	1.8	2.0	2.2	9.0	4.5
0.6	1.0	2.7	3.1	3.4	12.2	7.3
0.7	1.0	4.2	5.1	5.6	17.9	12.5
0.8	1.0	7.2	9.2	10.0	29.4	23.5
0.9	1.0	16.2	22.0	23.1	64.4	57.9

Table 4.2: Calculated WIP for non-equal product types case

c_a^2 depends on both c_a^2 and c_e^2 of the upstream workstation (2.6). For the observed case, c_e^2 is larger than 1 for all workstations. The further downstream, the larger becomes the influence of c_e^2 compared to the influence of the c_a^2 of lots at the beginning of the line.

Results of the Push control simulations

The results of the described simulation experiments are summarized in Table 4.3. For low throughput levels, little queueing occurs. Higher throughput levels, approaching the utilization limit of 1, require far more inventory.

$k_d = k_p = 1$ $\quad$ $k_d = 0.2, k_p = 5.0$

δ	$\bar{w}$
0.5	3.0
0.6	4.5
0.7	6.8
0.8	12.0
0.9	27.3

δ	$\bar{w}$
0.5	4.8
0.6	7.4
0.7	11.5
0.8	19.5
0.9	42.1

Table 4.3: Results of Push experiments

Figure 4.8 graphically represents the results.

The Pull control experiment

Simulation experiments have been done with a model for pull control. The maximum amount of WIP is controlled by setting the number of kanbans. For several numbers of kanbans, the flow time and throughput are determined. Section 2.4 describes the mechanism causing a maximum throughput for a given number of kanbans.

The exit process is used to model customer demand. Because goal of the experiments is to determine the maximum throughput for a certain WIP level, the order arrival rate must not be chosen lower than the maximum system throughput. This is realized by releasing the next order immediately after the previous one has been satisfied. Note that for product-specific orders, this method of modeling customer demand is not equal to taking a lot out of the last buffer as soon as it arrives. This method would result in a higher throughput at equal number of kanbans, but is not considered a realistic model of customer demand. The first method is more similar to the customer demand model used for push control. Because of this, a more fair comparison can be made. Nevertheless, note that this customer demand model only has stochastic order of product types, while the method used for modelling demand

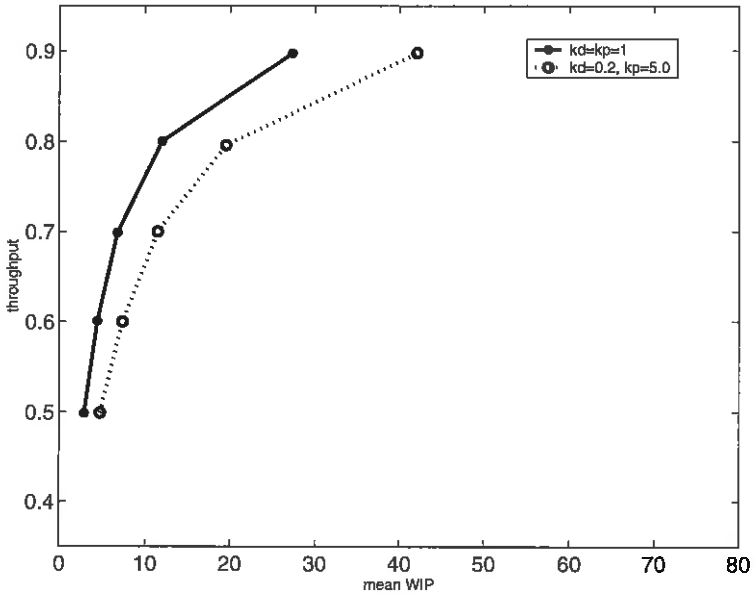

Figure 4.8: Performance of Push control

for push control also has stochastic demand inter arrival times. A graphical representation of the pull controlled flow line is given in Figure 4.4. For the χ implementation of the model is referred to Appendix B. Note that this χ implementation sends kanbans back to a kanban buffer after a lot has left a buffer instead of after a lot has arrived at the downstream machine. Though the results are equivalent, this implementation is not the common.

Note that a minimum number of kanban exists: each kanban loop must have at least one kanban for each product. Therefore, a number of k kanbans per product type per machine corresponds to a number of $8 \cdot 3 \cdot k$ kanbans for all machines and products. The lowest possible number of kanbans equals $8 \cdot 3 \cdot 1 = 24$, so the maximum amount of WIP $w_{\max}$ in this system can not be set less than 24. The WIP level caused by this minimum number of kanbans may be called the resident WIP [Kri00]. A lower demanded throughput does not result in a WIP level lower than the resident WIP. When no customer demands arrive, lots are processed until every kanban loop contains an equal number of lots and kanbans. In the simulation experiments, with high customer demand, a steady-state situation is expected to occur. In steady-state, the mean WIP level is lower than the resident WIP.

The number of kanbans k is varied from 1 to 3 per kanban loop per product type. To determine the mean WIP and throughput, for each number of kanbans 30 experiments are performed. Each experiment consists of 10000 lots. The throughput and average WIP are computed over the last 7000 lots.

Results of the Pull control simulations

Table 4.4 shows the throughput and mean WIP levels for $k = 1$, 2 and 3 kanbans. The minimum number of 1 kanban per product and kanban loop still achieves a throughput of 0.78 times the maximum throughput. Indeed, the WIP level shown in Table 4.4 is lower

$k_d = k_p = 1$

$w_{\max}$	δ	$\bar{w}$
24	0.78	18.3
48	0.88	32.1
72	0.92	44.4

$k_d = 0.2, k_p = 5.0$

$w_{\max}$	δ	$\bar{w}$
24	0.70	19.4
48	0.80	36.5
72	0.84	52.6
96	0.87	68.4

Table 4.4: Results of Pull experiments

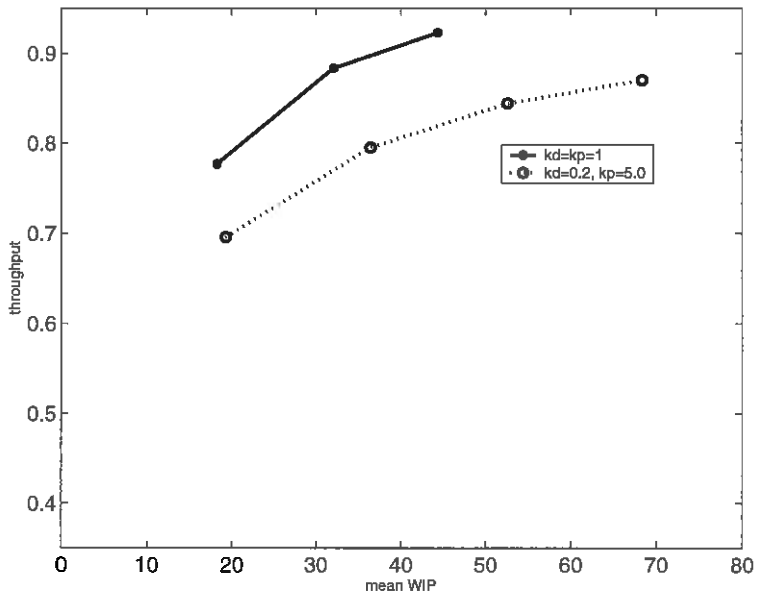

Figure 4.9: Performance of Pull control

than the resident WIP, because most of the time the buffers are not always entirely filled. Simulations with a lower demand rate would show a WIP level approaching the analytical resident WIP. The experimental results are graphically represented in Figure 4.9.

The (Pull) Conwip control experiment

Conwip is the control method with an upper limit to the amount of WIP in the entire controlled system. Pull conwip differs from hybrid conwip because of the use of product specific kanbans. The design variable for a conwip system is the number of kanban cards. The number of kanban cards defines the maximum amount of inventory in the system. This subsection describes the experiments using pull conwip control. The number of kanbans per product type is varied from 1 to 12. For a eight different product types, this implies a total number kanbans varying from 8 to 96. For each number of kanbans, 30 experiments are performed. The throughput δ and the mean flow time $\bar{\varphi}$ is measured for lots 3000 to 10000. Little's law (2.1) is used to determine the mean WIP level $\bar{w}$.

A graphical representation of the model of the pull conwip controlled line is given in

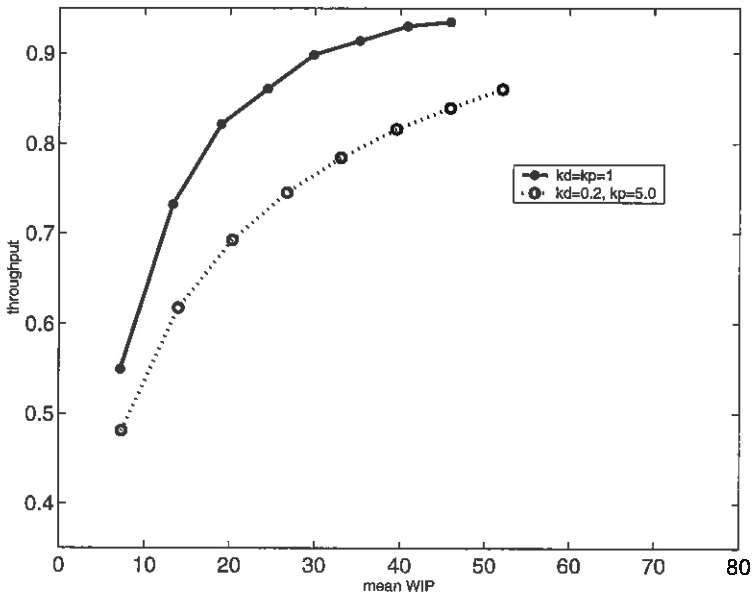

Figure 4.10: Performance of (pull) Conwip control

Figure 4.5. The generator process can always generate lots of the type demanded by the first machine. The exit process is modelled equal as the exit process for pull control simulations. An order for a specific product type is generated immediately after the previous order has been satisfied. No next order is released until the previous order is satisfied by a lot of the demanded product type. Because of this, the finished goods buffer will in general not be empty. For the code of the χ implementation is referred to Appendix B. Again, note that this χ implementation sends kanbans back to a kanban buffer after a lot has left a buffer instead of after a lot has arrived at the end of the line. Though the results are equivalent, this implementation is not the common.

Results of the (Pull) Conwip control simulations

The results of the pull Conwip simulations are presented in Table 4.5. The results show that (pull) Conwip slightly outperforms Pull control. Similar to pull control, also using product specific kanbans, (pull) conwip control suffers from a large amount of redundant resident WIP at higher utilization levels for the unequal product types variant ($k_d = 0, 2, k_p = 5.0$). This is caused by the fact that the number of kanbans is equal for each product type, while the demand for the one type is far larger than for the other type. A possible solution would be to adjust the number of kanbans per product type to the demand mix.

The (Hybrid) conwip experiment

A model of the (hybrid) conwip controlled flow line has been made. The graphical representation is equal to the model of the (pull) conwip model showed in Figure 4.5. The difference is the modelling of customer demand. Hybrid conwip uses generic kanban cards, not specifying the type of product that should be started up at the beginning of the line. Because

$k_d = k_p = 1$			$k_d = 0.2, k_p = 5.0$		
$w_{\max}$	δ	$\bar{w}$	$w_{\max}$	δ	$\bar{w}$
8	0.55	7.2	8	0.48	7.3
16	0.74	13.4	16	0.61	13.9
24	0.82	19.2	24	0.69	20.3
32	0.87	24.8	32	0.74	26.7
40	0.89	30.4	40	0.78	32.9
48	0.91	35.7	48	0.81	39.1
			56	0.84	45.5
			64	0.86	51.7

Table 4.5: The results of (pull) Conwip experiments

of that, a backlog order list is used to determine the type of lot inserted into the line by the generator process. Note that for this method of demand modelling, the performance of the line is independent of the number of product types. The only stochastic effect included in the model is the stochastic process times. The system does not contain lots for which no demand exists. Because the exit process is always able to accept all types of lots, the last buffer is always empty.

Experiments are performed for a number of kanbans varying from 3 to 30. For each number of kanbans, the throughput is determined by measuring the mean flow time from lot 3000 to lot 10000. Each experiment is repeated 30 times.

Results of the (Hybrid) conwip simulations

The results of the simulation experiments are presented in Table 4.6 and in Figure 4.11. The results show that the WIP levels required for a throughput level are low. These results are influenced by the few stochastic aspects in the model and the fact that the contents of only two buffers are observed, compared to three buffers for other control methods. The difference between the modelling of Hybrid and Pull Conwip systems causes the hybrid conwip system model to perform better. The Hybrid conwip model always immediately takes the lots out of the last buffer, while the pull conwip system has to wait for the appropriate type of product to be finished. This causes WIP in the last buffer and less throughput for the pull conwip controlled system.

The POLCA control experiment

The number of kanbans in each kanban loop of a POLCA system is specifically tuned for a certain throughput. The number of kanbans is equal to the expected mean WIP for the push controlled system. Section 2.4 described a method to compute this mean WIP level. Figure 4.12 shows the results of simulation experiments. For the POLCA controlled system, one may choose to model the release of lots into the system by releasing lots into the system at the demand rate for which the system has been designed, or only after demand from the first machine. In the first case, the amount of work in the first buffer becomes very large when the release rate comes close to the maximum system throughput. For that reason, the more stable option of releasing only after demand is chosen. This method of modelling

$k_d = k_p = 1$

$w_{\max}$	δ	$\bar{w}$
3	0.60	2.6
5	0.71	4.0
7	0.78	5.4
9	0.82	6.8
11	0.85	8.2
13	0.87	9.5
15	0.89	10.9
17	0.89	12.2
19	0.91	13.6

$k_d = 0.2, k_p = 5.0$

$w_{\max}$	δ	$\bar{w}$
3	0.52	2.5
5	0.61	3.9
7	0.67	5.3
9	0.71	6.7
11	0.75	8.1
13	0.78	9.4
15	0.8	10.7
17	0.82	11.9
19	0.84	13.3
21	0.85	14.6
23	0.86	15.7
25	0.87	17.2
27	0.88	18.3
29	0.89	19.7

Table 4.6: Results of hybrid Conwip experiments

customer demand does not include stochastic order inter arrival times and the stochastic sequence of product types does not affect the performance of the system.

POLCA control releases a lot at the due date of the lot, minus the service times, the expected queueing times and a safety lead time. For the simulation experiments, the MRP due dates for the second and the third machine are determined at the moment the generator process generates the lot for the first machine. The release time for the second machine equals the release time plus the expected service time for the first machine plus the expected queueing time for the second machine. The release time for the third machine equals the release time for the second machine plus the expected service time for the second machine plus the queueing time for the third machine. Safety lead times are not modelled.

Note that the limit of a POLCA system with a very high number of kanbans equals Push control, but with MRP release date authorization at all machines.

Results of the POLCA control simulations

Table 4.7 presents the results of simulation experiments for the POLCA controlled three machine flow line. The variable δ^* denotes the target throughput, for which the system was designed. The results from Tables 4.1 and 4.2 are used to compute the settings of the POLCA system. The number of kanbans in each POLCA loop is computed set at the WIP level for the same buffers and machines in the push controlled system. For example, for the number of kanbans for the first kanban loop, the first two buffers and machines are observed. In case of $k_d = k_p = 1$, Table 4.1 shows that for a utilization of 0.5, the queueing time for each workstation is 1 (unit time). The mean service time is also 1 (unit time) per machine. The expected flow time for two workstations equals 4 (units time). Little's law (2.1) states that the average WIP level $\bar{w}$ is $0.5 \times 4 = 2$. Because of that, the number of kanbans for the first POLCA loop is set to 4 as well. In Table 4.7, the number of kanbans for loop i is denoted as nk_i. Especially the experiments with the non-equal product types ($k_d = 0.2, k_p = 5.0$) show that the POLCA controlled system does not always reach the

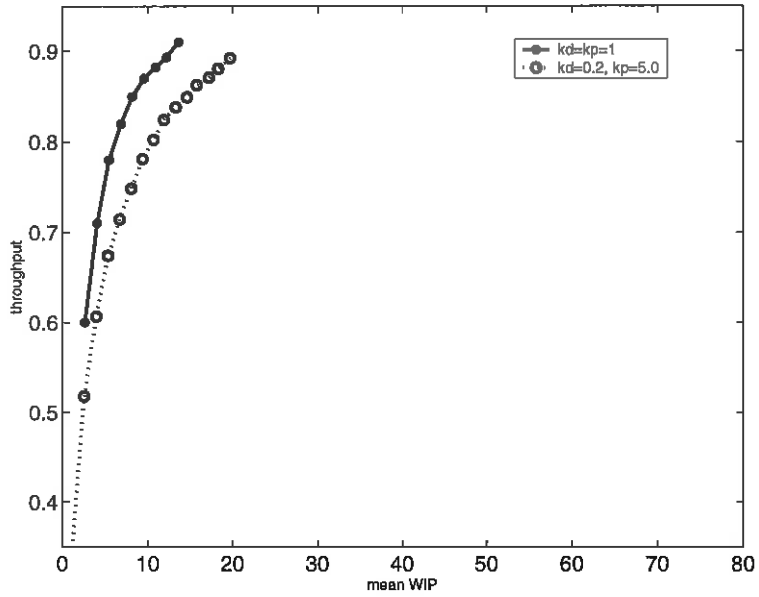

Figure 4.11: Performance of (hybrid) Conwip control

$k_d = k_p = 1$					$k_d = 0.2, k_p = 5.0$				
δ^*	k_1	k_1	$\bar{w}$	δ	δ^*	k_1	k_1	$\bar{w}$	δ
0.5	2	2	2.7	0.52	0.5	3	4	4.8	0.43
0.6	3	3	4.0	0.61	0.6	5	6	7.5	0.54
0.7	4	4	6.4	0.72	0.7	8	9	11.6	0.64
0.8	8	8	10.2	0.79	0.8	15	17	21.0	0.76
0.9	18	18	23.0	0.89	0.9	37	43	50.9	0.89

Table 4.7: Results of POLCA experiments

throughput for which the system was designed. This may easily be compensated by adding a number of kanbans. The settings for the MRP release dates for the second and third machine also influence the performance of the system. Table 4.8 show the performance of the POLCA controlled system when the MRP release times is not regarded. In that case, machines are always authorized to process the next lot when a kanban is available. The throughput of the system is higher than when the MRP releases are regarded. The benefit of using the MRP release dates only shows when for example tardiness of individual lots, or the variability of flow of outgoing lots is measured as well.

4.4 Evaluation of simulation results

The simulation results show that all discussed control methods perform better for the equal product types variant ($k_d = k_p = 1$) than for the variant with non-equal product types ($k_d = 0.2, k_p = 5.0$). One reason for this is the increased variability. Coefficient c_e^2 is more than twice as high for $k_d = 0.2, k_p = 5.0$ (see Section 4.3). The difference between

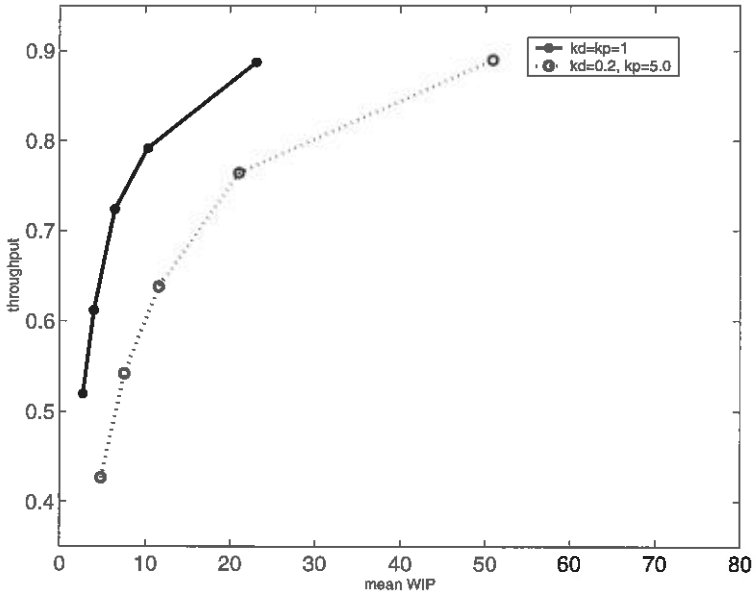

Figure 4.12: Performance of POLCA control

$$k_d = 0.2, k_p = 5.0$$

δ^*	k_1	k_1	$\bar{w}$	δ
0.5	3	4	3.9	0.60
0.6	5	6	6.1	0.68
0.7	8	9	9.1	0.77
0.8	15	17	16.2	0.85
0.9	37	43	38.3	0.93

Table 4.8: Results of POLCA experiments - no MRP

performance for the two product mix variants is even larger for systems using product specific kanbans: pull and pull-Conwip. This is caused by the fact that the same number of product-specific kanbans is used for both the highly demanded product types and the products types with a lower demand. At higher utilization levels, hybrid control methods, including aspects of both push and pull systems, outperform push and pull systems. This may be caused by the rigidness of both push and pull systems. Push control does not react to the state of the system. In case queue lengths are longer than average, for example due to stochastic effects, Push control does not react by releasing less lots into the system. Pull control is rigid as well. It strives for a fixed buffer level, regardless of the demands of the system. Furthermore, the methods for modelling demand introduce different stochastic effects for the different control methods.

The following chapters describe an alternative method for the common methods presented in this chapter. In chapter 5, the feedback controller is introduced.

Chapter 5

Feedback control for a manufacturing system

In the previous chapter, several common control strategies for a multi-product flow line have been described. A selection between these common control methods may be made using rules of thumb, or using simulation experiments and trial-and-error. An alternative for the common methods may be the use of feedback control. Feedback control is an extensively studied field of science, more often used to control for example mechanical systems than to control manufacturing systems. It may provide a better founded method to design a control strategy. Also, the performance of feedback controlled manufacturing system may be better than the performance of systems using more common control methods.

Chapters 5 to 9 describe a method for designing a feedback controller to a manufacturing system as the multi-product flow line described in Chapter 3. This chapter first introduces the concept of feedback control and some of the terminology of feedback control. After that, the approach for applying feedback control to the manufacturing system is described.

5.1 Feedback control

Control may be defined as the process of causing a variables of a system to follow a desired reference trajectory. In case of feedback control, output variables are measured. The controller uses the knowledge of the output variables to determine a proper reaction in order to affect the value of the output variables. This reaction is applied (or fed back) to the system. In [Ste84] three general classes of needs are distinguished to be satisfied by feedback control:

1. Suppressing the influence of external disturbances
2. Ensuring the stability of the process
3. Optimizing the performance of the process

The variables of the system may be classified into the following categories:

1. *Input* variables, which denote the effect of the surroundings on the process:

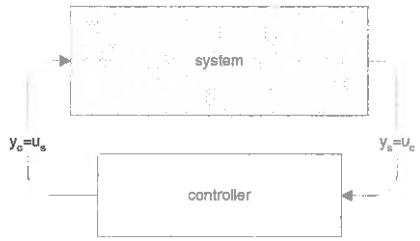

Figure 5.1: The concept of feedback control

- *Manipulated* (or adjustable) variables, if their values may be adjusted freely by operator or control mechanism.
- *Disturbances*, if their values are not the result of adjustment by an operator or control system.

2. *Output* variables, which denote the effect of the process on the surroundings:
   - *Measured* output variables.
   - *Unmeasured* output variables.

In general, input variables are denoted as u and output variables as y. In this research project, all input variables are assumed to be manipulated input variables, and all output variables are assumed to be measured output variables. These input and output variables of a system are graphically represented in Figure 5.1. For a basic feedback controlled system, the output y_s of the system is the input u_c of the controller. The output y_c of the controller is the input u_s for the system. The system output is measured and sent to the controller and the controller computes an appropriate output signal. The controller output signal is used as input signal for the system. These steps may be performed continuously (in a continuous time approach) or at discrete time steps only (in a discrete time approach). For a discrete time approach, a sample time is introduced. Periodically, a sample is drawn from the system output signal. The controller uses the measured system output to determine the appropriate control action. The control output is applied to the system. The sequence of sampling and applying is repeated after each sample time.

The following sequence of steps may be used to design a feedback controller:

- Define the system of interest. Define the objective of controlling the system and define the input and output variables of the system
- Construct a continuous dynamic model of the system. Many methods for designing a feedback controller require a dynamic model of the system. In this context, a dynamic model is a set of differential equations (or difference equations, in case of a discrete time approach) describing the behavior of the system. In this context, the behavior of the system is the response of the system's output variables to its input variables. The model class that is to be used depends on the system and the selected method for designing the controller.
- Design the controller. Numerous methods for designing a feedback controller are available. Which method to use depends amongst other reasons on the properties of the system and the model, and the purpose of the feedback controller.

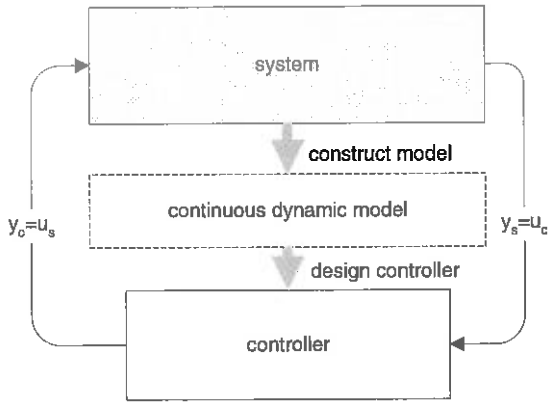

Figure 5.2: Feedback controller design

- Apply the controller to the system.

The approach for designing a feedback controller is represented by Figure 5.2. Now that the terminology and the general principles of feedback control have been described, feedback control can be applied to a manufacturing system.

5.2 Feedback control for the manufacturing system

This section describes an approach to apply feedback control to the manufacturing system described in Chapter 3. The system of interest, the input and output variables and the objective of controlling the manufacturing system are defined.

Controlling the manufacturing system

In Chapter 2, controlling a discrete event model is described as specifying or limiting the behavior of the processes of the discrete event model. The control method should define which controlled events take place at which conditions. Purpose of controlling the manufacturing system is to optimize the behavior of the manufacturing system. For this research project, the main indicators for performance are throughput and WIP level.

The system of interest is a manufacturing system. The controlled events (for example: machine i should start processing a product of type j) are the variables that denote the effect of the surroundings on the system. The controlled events should be adjusted by an operator or control system. For that reason, these controlled events are selected as the input for the system of interest, the manufacturing system.

Purpose of feedback control is to let the output variables of the system follow a certain reference trajectory. A purpose of controlling a manufacturing system is to optimize the performance of the system. For this research project, the objective of the control method is mainly to achieve a high throughput at low WIP levels. Feedback control can only achieve that objective when the system outputs are chosen such that the objective can be expressed using the system's output variables. For that reason, the WIP levels (in each buffer, for

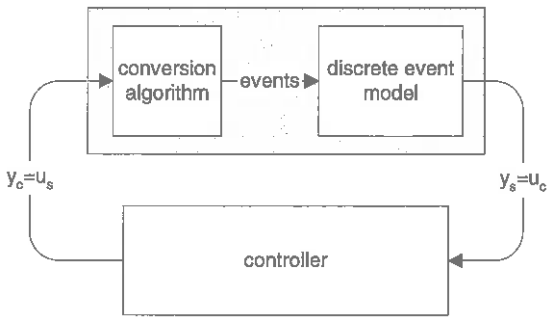

Figure 5.3: Feedback control for the discrete event system

each product) and the finished goods levels (for each product) are selected as the system's output variables.

The system of interest is defined as the manufacturing system described in Chapter 3. The system input is defined as the free events and the system outputs as WIP and finished goods levels. Objective of the controller is to generate these events that cause the WIP and finished goods levels to follow a certain reference trajectory. This reference trajectory should defined in such a way that it corresponds to the desired behavior of the system.

Continuous inputs for the manufacturing system

Section 5.1 described a sequence of steps to design a feedback controlled system: definition of the system, construction of a continuous dynamic model and design of the controller. In this context, the dynamic model is a set of difference equations describing the relation between the system's input and output variables. In this section, the input signal of the manufacturing system is defined as the free events. This signal, a sequence of discrete events, can not be used as a variable of a difference equation. The system of which the dynamic model is to be made (and to which the controller is to be connected) needs to have a continuous variable as input instead of a sequence of discrete events. For that purpose, the discrete event model of the manufacturing system needs to be augmented with a conversion algorithm. Purpose of this conversion algorithm is to convert a continuous input signal into discrete events. A dynamic model can be made of this augmented system (consisting of manufacturing system and conversion algorithm) when the relation between input and output signal can be described using difference equations. The controller can be interconnected to the conversion algorithm, because the continuous controller output signal is compatible to the continuous input signal of the conversion algorithm. The conversion algorithm uses the controller output to generate appropriate events. Figure 5.3 represents the controller and the augmented system of manufacturing system and conversion algorithm.

The next chapter describes a conversion algorithm, suitable to convert a continuous signal into free events for the manufacturing system. Chapter 7 describes the construction of a continuous dynamic model of the augmented system. In Chapter 8, a method of feedback control, Model Predictive Control (MPC), is introduced. A controller is designed using the model derived in Chapter 7. In Chapter 9 the designed controller is applied to the discrete event model of the multi-product flow line.

Chapter 6

Conversion of signals

As described in Chapter 5, the input signal for the discrete event manufacturing system consists of a sequence of controlled events. The chosen approach to design a feedback controller (Section 5.1) includes the construction of a continuous dynamic model of the system to be controlled. Such a model consists of difference equations describing the response of a continuous output variable to a continuous input signal. A sequence of controlled discrete events is not a continuous signal. A conversion algorithm needs to be defined in order to convert the continuous system input to a sequence discrete events, which can be offered to the manufacturing system. This conversion is not only required for constructing a continuous dynamic model, but also for applying the continuous controller to the manufacturing system. The controller provides a continuous output signal that also needs to be transformed into discrete events before it can be applied to the manufacturing system.

This chapter describes the conditions that the conversion algorithm should obey and a concept for the conversion algorithm, obeying these conditions. After that, a possible implementation of the algorithm is presented. At last, the performance of the algorithm is tested experimentally.

6.1 Conditions for the conversion algorithm

This chapter describes a conversion algorithm to convert a continuous input variable into a sequence of discrete events. In this section, conditions the conversion algorithm should obey are described.

First, the input variable of the conversion algorithm should be defined as a continuous variable. Both constructing a continuous dynamic model and connecting the system to the continuous controller output require a continuous input signal for the conversion algorithm.

Second, the continuous input variable should preferably be defined such that a linear relation exists between the input of the conversion algorithm and the output of the manufacturing system. Linear models may be used to describe such a linear relation between input and output, only causing relatively uncomplicated calculations, both for construction of the model and for design of the controller.

6.2 A concept for the continuous input variable

The previous section describes the conditions for a conversion algorithm. In this section, a proposal for a continuous input variable for the conversion algorithm is made. The conversion algorithm itself is only described in Section 6.3

Assume that a target throughput per machine per product is defined as the input for the conversion algorithm, and assume that a conversion algorithm can be defined that adequately transforms the continuous input signal into a sequence of discrete events. The property throughput is an average, and therefore specified for a time interval (see Section 2.1. Define this time interval as the controller sample time, and denote the target throughput for product i and machine j over a sample interval as δ^*_{ij}. A target throughput δ^*_{ij} is defined for each machine and each product. For the case described in Chapter 3, this implies a number of $3 \times 8 = 24$ input variables.

In Section 2.2, the generator process was introduced, responsible for the release of new lots into the system. The generator process may release new lots either in a controlled way (on demand), or in an uncontrolled way (for example with fixed or stochastic inter release times). In case the release of lots is assumed to be controllable, an input signal can be defined for the generator process. The input signal may be defined in an equal way as for the machines: δ^*_{iG} is the target throughput of product i for the generator process. For modelling purposes, the generator process may be considered as a machine process with zero service time and an infinite inventory of lots upstream. For the feedback controlled flow line, the generator process is assumed to be controlled. Therefore, corresponding to the number of product types for the generator process, 8 input variables are added to the 24 input variables for the machines. This implies a total number of input variables of 32 for the observed multi-product flow line.

Recall that the output of the manufacturing system is defined as the buffer levels per buffer per product and the finished good levels per product. The response of these outputs to input δ^*_{ij} is expected to be linear: for example, the rate of change of buffer contents is proportional to the throughput of the adjacent machines. Furthermore, the response on two excitations is expected to equal the sum of the two individual responses (superposition).

Note that a linear response is only expected for a bounded set of input signals. Buffer levels cannot become negative and machine capacity is bounded by a utilization constraint (see Section 2.4). The boundaries on the input signal are discussed in more detail in Chapter 7.

6.3 Implementation of the conversion algorithm

The previous section introduced an input variable for a conversion algorithm. A conversion algorithm to transform this input variable into discrete events obeys the conditions described in Section 6.1. This section describes an algorithm to adequately convert the input variable (target throughput δ^*_{ij}) into a sequence of discrete events for the manufacturing system. This may be represented as a control problem within the larger control problem, as represented by Figure 6.1: the conversion algorithm is the controller for the manufacturing system. Denote the input of the manufacturing system as u_m and define an alternative manufacturing system output $y_m = \delta_{ij}$. Note that this output is not the system output y_s. Purpose of the conversion algorithm may then be defined as to generate events u_m in order to let

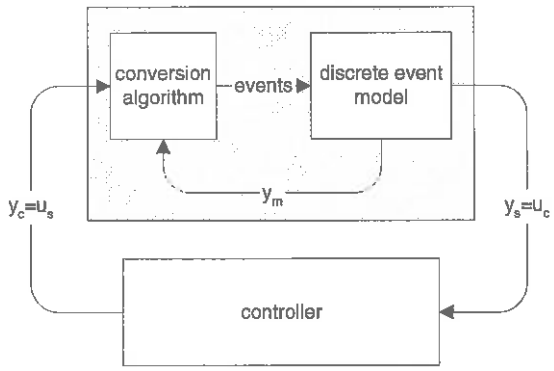

Figure 6.1: Conversion as control within the controlled system

the alternative manufacturing system output y_m follow reference trajectory u_s. A target throughput δ^*_{ij} for each sample interval corresponds to a target production of $\int_0^t \delta^*_{ij} dt$ in the interval of time from 0 to t. Define the actual production p_{ij} as the sum of lots of product type i processed on machine j. An error ϵ_{ij} may be defined, denoting the shortage of processed lots of type i on machine j:

$$\epsilon_{ij} = \int_0^t \delta^*_{ij} dt - p_{ij}. \tag{6.1}$$

Note that this error does not equal the deviation of the system output variables y_s (buffer levels and finished goods) from the reference trajectory for these variables. Introduce a production threshold value $\epsilon_{\text{threshold}}$. When the error is smaller than $\epsilon_{\text{threshold}}$, the demand is considered satisfied, so that no further lots need to be processed. To determine whether a machine should start processing a product, and if so, which type of product, the series of questions are answered for each machine:

- Which set of product types satisfies $\epsilon_{ij} > \epsilon_{\text{threshold}}$?
- Which set of product types are available in the upstream buffer?
- Take the section of these both sets. In case this section is empty, no product can or should be processed on that machine. In case the section contains one or more product types, the type with the largest error ϵ_{ij} is selected to be processed.

The state of each machine may be represented as in Figure 6.2. A machine has three states: evaluating, processing and idle. From the state evaluating, it can only move to either processing or idle, and from these two states back to evaluating. In the state evaluating, the described procedure is used to determine whether or not the machine should start processing another lot. In case the machine starts processing, the state becomes processing. The machine remains in that state until processing has finished. After that, it returns to the state evaluating. In case the machine should not, or can not, process a next lot, the state becomes idle. This occurs when no lots are available in the upstream buffer of the product types with $\epsilon_{ij} > \epsilon_{\text{threshold}}$. The machine can only leave that state and return to evaluating when either the buffer contents have changed, or at sample instants, when a new control action is computed. Sample instants are discussed in the next subsection.

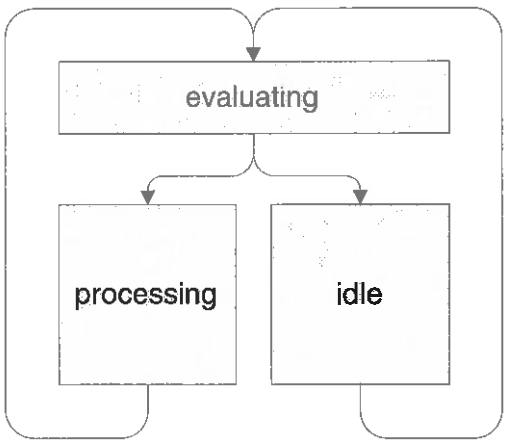

Figure 6.2: The three states of a machine

The error ϵ_{ij} is updated after a lot has been processed (the number 1 is subtracted from the corresponding error ϵ) and after a control action is received from the controller at sample times. The buffer level of workstation j increases when workstation $j-1$ has finished processing a lot and decreases when workstation j starts processing a lot. The sequence of updating error ϵ is described in the next subsection.

Sample instants

As described in Chapter 5, a discrete time feedback controller samples at discrete times. After each sample interval, the outputs are measured, a control action is computed and the control action is applied to the system. For that reason, after each sample period, the output values of the discrete event system (buffer levels and cumulative production) are sent to the controller. The controller uses these values to determine new controller output values, which are used as input signal for the conversion algorithm. The length of sample time t_s is chosen such that a number of discrete events take place within one sample time.

At the end of a sampling interval, the backlog error ϵ in generally is not exactly zero. These errors may be small (when the backlog error is lower than the production threshold value) or larger (in case the system was not able to meet the demanded production). In both cases, the error should be compensated during the next intervals. The error may be 100% compensated, or for only a smaller part. Compensating only a smaller part may be beneficial in case a very high input signal has been applied to the system for a longer period of time. In case of 100% compensation, the conversion algorithm might remain authorizing lots at a high rate because of a supposed shortage from the past high input signals, while the controller may already have decreased the output signal. For filtering the past errors, a filter is designed. After each sampling period, the remaining error is sent to the filter, and, depending on the properties of the filter, part of the error is sent back to the conversion mechanism to compensate. The design of the filter determines to which degree the error is compensated in next sampling periods. This subject is discussed in the next section.

The system is controlled in a discrete time. A first order approach is used to reconstruct the controller output signal between the sample points. Let $\delta_{i,j}(k)$ denote the controller

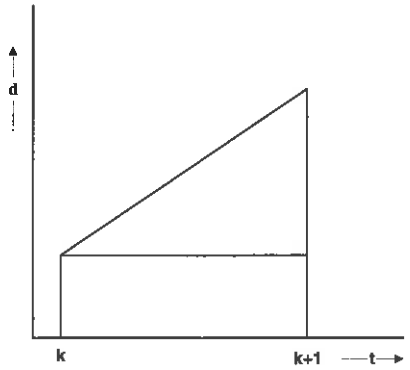

Figure 6.3: The controller output signal

output at time step k. At the beginning of a sample period, the backlog error ϵ is increased by $t_s \cdot \delta_{i,j}(k)$. At the beginning of the next sample period, when $\delta_{i,j}(k+1)$ is available, the first order approximation correction is made: $t_s/2 \cdot (\delta_{i,j}(k+1) - \delta_{i,j}(k))$ is added to ϵ too. A graphical representation of this is presented in Figure 6.3.

The next subsection is dedicated to the filter used filtering the remaining errors after each sample interval.

Filter

An algorithm is described to update the error ϵ at each sample moment. The algorithm contains a filter, that determines how errors from previous sample intervals are compensated in the current (and future) sample intervals. Errors remaining from further in the past should not be fully compensated, because the relevance of past input values is far lower than the relevance of recent input values. The total system of manufacturing system, controller and conversion algorithm would even function without compensation of errors from previous sample intervals by the conversion algorithm. In that case, an uncompensated error from the conversion algorithm causes a deviation of the system outputs from the reference output. The controller compensates this deviation by demanding higher controller output values.

Because of the low relevance of errors from far past sample times, an exponentially decaying compensation of past errors is chosen. Let the unfiltered backlog error at the end of a sample period be denoted as ϵ_u, and the filtered backlog error as $\epsilon_f(k)$. The filter is implemented as:

$$\epsilon_f(k) = \alpha \cdot \epsilon_u(k) \quad (0 < \alpha < 1). \tag{6.2}$$

Note that for $\alpha = 0$ no error from previous filter periods is compensated, and for $\alpha = 1$ the error is fully compensated.

In case an error ϵ remains after sample time k, and no events take place reducing the error and no inputs are offered making the error increase, the error decays exponentially:

$$\epsilon(k+n) = \epsilon(k) \cdot \alpha^n. \tag{6.3}$$

The output values of the controller may be saturated. In case of saturation, the variable is subject to lower and/or upper bounds. Assume the controller output signal is subject to

a saturation upper limit $u_{\max}$ and the filtering factor is set to α (with $0 < \alpha < 1$). Then, error ϵ also has an upper bound. This upper bound is derived in this subsection. Assume that for a long period of time the controller output remains at the saturation upper limit $u_{\max}$. t_s denotes the sample time. Backlog error ϵ can be described as in the next equations:

$$e(k+1) = \alpha\ e(k) + t_s u_{\max}, \tag{6.4}$$

$$\begin{aligned} e(k+2) &= \alpha\ e(k+1) + t_s u_{\max}, && \text{(6.5a)} \\ &= \alpha(\alpha\ e(k) + t_s u_{\max}) + t_s\ u_{\max}. && \text{(6.5b)} \end{aligned}$$

Therefore,

$$e(k+n) = \alpha^n\ e(k) + \sum_{i=0}^{n-1} t_s\ u_{\max}\ \alpha^i. \tag{6.6}$$

Because $0 < \alpha < 1$, the first term becomes zero as n approaches infinity. For the second term, use the relation:

$$\sum_{i=n}^{m+n} b\ r^i = b\ r^n \frac{1 - r^{m+1}}{1 - r}. \tag{6.7}$$

(6.6) and (6.7) can be used for determining the limit for backlog error ϵ after a period of maximum controller output signal:

$$\lim_{n \to \infty} e(k+n) = \frac{t_s\ u_{\max}}{1 - \alpha}. \tag{6.8}$$

An implementation of the conversion algorithm defined in this chapter has been made in the specification language χ. In the implementation, the conversion algorithm is integrated in the controller process of the discrete event model of the manufacturing system. For the implementation code is referred to the Appendix B.

6.4 Performance of the conversion algorithm

The purpose of the conversion algorithm is to convert a continuous input signal to a series of discrete events for the manufacturing system. In the previous sections, the conversion algorithm is designed in such a way that it attempts to make the throughput of the manufacturing system follow the target throughput provided by the controller. This may be regarded as a control problem as well. Different choices had been possible as well for the implementation of the conversion algorithm. This section, the behavior of the conversion algorithm in an open-loop situation is observed. Input signals are applied to the conversion algorithm to observe the response of the conversion algorithm and the manufacturing system. In the next chapter, Chapter 7, a continuous dynamic model is made of the system consisting of both the conversion algorithm and the discrete event model of the manufacturing system.

Figure 6.4 gives an example of the behavior of backlog error ϵ as a function of time. The values in the right corner are the controller output values δ. The sample time is 10 [units time]. At each sample instant, the backlog error is filtered and the filtered error is transferred to the next sample interval. This corresponds to the vertical lines downwards in

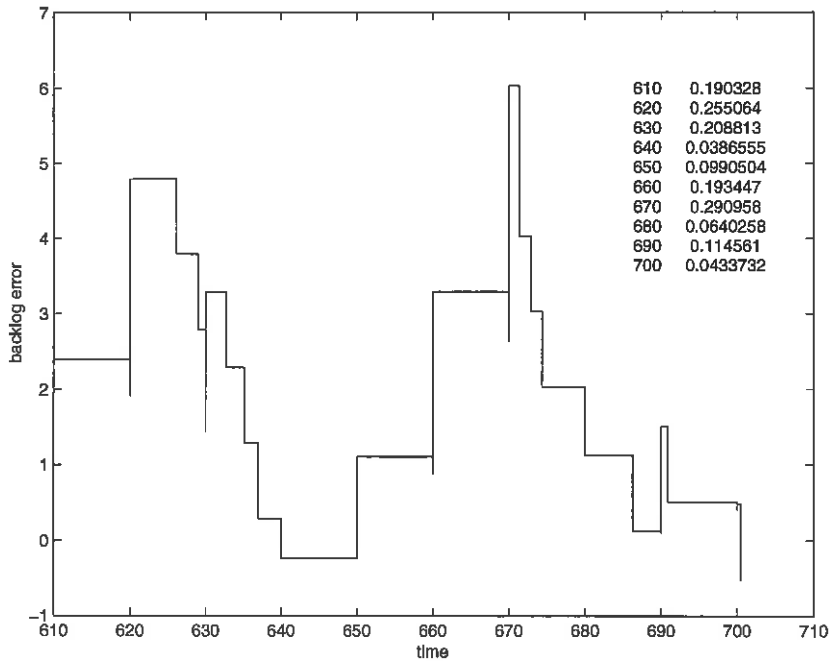

Figure 6.4: Example of conversion backlog error ϵ

the figure at each multiple of the sample time. Immediately after that, the new controller output is measured and the required production for the next sample period and the first order approximation are added to the filtered backlog error. When a lot is finished, the corresponding backlog error is reduced by one. This corresponds to the steps down with a height of one during the sample intervals. Figure 6.5 shows both the conversion algorithm input and the discrete event model throughput per sample interval for a certain machine and product type. Due to the discrete number of events, the mean throughput varies around the controller signal. Figure 6.6 shows that the cumulative production of the discrete event model and the cumulative conversion input signal. As expected, the cumulative production follows the conversion algorithm input signal. Note that an offset between the lines in Figure 6.6 is not a matter of concern. In a closed loop situation, an offset of the system output from the reference signal woud be compensated by the controller.

The sequence of steps to design a feedback controller included the construction of a continuous dynamic model of the phenomenon of interest. In this chapter, a conversion algorithm has been described in order to convert a continuous system input into discrete events, the required input for the manufacturing system. A continuous dynamic model can be constructed of the system consisting of both the conversion algorithm and the model of the manufacturing system. Developing this model is described in Chapter 7. The model may be used to design a suitable feedback controller (Chapter 8). After that, the feedback controller may be applied to the system of conversion algorithm and discrete event model (in Chapter 9).

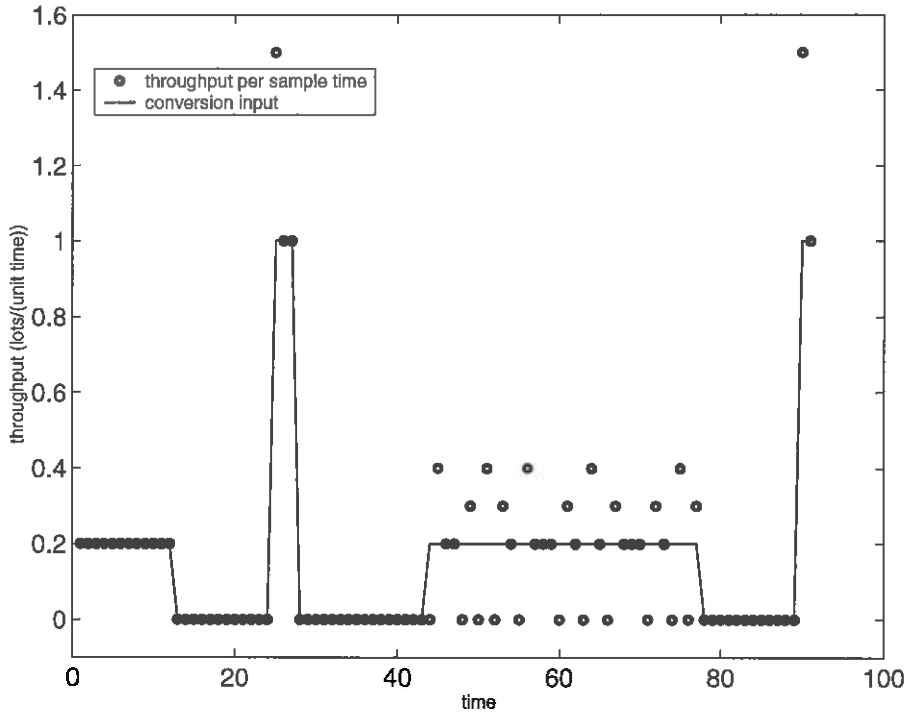

Figure 6.5: Example of conversion input and response (per sample)

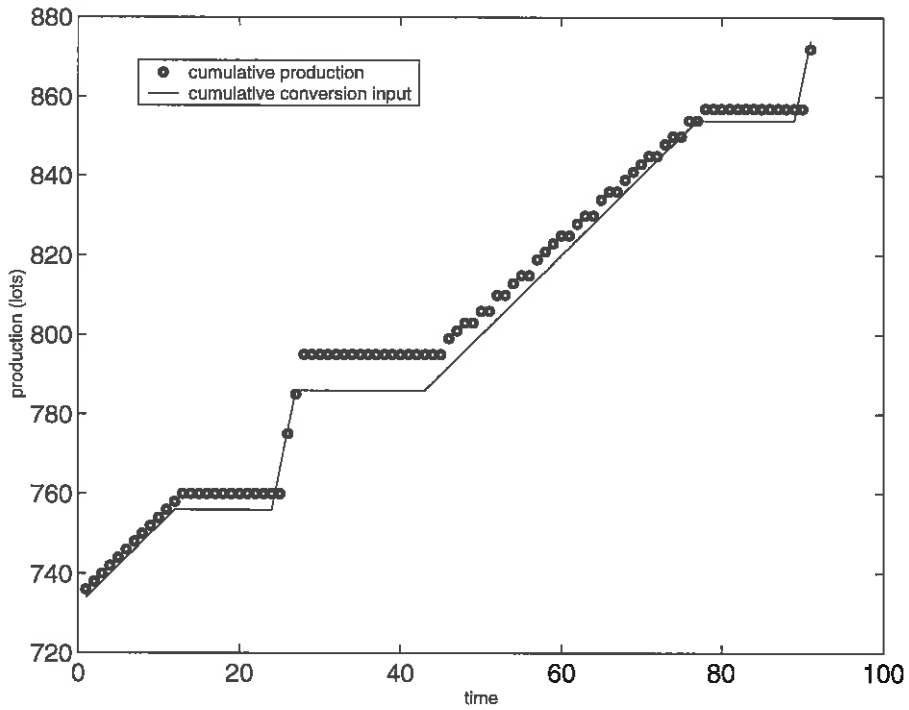

Figure 6.6: Example of conversion input and response (cumulative)

Chapter 7

The continuous dynamic model

Chapter 5 described a sequence of steps for designing a feedback controller. One of these steps was the construction of a continuous dynamic model of the system of interest. The continuous dynamic model is a set of difference equations describing the behavior of the system. Purpose of the model is to use it for designing a feedback controller.

This chapter describes the method of System Identification for obtaining a dynamic model. After that, the technique of System Identification is used to obtain the dynamic model for the case system.

7.1 System Identification

This chapter describes System Identification. System Identification is a method to obtain a model of a dynamic system. The method is an alternative for first principle design. First principle design uses the knowledge of the physical causes of the behavior of a system to create a model. The alternative, system identification, treats the system as a black box: an input signal is offered to the system and the system response is measured. System identification fits a model to these experimental data, not considering the physical causes of the behavior.

The dynamic model is to be used for designing a feedback controller for the modelled system. The performance of the controller strongly depends on the correctness of the model. For a relatively simple case, constructing a model using a first-principles is possible. The decision to use the method of System Identification is made because the controller design method should not only work for the simple case observed for this research project. It must also be possible to obtain a dynamic model for more complicated cases, maybe even too complicated to be modelled using first principles. System identification allows to construct a model for the system of both manufacturing line and the conversion algorithm described in Chapter 6 even for more complicated conversion algorithms or manufacturing systems.

The cycle of generating an identified model consists of the following steps:

- Specification of the phenomenon of interest
- Specification of model class

- Search of relevant training data
- Estimation of a model
- Evaluation of the model

After the phenomenon of interest has been specified, a model class, experimental data, and a criterion have to be specified. The experimental data consist of an input signal for the identified system, and the response of the system to this input signal. The system identification algorithms fit the parameters of the model within the selected model class, using the experimental data.. Depending on the results of the evaluation step, the model may be accepted or rejected.

State Space models

The reason for developing a dynamical model in this research project is to use it to design a feedback controller using Model Predictive Control (MPC). Model Predictive Control (MPC) is an advanced control technique, using an internal model of the system to predict the behavior of the system over a certain prediction horizon. MPC is described in more detail in Chapter 8. Most MPC algorithms require a linear, time-invariant (LTI), discrete-time, state-space model of the system. For that reason, this is the only model class considered in this report. The general form for an LTI State-Space model is:

$$x(k+1) = Ax(k) + Bu(k) + Ke(k) \tag{7.1}$$

$$y(k) = Cx(k) + Du(k) + e(k). \tag{7.2}$$

Vector $y \in \mathbb{R}^{n_y}$ is a vector of output variables and vector $u \in \mathbb{R}^{n_u}$ is a vector of input variables. Parameter k is an index for the time step. In the closed-loop system at each time step k, the values of output y are measured. These values are used to determine the appropriate controller action u to apply to the system. Vector $x \in \mathbb{R}^{n_x}$ contains the state variables. The state variables are additional variables, used to describe the relation between the input and the output. Equation 7.2 shows that the state on time $x(k+1)$ is a linear combination of the state $x(k)$ and the input signal on time k. The output $y(k)$ is a linear combination of the state $x(k)$ and the input $u(k)$. The state variables may or may not have a physical meaning. Vector $e \in \mathbb{R}^{n_y}$ is a vector containing white noise with mean 0 and variance 1.

Estimation

In this chapter, system identification is introduced as a method to obtain a state space model for a stochastic input-output system. For the estimation of the models, the MATLAB® System Identification Toolbox has been used. This subsection briefly summarizes the estimation method used by the MATLAB® toolbox.

The approach to obtain a state-space model for a stochastic input-output system is to first regress the outputs on the inputs, in order to remove the effects of the inputs. The resulting data form a state-output relation, from which the model parameters A and C can be estimated. After this, the input-state and input-output parameters B and D can be estimated, using the original data. At last, the noise parameters can be estimated using the residuals. For more detailed information about the mathematical procedures is referred to [Hei97].

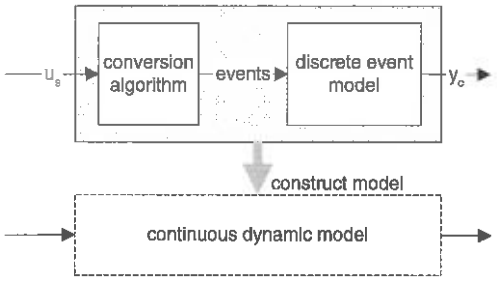

Figure 7.1: The system of conversion algorithm and manufacturing system

7.2 Identification of the three machine flow line

In this section, a model for the three machine flow line is identified, following the procedure described in the previous section.

Specification of the phenomenon of interest

In Chapter 5, inputs and outputs are selected for the discrete event model of the multi-product flow line described in Chapter 3. The discrete event system requires discrete events as input signal. To be able to feed that system with the continuous controller output signal, and to be able to describe the system as a continuous (discrete time) model, a conversion algorithm has been described in Chapter 6. The phenomenon of interest, or the system to be identified, is defined as the discrete event system together with the conversion algorithm for converting the continuous signal to discrete events. This system is represented in Figure 7.1. In the following subsections, a model of this system is made using system identification.

The input variables of the conversion algorithm and the output variable of the discrete event system have so far been described using a notation with two indices. For example, u_{ij} is the input variable corresponding to product i and resource j. The resources are the generator process and the machines. The output corresponding to product i in buffer j is denoted as y_{ij}. The level of finished goods is considered as the last buffer. The notation corresponds to a matrix representation of the variables, with the first index representing the row number and the second index the column number. A state space representation of the system requires a vector notation of input and output variables instead of a matrix notation. The vector representation is found by stacking the columns of the matrix representations. The vector notation of the input signal starts with the input variables for all product types for the generator, then all inputs for the first machine, and so on. The vector notation of the output signal starts with the output variables for all product types of the variables corresponding to the first buffer, then all output variables for the second buffer, and so on. An example of the vector representation for $i \in 0, 1$ and $j \in 0, 1, 2$ is given in the following example:

$$\text{matrix notation} \begin{bmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \end{bmatrix} \tag{7.3}$$

$$\text{corresponds to vector notation} \begin{bmatrix} u_{00} & u_{10} & u_{01} & u_{11} & u_{02} & u_{12} \end{bmatrix}^T. \tag{7.4}$$

Specification of model class

As described in the previous section, the model class best suited for designing a MPC controller is a discrete time state space model. This subsection describes the selection of the order of the state space model.

In Chapter 6, the number of input variables of the system has been defined as 32: one input per product type, for three machines and one generator process. The number of product types is 8. Chapter 5 defined the output variables of the manufacturing system as the buffer levels per product type for all three buffers and the finished goods per product type. This implies a number of output variables of 32. Before a state space model of the system can be constructed, the order of the model has to be chosen. The order of the model is the number of state variables in vector x. This number can be freely chosen, but a low order model may not be able to represent the behavior of the system correctly, while a high order model may be unnecessarily complicated, causing expensive calculations. Furthermore, a too high order model does not only describe the behavior of the system, but also the measurement noise. The knowledge of the manufacturing system and the relation between input and output signals may be useful for selecting an adequate model order. Chapters 5 and 6 and Subsection 7.2 describe that the system to be identified consists of both the flow line and the conversion algorithms. The system inputs are the target throughputs and the system outputs are the buffer levels, both per product per buffer or machine. Although the state variables in the identified state representation do not have to correspond to a physical property of the system, it is clear that the relation between input and output cannot be described without having information about previous buffer levels. This implies that the number of state variables should at least equal the number of outputs.

Specification of training data

System Identification is a method to obtain a dynamic model for a system by examining input and output data and fitting a model best corresponding to these data. For that reason, the selection of training data is essential for obtaining a correct model. In the previous subsection, a linear state-space model is selected to describe the behavior of the system. Non-linear behavior of the system cannot be described correctly by this type of model. For that reason, the input signals should be selected in such a way that an approximately linear response is expected. The input signals should satisfy two conditions:

The utilization constraint.

> In case the input signal imposes a higher throughput than the theoretical capacity, the machine is not be able to process as much as is demanded. The corresponding output buffer levels do not react in a linear way to such an input signal.

The cumulative production constraint.

> When inputs demand the system to process a product that is not available in the previous buffer, the machine is not able to process the product. In that case, the corresponding output buffer levels do not react linearly to the imposed input signal. To obtain a linear response, the cumulative demanded throughput for a machine is not allowed to exceed the cumulative demanded throughput of the previous machine (or generator).

In this report, an input signal meeting these two constraints is called a *valid* input signal. Note that even for a valid input signal, non-linear response may occur because of the stochastic properties of the machine service times. Though theoretically valid, an input signal demanding a high throughput may not be realized in case of occasional longer service times.

A series of input data is generated, exciting all inputs. The value of the input signals vary from zero to the maximum throughputs allowed by the utilization constraint. Different inputs are excited in combination to show the dependency between different inputs would show. The size of the experiment is about 6000 data points. The input signal is applied to the system of conversion algorithm and discrete event model. The sample time t_s is 10 units time. During this experiment, the output variables y_s are measured. Input and output data form a data set of 6000 input and output points. Each input point consists of a value for each of the 32 inputs. Each output point contains the values of the 32 measured output variables. The data set can be used to estimate the parameters of a model using system identification. Note that the simulation experiment to gather data points is an open-loop simulation: the input trajectory is selected prior to the experiment, and does not depend on the measurements of the system outputs.

Estimation of the model

This subsection describes the estimation of a state space model for the three machine multi-product flow line. The MATLAB® System Identification Toolbox estimates the parameters for a model, using the set of training data described in the previous subsection.

The toolbox offers the possibilities to load and pre-process data, and to estimate a (state-space) model for a given system order. The model may be pre-structured, or entirely free. In case of a structured model, some or all elements of some of the matrices of the model are given so that the identification algorithm only fits the missing parameters. For a free model, the identification algorithm determines values for all model matrices. The obtained models can be analyzed using (other) verification data.

Recall that the number of product types is set to 8. A lot generator process and three machines are controlled. For that purpose, the contents of three buffers and the finished goods levels are measured. Because of that, the system has $8 \cdot (1 + 3) = 32$ input variables and $8 \cdot (3+1) = 32$ output variables. Section 7.2 described that the number of state variables should be at least 32 as well.

Sequentially, three approaches have been used to estimate a model using the system identification. These three attempts are successively described in this subsection.

- The first approach was to estimate all parameters of the complete system For a number of 32 input, output and state variables, matrices A, B, C, D and K of (7.1) and (7.2) all have the size 32×32. To estimate these matrices, $5 \cdot 32^2 = 5120$ parameters have to be estimated. The MATLAB® System Identification Toolbox suffered from insufficient memory attempting to estimate all 5120 parameters, using the 6000 data points consisting of $2 \cdot 32$ values each. Because of that, a smaller problem was offered to the toolbox.

- Target of the smaller problem was to estimate a model for the same system, but only two of the eight product types were observed. The relation between input and output

is expected to be equal for all product types. The interrelations between product types can be estimated by identifying the system for two product types. As long as the input signal is valid, the input for the one product type is expected to have only little effect on the outputs of the other product types. The same experimental data was used, but only the data of the two selected product types were considered. With only two product types observed, the system only requires $2 \cdot 4 = 8$ inputs, outputs and state variables. This implies a number of only 320 parameters to be estimated. The MATLAB® System Identification Toolbox succeeded in estimating the model.

The identified model showed a clearly recognizable structure. Matrix A almost equals the unity matrix. This implies (see (7.1)) that state variables do not depend on other state variables, but only on their previous value and the input signal. In case no input signal is applied, the state remains at its previous value. Matrix D almost equals the zero matrix. This is caused by the fact that no direct feed-through of the input to the output occurs. It takes at least one sample time for an input signal to influence the output signal. The structure of B and C were less obvious. Not B and C separately, but the product CB showed a recognizable structure. The product CB is (rounded) represented in (7.5):

$$CB = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & -10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}. \tag{7.5}$$

The values in product CB correspond to the change in buffer contents when a unity input signal is applied to the corresponding input during a sample time of 10 time units. The exact value of the non-zero elements is slightly smaller than 10, because a theoretically valid signal may not be realized due to stochastic service times, as described in Section 7.2. The state variables do not have a clear physical representation. The structure of product CB may be represented by the following equation:

$$CB_{ij} = \begin{cases} 10 & \text{for } i = j, \\ -10 & \text{for } i = j + N, \\ 0 & \text{for all other elements.} \end{cases} \tag{7.6}$$

with N representing the number of product types.

Identification of the system, only considering two out of eight product types, yielded a model for the relation between the inputs and outputs for the two product types. To design a feedback controller for the manufacturing system, a model for the system including all product types is required. The structure of the obtained model may be used to construct models for the system for more product types.

- The knowledge of the structure of the smaller model may be used for identifying a larger model. The MATLAB® System Identification Toolbox is able to estimate the parameters of a model with a given structure. In a structured model, distinction can be made between free and fixed parameters and free parameters of the model. The fixed parameters are fixed at a certain value by the user. The free parameters are

estimated by the identification toolbox. A structured model may be offered to the toolbox. For a structured model, only the values for the free parameters are estimated instead of estimating the values for all elements of the matrices A, B, C, D and K.

Define the state variables as the buffer contents and the finished goods levels. In that case, the state corresponds to the outputs, implying C is a unity matrix. Note that this definition has the advantage that the state can directly be measured. The relation between the inputs and the outputs should be equal as the relation described by the product CB of the previous identified model. A structured state-space model for 32 inputs, outputs and state variables may be offered to the toolbox. It has the following structure:

Matrix A is a unity matrix of size 32×32. Matrix D is a zero matrix of 32×32. Matrix C is a unity matrix of size 32×32. Because of that, the structure of matrix B is equal to the structure for the product CB of the previous identified model. Therefore, the model is represented by:

$$A = I, \tag{7.7}$$

$$B_{ij} = \begin{cases} b & \text{for } i = j, \\ -b & \text{for } i = j + N, \\ 0 & \text{for all other elements,} \end{cases} \tag{7.8}$$

$$C = I, \tag{7.9}$$

$$D = 0. \tag{7.10}$$

The number of product types N is 8, and the size of the matrix is 32×32. All zeros are fixed, and all elements denotes as b or $-b$ are free. This implies a number of free variables of $7N = 56$ free variables. Disturbance matrix K may be fixed at zero, or estimated, implying $32^2 = 1024$ extra variables. The values for these free parameters may be obtained for the complete system (all inputs and outputs) without a lack of memory.

The structured model with identified free parameters describes the relation between the input and output signal for all product types. The parameters can be estimated without problems due to a lack of computer memory. In the next subsection, the performance of the model is discussed.

Evaluation of the model

The free parameters of the model described in the previous subsection have been estimated by the System Identification toolbox. The estimated values for the free parameters b in (7.8) varied depending on the data set used for the estimation process. As remarked in Section 7.2, a theoretically valid signal may not be realized by the machines because of their stochastic service times. For a data set demanding a low machine utilization, the value for b almost reached the limit of 10.

The model may be graphically evaluated by plotting the output values predicted by the fitted model together with experimental data. The model can best be evaluated with other data than the data used to fit the model. An example of a plot of two input signals and the response predicted by the model and the experimental response is given in Figure 7.2.

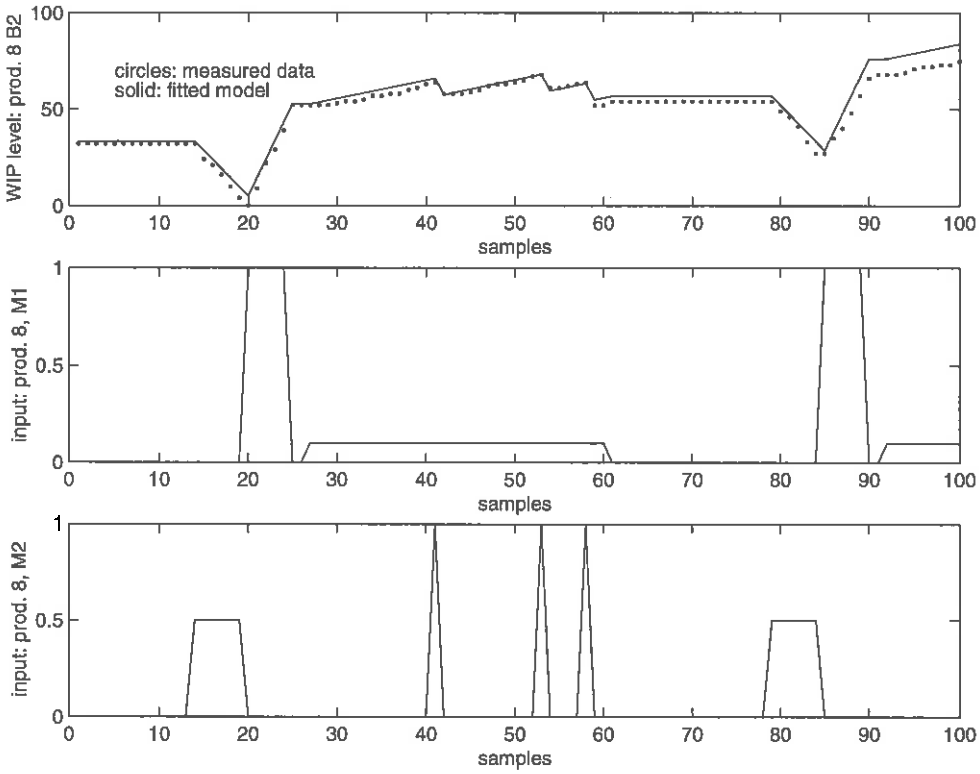

Figure 7.2: Experimental output and model output and corresponding input signals

Similar plots are made for all output signals. For all outputs, the model output behaves similar to the experimental data. The experimental data may deviate from the model output because of the stochastic behavior of the machines, as described in Section 7.2. The model contains both the information which inputs are related to which outputs, and the magnitude and direction of the influence of an input to an output. In the next chapters, the model is used to design an MPC (Model Predictive Control) feedback controller. This feedback controller contains the model to predict the behavior of the system for a certain prediction interval. Note that the time scale of Figure 7.2 (100 samples) is larger than the prediction horizons used in Chapter 9 (up to approximately 10 samples). Therefore, small deviations of the predicted output and the experimental output that only show after large time, are not of great importance.

In the next chapter, the obtained continuous model is used for designing a suitable controller. The matrices of the state-space model are described in (7.8), and the value of b in B is set to exactly 10.

Chapter 8

The MPC controller

Chapter 8 gives an introduction to Model Predictive Control (MPC) and describes how MPC may be applied to the multi-product flow line.

8.1 Model Predictive Control

Model Predictive Control (MPC) is an advanced control technique widely used in for example industrial process control. The benefits of MPC include that it can easily be applied to multi variable systems, it can deal with constraints and it is suited to control systems containing time delays ([Mac02]). Feedback control may be defined as the process of measuring output variables to use that information to cause a system variable to accurately follow a reference trajectory. Characteristic aspects of MPC are the presence of an internal model, representing the behavior of the system, and the use of a receding horizon.

As for most discrete time feedback controlled systems, at sample k, the system output $y_s(k)$ is measured. The task of the controller is to determine an appropriate controller output $y_c(k)$ to used as system input $u_s(k)$. Figure 8.1 shows the concept that the MPC controller uses to determine the controller output $y_c(k)$. At sample k, the following information is available:

- System output $y_s(k)$.
- The reference trajectory for the system output y_r^p. The number p represents the control horizon. The reference signal should be known for the next p samples:

$$y_r^p(k) = y_r(k+1), y_r(k+2), \ldots, y_r(k+p). \tag{8.1}$$

Now assume a system input trajectory $\hat{u}_s^m(k)$ for the next m steps ahead:

$$\hat{u}_s^m(k) = \hat{u}_s(k), \hat{u}_s(k+1), \ldots, \hat{u}_s(k+m). \tag{8.2}$$

The number of input moves ahead, m, is referred to as the control horizon. The assumed input trajectory $\hat{u}_s^m(k)$ and the model, representing the system's behavior, can be used to predict the future response of the system $\hat{y}_s^p(k+1)$, to the assumed input trajectory:

$$\hat{y}_s^p(k+1) = \hat{y}_s(k+1), \hat{y}_s(k+2), \ldots, \hat{y}_s(k+p). \tag{8.3}$$

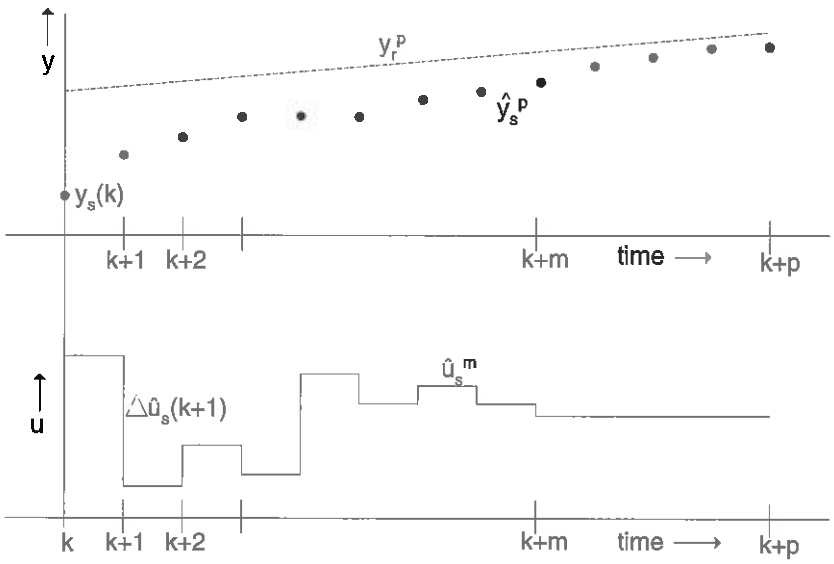

Figure 8.1: The concept of Model Predictive Control

A criterion may be defined to quantify the performance of the assumed system input trajectory. A generally used criterion quadratically penalizes errors (differences between the system output and the system output reference) and moves in system input signal:

$$\text{Minimize } J(k) = \sum_{j=1}^{p}\sum_{i=1}^{n_y}(w_y(i,j)[y_r(k+j) - \hat{y}_s(k+j)])^2 + \sum_{j=1}^{m}\sum_{i=1}^{n_u}(w_u(i,j)\Delta u_s(i,j+k-1))^2, \tag{8.4}$$

with

$$\Delta u_s(k) = u_s(k) - u_s(k-1). \tag{8.5}$$

Weighing factors $w_y(i,j)$ and $w_u(i,j)$ define the ratios between the penalties. The response of the system is predicted and errors are penalized over prediction horizon p. The value of input signals of the input trajectory is only allowed to change over control horizon m. After that, the input is assumed to remain constant. Control horizon m is required to be smaller or equal to prediction horizon p.

Of the optimal system input trajectory, defined by (8.4), only the first step is applied to the system. At the next sample instant, an optimal input trajectory is determined again. This concept is referred to as a receding horizon.

Constraints may be imposed to the system's variables. For example, the step of system input, Δu_s, may be limited, or the value of the state variables may be bounded to a certain interval. In case of constrained variables, the criterion (8.4) becomes a quadratic programming (QP) problem ([Mac02]). For each control action, the problem has to be solved again. When no constraints are applied to the variables, finding the optimal system inputs u_s in (8.4) is a least squares problem. The unconstrained MPC control law may be written as a matrix multiplication, including a fixed MPC controller gain matrix (K_{MPC}). This matrix only needs to be determined once, off-line. Solving a least-squares problem once off-line is computationally less complicated than solving a QP problem for each control action. For that reason, unconstrained MPC is used to control the multi-product flow line in this

research project. In case the problem proves to require constraints, one may switch over to constrained MPC.

In this section, the general concept of MPC is described. The next two sections describe two implementations to design an MPC controller. The first implementation uses the MATLAB® MPC toolbox. Its use is limited to constant reference signals. Because of that, an alternative, more general implementation for the design of an MPC controller for the multi-product flow line is described in Section 8.3

8.2 Matlab MPC Toolbox

MATLAB® is equipped with an MPC Toolbox. The MPC Toolbox contains tools and functions to design and evaluate MPC controllers. Simulations with the MPC toolbox are limited to simulations with the MPC controller applied to a continuous model in the MATLAB® workspace. The purpose of this research project is to apply the feedback controller to a discrete event model of a manufacturing line. Because of that, the MATLAB® MPC Toolbox is only used to determine the controller gain matrices. The determined matrices are used in MATLAB®, but outside the MPC Toolbox.

The function smpccon

The function of the MPC Toolbox to design MPC controller gain matrices is named SMPCCON. It uses the objective defined in (8.4) to determine controller gain matrices. Arguments of the function are a continuous dynamic model, a prediction horizon, a control horizon and weighing factors for the output tracking error and the input steps. The continuous model is used to predict the response of the system to an assumed input trajectory. The other arguments correspond to the variables in (8.4). The matrix K_s, returned by the function SMPCCON, is consists of three matrices:

$$K_s = [K_{\xi 1}\ K_{\xi 2}\ K_r]. \tag{8.6}$$

Matrix k_r equals $-k_{\xi 2}$. Because of that, only one of these matrices is used in the control law. The closed loop control law is the following:

$$\Delta u_c(k) = y_c(k) = k_{\xi 1}\Delta x(k) + k_{\xi 2}(y_s(k) - y_r(k)), \tag{8.7}$$

with $\Delta x(k)$ denoting $x(k) - x(k-1)$, the deviation of the state variables. In general, the state of a system cannot be measured directly. When the state cannot be measured directly, it can be reconstructed by a state observer, using the input and output signals and a model of the system. A simple state observer is described in Appendix A. The states of the observed system described in Chapter 5 can directly be measured, because the states are equal to the outputs. The control law provides steps of the controller output instead of controller outputs. Therefore, the input applied to the system, $u_s(k)$, is the sum of controller output y_c and previous system input $u_s(k-1)$. Vector y_r is a vector containing the reference values for the outputs. Equation (8.7) is the solution to the least squares problem of (8.4), assuming the output reference values remain at the constant value of $y_r(k)$ during the prediction horizon. For the next control action, the reference is assumed constant at $y_r(k+1)$, etcetera.

Interpretation of the controller gain matrices

The function SMPCCON may be used to determine controller matrices $k_{\xi 1}$ and $k_{\xi 2}$ to control the discrete event model. For certain parameters, the matrices have the following values:

$$k_{\xi 1} = \begin{bmatrix} 0.0845 & 0.0763 & 0.0720 & 0.0701 \\ -0.0081 & 0.0802 & 0.0744 & 0.0720 \\ -0.0043 & -0.0100 & 0.0802 & 0.0763 \\ -0.0018 & -0.0043 & -0.0081 & 0.0845 \end{bmatrix} \tag{8.8}$$

and one gain for error e:

$$k_{\xi 2} = \begin{bmatrix} -0.0641 & -0.0475 & -0.0395 & -0.0362 \\ 0.0166 & -0.0561 & -0.0442 & -0.0395 \\ 0.0079 & 0.0199 & -0.0561 & -0.0475 \\ 0.0032 & 0.0076 & 0.0166 & -0.0641 \end{bmatrix} \tag{8.9}$$

For the system of conversion algorithm and discrete event model, inputs have been defined as target throughputs (of generator and machines, per product type) and outputs buffer levels and finished goods levels (per product type). The state variables are equal to the outputs. Because of this, matrices (8.8) and (8.9) may be interpreted. Matrix (8.9) defines the step of input signal for the generator and each machine separately as a function of the system output reference tracking error $e = y_s - y_r$. Each row corresponds to a system input (generator or machine) and each column corresponds to an output (buffer or finished goods). For example, an negative error of size 1 in the first buffer (one lot less than the reference level), adds a positive step (of size 0.0641) to the input of the upstream generator, while the downstream machines are slightly slowed down. The larger the physical distance between an input and an output is, the less influence output errors have on the input.

This section describes the MPC implementation of the MATLAB® MPC Toolbox function SMPCCON. It provides optimal system input steps Δu with respect to (8.4), assuming the reference signal to remain constant during the prediction horizon. In general, a reference trajectory does not remain constant. For example, the demanded finished goods level, one of the outputs of the discrete event model, should increase in time. The next section describes a method to obtain MPC controller gain matrices without using the MATLAB® MPC Toolbox. The method can determine the optimal controller outputs for a reference signal that is not constant over the prediction horizon.

8.3 MPC implementation for a time varying reference trajectory

The previous section describes an implementation of MPC using the MATLAB® MPC Toolbox. The function SMPCCON is used to determine controller gain matrices. Multiplied by the error of the output variables and the steps of state variables, these matrices provide the solution to the least squares problem defined by (8.4). A disadvantage of the implementation is that the reference signal is assumed to remain at a constant value for the length of the control horizon. This section describes an alternative control law, not using the MATLAB® MPC Toolbox. This implementation can handle a reference that varies during the prediction horizon.

An MPC controller gain matrix K_{MPC} is determined by solving a least squares problem containing the weighing factors, control horizon and prediction horizon from (8.4) and the continuous dynamic model. For a definition of the least squares problem is referred to Appendix A. The control law becomes the following:

$$\Delta u_s^m = K_{MPC}\, e^p_{0,\Delta u=0}, \tag{8.10}$$

with Δu_s^m denoting the steps in system input during the prediction horizon, and $e^p_{0,\Delta u=0}$ denoting the error between the system output and the output reference trajectory during the prediction horizon. The error $e^p_{0,\Delta u=0}$ may be reformulated, using the continuous dynamic model to predict the response of the system:

$$e^p_{\Delta u=0} = \begin{bmatrix} -C_A\Phi_A \\ -C_A\Phi_A^2 \\ \vdots \\ -C_A\Phi_A^p \end{bmatrix} \begin{bmatrix} \Delta x \\ y \end{bmatrix} - y_r^p, \tag{8.11}$$

with r_p denoting the output reference trajectory for the prediction horizon. For the meaning and computation of the matrix containing the elements $-C_A\Phi_A^i$ is referred to Appendix A. The control law may be rewritten by substituting (8.11) in (8.10):

$$\Delta u_s^m = K_{MPC} \begin{bmatrix} -C_A\Phi_A \\ -C_A\Phi_A^2 \\ \vdots \\ -C_A\Phi_A^p \end{bmatrix} \begin{bmatrix} \Delta x \\ y \end{bmatrix} - y_r^p. \tag{8.12}$$

Appendix A is dedicated to the mathematical implementation of the MPC controller. The next chapter, Chapter 9, describes the closed loop system of discrete event model, MPC controller and conversion algorithm. Simulation experiments are used to determine the performance of the MPC controlled system.

Chapter 9

MPC applied to the multi-product flow line

This chapter first summarizes the previous chapters, in which a discrete event model, a conversion algorithm, a continuous dynamic model and an MPC controller are introduced. After that, the discrete event model (DEM), the MPC controller and the conversion algorithm are connected. Several parameters of this system are described, and a suggestion for the initial value for these parameters is given. After that, a method to find the optimal parameter values is presented. Simulation experiments are used to measure the performance of the controlled DEM. The results of these experiments may be compared to the results of the common control methods, described in Chapter 4. Similar experiments are performed for a system with limited buffer capacities.

9.1 The MPC controlled flow line

The discrete event models.

In Chapter 3, a manufacturing system is introduced. The manufacturing system is a three machine multi-product flow line. Various control methods are applied to it, to measure their Performance may be measured in many ways, but this research project focuses on the relation between throughput and WIP level. The manufacturing system may be modelled used discrete event models (DEMs). Input of the discrete event system is a sequence of authorizations, a sequence of discrete events. Outputs are defined as the buffer levels and the finished goods levels.

The conversion algorithm

The discrete event model is controlled by defining which events should take place at which moments. Because a goal of this research project is to use feedback control to control the discrete event model of the manufacturing system, the input variable of the system should be a continuous signal instead of a sequence of discrete events. For that reason, a conversion algorithm is described in Chapter 6. The algorithm converts a continuous input signal (δ^*, a target throughput per product per machine) into an appropriate sequence of events.

The continuous model

A continuous dynamic model is made of the system consisting of conversion algorithm and discrete event model. The continuous dynamic model is a set of difference equations, describing the relations between the input and output variables of the system. The model is a linear time-invariant state space model. It is derived using System Identification. The model is described in Chapter 7.

The MPC controller

The continuous model is used to design a feedback controller. The method of control is Model Predictive Control (MPC). Characteristic aspects of MPC is the presence of a receding horizon and an internal model in the controller. The internal model is used to predict the response of the system. More information about the MPC controller is found in Chapter 8.

This chapter describes the system of discrete event model, MPC controller and conversion algorithm. These three components are connected. At each sample time, the output of the discrete event model is measured. The measurements are used by the controller to compute an appropriate control action. The control action is applied to the conversion algorithm. The conversion algorithm uses the controller output to generate events for the discrete event system.

9.2 Implementation

A discrete event model of the three machine manufacturing line is made using the specification language χ ([Roo01]). The conversion mechanism and a sampling mechanism are implemented in the model's controller process. The System Identification Toolbox of MATLAB® is used to obtain the continuous dynamic model. The experimental data, used to identify the continuous model are generated by a discrete event model implemented in χ. The MPC controller, containing the dynamic model, is implemented in MATLAB®. Communication between χ and MATLAB® takes place using the pymat-module of the language Python. For implementation codes is referred to Appendix A.

The concepts of the implementation of the conversion algorithm, the continuous model and the controller are discussed in Chapters 6 to 8. The concepts of the discrete event model of the manufacturing system are similar to the discrete event models described in Chapter 4. Some differences are discussed in the next subsection.

Implementation of the discrete event model

The performance of the MPC controlled model of the three machine manufacturing line has to be compared to the performance of the common control strategies described in Chapter 4. For that reason, the MPC controlled systems and the systems using common control strategies must be modelled as equally as possible. Chapter 4 showed that for different control strategies, the demand has to modelled differently.

In all experiments, the contents of three buffers was observed. For some control methods, these three buffers were the three buffers in front of the three machines, while for other methods the contents of three buffers after the three machines were observed. Also for the

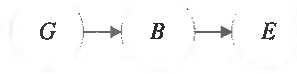

Figure 9.1: Example for demand modeling

MPC controlled line decisions have to be made which buffers to observe and which method to use for demand modeling.

The method for modeling demand for the pull and pull-conwip controlled line seems a reasonable method for modeling demand for the MPC controlled manufacturing line. Unfortunately, this method of modeling demand is not possible for the MPC controlled system. First, the method for modeling demand for pull and pull-conwip systems is summarized. After that, an explanation is given why this method cannot be used for modeling demand of the MPC controlled system.

For pull and pull conwip, demand is modelled as orders arriving at the last buffer. In case an order cannot be satisfied immediately, because the demanded product type is not available in the finished goods buffer, the order is stored in a backlog order list. The backlog order list is FIFO: the second order in the list can only be satisfied after the first one has been satisfied. The arrival rate of orders should not be set too low, because the goal of the experiments is to determine the maximum throughput that the system can reach. For that reason, the arrival rate is set higher than the maximum throughput of the system. Because of the order arrival rate higher than the throughput of the system, the length of the backlog order list increases. An ever-growing backlog order list is equivalent to generating the next order for a product type immediately after the previous order has been satisfied.

This method of modeling demand does not work for the MPC controlled system. Reason for this is that the release of lots into the system is explicitly limited by the WIP level in the system. In other cases, as the MPC controlled manufacturing line, one of the buffer levels does not reach equilibrium, but explodes. This phenomenon is explained by the following example. An example of a system with similar behavior is a system consisting of only a generator, buffer and exit process processing more than one product type. This figure is represented by Figure 9.1. Assume that the generator releases lots (with stochastic or deterministic inter release times) of the types in a certain ratio. Let the exit process immediately generate an order for a product type after the previous order has been satisfied. When the demanded product type is available in the buffer, the lot is immediately removed from the buffer and the order is satisfied. This simple system does not reach equilibrium, but the number of lots in the buffer of one of the product types continues increasing.

Because the described method used to model demand for pull and pull conwip cannot be used for the MPC controlled system, a different method has to be used. In the χ implementation of the MPC controlled discrete event model, lots are assumed to immediately leave the finished goods buffer. Because of that, the system contains less stochastic effects than the push, pull and pull-conwip models (see Chapter 4). Because lots are immediately removed from the last buffer, mean WIP level in the last buffer is zero. In order to make a more fair comparison with the other control methods observing the WIP levels in 3 buffers, the WIP level in the buffer in front of the first machine is observed as well. Figure 9.2 represents the χ implementation of the discrete event model. A circle represents a process and an arrow represents a flow of lots or information. The generator G and machines M are authorized by a controller process C, containing the conversion algorithm. Information

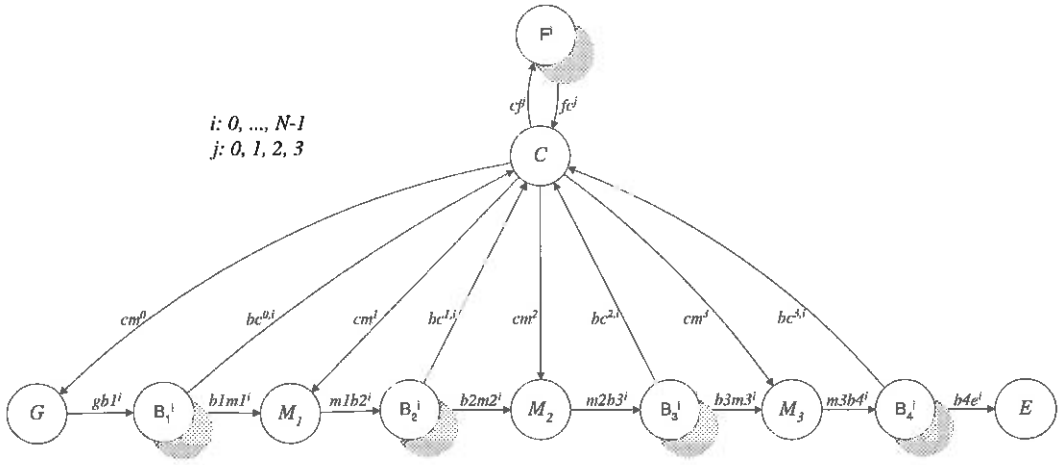

Figure 9.2: Implementation of the MPC controlled flow line

about the buffer levels is sent to the controller by the buffers B. Process F is the filter process (see Chapter 6 in the conversion algorithm.

9.3 Parameters of the MPC controlled system

In this section, the MPC controlled system is described, and simulation experiments are done, to measure the performance of the MPC controlled system. The MPC controlled system has many parameters. The value of these parameters influence the behavior and performance of the system. In this section, the system parameters are described and suggestions for their initial values are given. The parameters may be divided into parameters of the conversion algorithm, of the controller design and the reference signal.

Conversion algorithm parameters

The implementation of the conversion algorithm contains several parameters that influence the behavior of the algorithm and therefore the entire closed loop system. These parameters are:

1. the saturation limits for the controller output,
2. the production threshold value.
3. the filtering factor α at sample instants.

The values of the conversion input signal may be bounded by a lower and/or and upper bound. When the signal threatens to exceed one of the bounds, its value is kept at the boundary value. Imposing these bounds may be referred to as saturating the signal. A target of saturation is to avoid the controller to generate input signals that cannot be realized by the system. For example, a valve cannot be opened for more than 100%. Closed loop simulation experiments showed that the output signals of the controller remained relatively

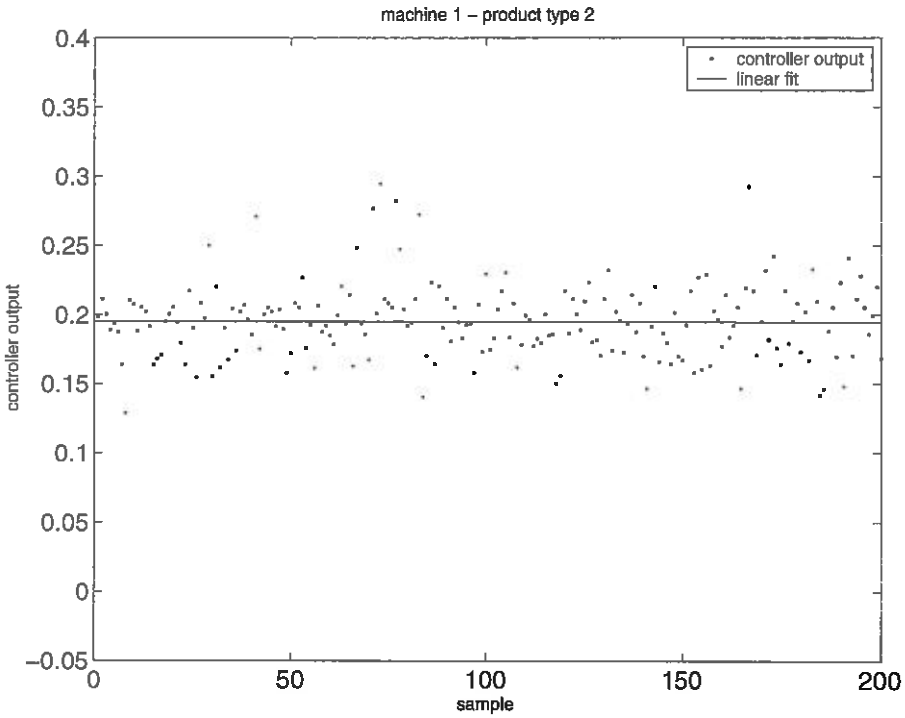

Figure 9.3: Example of controller output values

low, even for high machine utilizations. As an example, Figure 9.3 shows the controller output signal for the first machine for the product type with the largest demand, during a number of samples. The utilization of the system is 90% and product mix parameters $k_d = 0.2$ and $k_p = 5.0$. The relatively low controller output values give no reason to apply saturation.

The production threshold value $\epsilon_{\text{threshold}}$ is described in Chapter 6. If the conversion algorithm backlog error ϵ is higher than the $\epsilon_{\text{threshold}}$, a machine may be authorized to produce a lot of the corresponding product type. The value of parameter $\epsilon_{\text{threshold}}$ influences the reaction of the conversion algorithm to small inputs. For a high value of $\epsilon_{\text{threshold}}$, the conversion algorithm may not authorize a machine to process a lot, while for a low $\epsilon_{\text{threshold}}$ value the algorithm may have. Recall that conversion algorithm backlog error e does not automatically lead to system backlog error e, because the feedback controller measures and compensates the system backlog (see Chapter 6). In closed loop simulations, the effect of parameter $\epsilon_{\text{threshold}}$ is not clearly visible. Probably, a different $\epsilon_{\text{threshold}}$ value results in a slightly different values in matrix B in the continuous model (see 7) and therefore a slightly different MPC controller gain matrix.

Filtering factor α may be considered as the variable distributing the task of compensating backlog errors over the conversion algorithm and the controller. For $\alpha = 0$, all conversion backlog error ϵ is neglected. Therefore, system backlog error e occurs and this backlog has to be compensated by the controller: a lack of production causes a deviation from the reference signal and therefore a larger controller output. For $\alpha = 1$, all conversion backlog errors are compensated by the conversion algorithm. Less system backlog occurs so the controller has to compensate less. Note that the conversion backlog error ϵ does not decrease exponentially

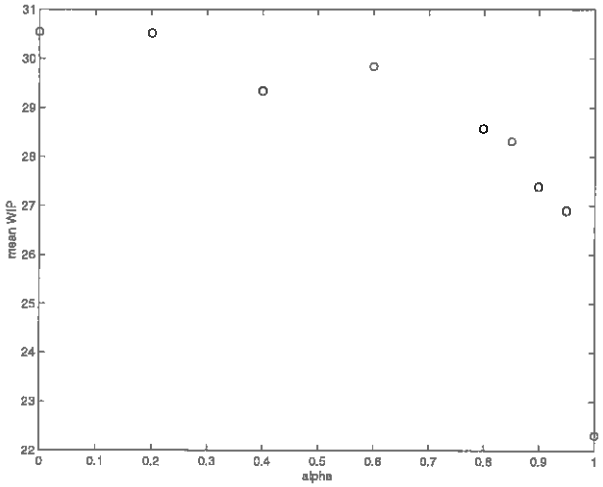

Figure 9.4: Mean WIP as a function of filtering factor α

to zero in time in case $\alpha = 1$ (see Chapter 6).

Experiments have been done, varying α from 0 to 1. Of course the exact results of the experiments are only valid at the observed throughput level and at equal settings for the other parameters of the system.Nevertheless, the experiments show that the value of α has a large influence on the performance of the system. Figure 9.4 shows the mean WIP required to reach a 90% system utilization for several levels of α. The other parameters of the experiments are: mix parameters $k_d = 0.2$, $k_p = 5.0$, reference WIP level is 0.6 lot for all products and buffers, prediction and control horizon 10 steps and ratio of weighing factors $w_u : w_y = 4$. These parameters are described in the next subsections. Figure 9.4 shows that the mean WIP level strongly decreases at higher values for α.

Controller design parameters

Several parameters have to be set when designing the MPC controller. The controller gain matrix is a function of the ratio of weighing factors w_u and w_y, penalizing controller output steps and deviation of the system output from the reference signal. Furthermore, the controller gain matrix is a function of the control horizon m and prediction horizon p.

For an initial setting of the weighing factor, estimate the order of magnitude of the two penalized aspects in (8.4): the deviation of the system output from the reference signal e, and the controller output steps. In simulation experiments, the buffer levels per product type show to be either zero or one, and rarely two or higher. Depending on the reference signal, the order of magnitude of error e for buffer levels is one. For finished good levels, the order of magnitude of the error e is about one as well. The order of magnitude of the controller output is about 0.1: assume the throughput of the system, δ, is close to the utilization limit. For mean processing times of 1 (units time), this

implies that the sum of target throughput δ^* for all products of one machine should be close to 1. Throughput about evenly spread over the different product types implies a controller output signal with order of magnitude 1/8. Assume that the steps in the controller output signal are not larger than the order of magnitude of the output signal itself. In that

case, the order of magnitude of Δu_s is about 1/8 as well. In order to penalize a normal reference tracking error e about equal as a normal step in controller output, the ratio of weighing factors w_u and w_y in (8.4) should be approximately opposite to the ratio of orders of magnitude of Δu and e. The orders of magnitude of w_u and w_y in this subsection suggest a ratio of weighing factors w_u and w_y of approximately 8 : 1. For this research project, weighing factors are chosen equally for all inputs and weighing factors are chosen equally for all outputs.

The control and prediction horizon m and prediction horizon p determine how many sample times ahead the inputs may be changed and the response is predicted (see 8.4). For this research project, the prediction horizon m is chosen to be equal to the prediction horizon to reduce the number of variables for which experiments had to be performed. The control horizon should be selected long enough that the effects of an input signal have reached the output of the system. The model of the manufacturing line does not include time delays. Nevertheless, longer horizons provide a smoother controller output signal. The performance is expected to increase for longer horizons, but the increase of performance is negligible for very long horizons. Furthermore, the MPC implementation described in Section 8.2, and used for the experiments in this chapter, uses a constant reference value during the prediction horizon. The increase of the reference level for the finished goods is ignored. For that reason, very long prediction horizons should be avoided when this implementation is used. The alternative implementation, described in Section 8.3 is able to use time-varying reference signals during the prediction horizon.

The output reference signal

Control may be defined as the process to cause a system's variable to follow a certain reference trajectory. In this research project, the output of the discrete event manufacturing system should follow a certain reference trajectory. The outputs of the discrete event system are the WIP levels per buffer and product, and the cumulative finished lots per product. The derivative of the cumulative finished lots is the throughput of the system. Therefore, imposing a reference for the output signal is the equivalent of imposing reference values on the WIP levels and the throughput of the system.

Purpose of controlling the manufacturing system is to cause the system to achieve a high throughput at a given WIP level, or, vice versa, or to cause the system to require a low WIP level for a given throughput. To reach this goal, selection of an appropriate reference signal is essential. The system cannot follow each reference signal. The throughput is, for example, bounded by a utilization constraint (2.4). Furthermore, variability causes queueing. The higher the utilization is, the more queueing occurs. Because of that, a higher utilization requires a higher mean WIP level. This phenomenon may as well cause the system not to be able to follow the reference signal. A system cannot achieve a very high utilization with a very low WIP level. What the lower limit of the mean WIP level is to achieve a certain throughput, depends on the properties of the manufacturing system, but also on the control method. In Chapter 4, the relation between throughput and mean WIP level is determined experimentally for some common control methods.

In case a reference signal is offered to the system, demanding a throughput and a relatively high WIP level, the controller is able to let the system outputs follow this reference signal. In case a very difficult reference signal is offered, for example: achieve a certain throughput with a mean WIP level of zero, the reference signal cannot be followed. The

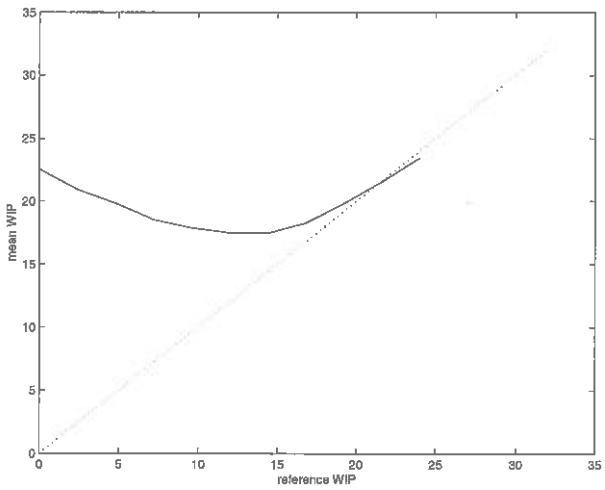

Figure 9.5: WIP versus reference WIP

definition output signals and cost criterion (8.4) determine the reaction of the system to an unrealizable input signal. The controller is expected to try and find a compromise. Because outputs are defined as buffer levels and finished goods levels and all output tracking errors are weighed equally, the magnitude of the error for the finished goods is expected to be comparable to the magnitude of the error for the buffer levels. Note that an absolute error of a few lots over the cumulative production corresponds to only a very small loss of throughput over a large interval. Because of that, the system might still perform very well (achieving a high throughput with a low mean WIP level), while the reference signal for buffer levels cannot be reached.

Experimentally, an optimal reference WIP level may be determined. For that purpose, experiments are done, starting with a high, easily realizable, reference WIP level. Then, the reference WIP level is decreased. The mean WIP level of the system is measured for all reference WIP levels. For a fixed throughput level, the mean WIP level of a system has the the characteristic curve shown in Figure 9.5, as a function of the reference WIP level. The system's WIP can easily follow relatively high reference WIP settings. As long as the reference can be followed the system's mean WIP level equals the reference WIP, so that the curve in the figure follows $y = x$. For lower reference WIP values, the deviation between reference WIP and achieved mean WIP increases. For very low reference WIP values, the system's mean WIP even shows an absolute increase. A possible cause for this increase is the fact that with a very low WIP level, the demanded throughput can not be met. The deviation in throughput can be compensated by allowing the WIP levels to rise. The measurements for Figure 9.5 have been performed at low utilization ($u = 0.6$). Higher utilizations show a similar relation between mean WIP and reference WIP, but the higher the utilization is, the less smooth the resulting figures are.

Figure 9.5 shows that a reference WIP value exists causing a minimum achieved mean WIP. In this chapter, this minimum has been approached by performing experiments for many reference WIP values. A rule of thumb for a reasonable reference WIP level seems to be to take half of the amount of WIP achieved by a simulation with reference WIP level of zero.

Design of Experiments

The performance of the MPC controlled system depends on several parameters. The optimal settings for these parameters depend on the required performance of the system. For example, a lower demanded throughput has a lower optimal reference WIP level. Furthermore, interaction between the parameters may be present.

Design of Experiments (DOE) may be a suitable method to find the optimal parameter settings with a relatively small number of experiments and only little trial-and-error. For a given throughput level, the optimal parameter settings are the settings resulting in the minimum mean WIP level. The parameters are referred to as the design variables. The approach may be the following:

- Define a domain for the design variables. This domain may be a region around an initial estimate of the design variables.
- Design an experiment. In case the number of design variables is not too large and the computations do not require a very large amount of time, the experiment may be a full-factorial design.
- Fit a response surface model. The class of model depends on the expected and observed response. For a quadratic response, a full quadratic model may be suitable. In case interaction between the design variables is observed, interaction terms should be added.
- Find the minimum of the response surface model. Constraints may bound the value of the design variables to the domain of the training points used to fit the model. The model may not be valid for point outside the domain of training points. In case the optimum is found at the edge of the training domain, a new model should be fitted, using better training data points. An optimum in the domain of the response surface model should be verified by discrete event simulations.

For some of the systems parameters, such as the reference WIP levels, an optimal setting can be found using DOE. This section does not have the objective of experimentally determining the optimal settings for the hypothetical flow line. Because of that, the parameters of the experiments presented in this section are not further optimized using DOE.

9.4 Experiments

The previous chapter describes an MPC controlled discrete event model of a manufacturing system. In this section, simulation experiments are used to measure the performance of the system. The main aspects of the performance of a manufacturing system for this research project are the throughput and the mean WIP level. Similar experiments determining the relation between throughput and WIP level for more common control strategies such as push and pull, are described in Chapter 4.

An MPC controller, identical for all throughput values, is designed using the following parameters:

- Control horizon and prediction horizon $m = p = 8$.

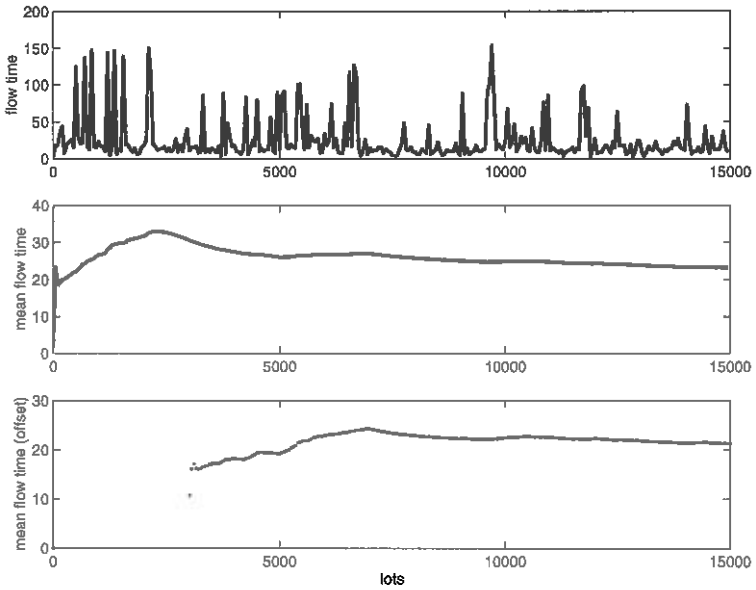

Figure 9.6: MPC experiment: (mean) flow times

- Ratio of weighing factors for error and input steps $w_u : w_y = 8 : 1$.

The settings of the conversion algorithm are the following:

- Conversion backlog error filtering factor $\alpha = 1$.
- No saturation on controller output or controller output steps.
- Production threshold $\epsilon_{\text{threshold}} = 0.3$.

The reference signal of the WIP levels is set at 0.25 for each product and each buffer. For 3 buffers and 8 products, the total reference WIP equals 6.

The experiments described in this chapter are performed using the MPC implementation of Section 8.2. Simulations are performed for both the equal product types variant ($k_d = k_p = 1$, see Chapter 3) and for a variant with unequal product types, with $k_d = 0.2$, $k_p = 5.0$. These settings correspond to the settings of the experiments for common control methods, described in Chapter 4, so that the results may be compared. Figure 9.6 show flow times and mean flow times for one simulation experiment. The mix parameters are $k_d = 0.2$ and $k_p = 5.0$ and the utilization is 90%. For computing the mean flow time, a range of lots has to be determined. The data from the first lots should not be used because of possible transient responses. Enough lots should be used, to decrease the stochastic effects. An offset of 3000 lots is chosen. The mean flow time is computed over the next 7000 lots. In Figure 9.6, the mean flow time using an offset of 3000 lots is presented as well. For each throughput level, 30 experiments are performed. Little's law (2.1) is used to determine the mean WIP level $\bar{w}$.

Table 9.1 shows the mean WIP level for several throughput levels. Because the mean effective process time t_e is set to 1 (unit time), a throughput of 0.9 corresponds to a utilization of 90%. Table 9.1 show the remarkable result that the higher variable situation

$k_d = k_p = 1$		$k_d = 0.2, k_p = 5.0$	
δ	$\bar{w}$	δ	$\bar{w}$
0.50	4.80	0.50	6.16
0.55	5.31	0.55	6.74
0.60	5.93	0.60	7.50
0.65	6.71	0.65	8.33
0.70	7.63	0.70	9.38
0.75	8.94	0.75	10.60
0.80	11.07	0.80	12.30
0.85	14.15	0.85	14.70
0.90	19.99	0.90	18.78

Table 9.1: Results of MPC experiments

($k_d = 0.2, k_p = 5.0$) requires a lower mean WIP level than the situation with equal product types ($k_d = k_p = 1.0$). The experimental results are also presented in Figure 9.7. An explanation for the remarkable phenomenon of a lower WIP level for a more variable situation is presented in the following example.

Example 9.4.1 *Assume a workstation, consisting of a buffer and a single machine, needs to process two lots. Both lots have a (deterministic) service time $t_s = 2$, and are present in the buffer. No further lots arrive during this small experiment. Because the lots have equal service times, it does not matter which lot is processed first. Assume lot 1 is processed first. The total flow time, consisting of queueing time in the buffer (φ_Q) and processing time (t_s), are presented in Table 9.2. Now assume the two lots have different process times,*

lot 1 first			
lot no.	φ_Q	t_s	φ_{total}
1	0	2	2
2	2	2	4
		sum:	6
		mean:	3

Table 9.2: Example: Flow times for equal lots

for example: $t_s = 1$ for lot 1, and $t_s = 3$ for lot 2. The mean process time equals still 2. Either lot 1 or lot 2 may be processed first. Table 9.3 shows the flow times for both options. Table 9.3 shows that for the situation of two lots with different service times queueing, the

lot 1 first				lot 2 first			
lot no.	φ_Q	t_s	φ_{total}	lot no.	φ_Q	t_s	φ_{total}
1	0	1	1	1	3	1	4
2	1	3	4	2	0	3	3
		sum:	5			sum:	7
		mean:	2.5			mean:	3.5

Table 9.3: Example: Flow time for different lots

one schedule results in a lower flow time than the other schedule. The better schedule results in a flow time lower than for two lots both having the mean service time.

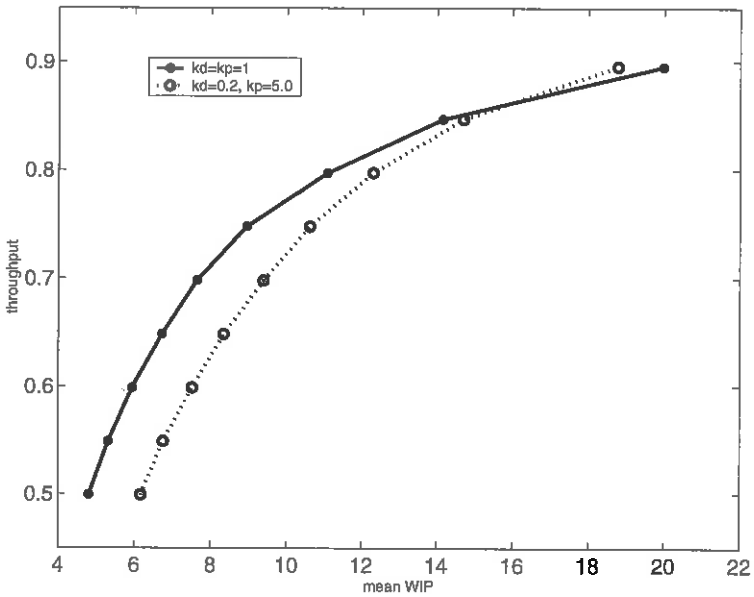

Figure 9.7: Performance of MPC control

The situation described in the example may occur in the MPC controlled flow line as well. The mix parameter $k_p = 5.0$ implies that the service times of the one product type are five times as long as the service times of the other (see Chapter 3). The simulation results in Figure 9.7 show that at higher utilization levels, when queueing plays an important role, the performance of the non-equal product types variant does not vary much from the equal product types variant ($k_d = k_p = 1$). The results suggest that the MPC controller is able to determine a schedule using the benefits of different service times, rather than to be disturbed by the higher variation for the non-equal product types variant.

Finite buffer capacities

In this chapter, a feedback controller is used to control a model of a manufacturing system. The feedback controller is designed using a continuous dynamic model. This model, a set of difference equations in state-space form, attempts to describe the relation between the inputs and the outputs of the controlled system. The inputs are target throughputs, offered to a conversion algorithm, which uses the input signal to generate appropriate events for the discrete event model. The outputs of the discrete event model are the buffer levels and the finished goods levels.

Only for a certain domain of inputs and outputs, the model is valid. For example, the buffer contents of the discrete event model cannot become negative. This constraint is not included in the continuous model. A discrepancy between continuous model and discrete event model may disable the controller to properly control the system. The simulation experiments presented in the previous subsection did not reveal any problems caused by the discrepancy the two models. In this subsection, an extra discrepancy between the discrete event model and the continuous dynamic model is introduced: finite buffer capacities.

So far, infinite buffer capacities have been assumed. In reality, buffer capacities are

never infinite. A lack of buffer space may cause blocking. Blocking is the phenomenon of stoppages at workstations because of a lack of storage space in downstream stages ([Alt97]). An example of the effects of blocking on the throughput of a simple system is presented in Section 2.4. A finite buffer capacity can easily be introduced in a discrete event model. The continuous model does not contain information about a constraint on the buffer levels. A constraint may be added to the MPC controller, but constrained MPC is computationally more complicated than unconstrained MPC (see Chapter 8).

Simulation experiments are performed to determine the influence of the extra discrepancy between the continuous model and the discrete event model. The controller used for the experiments is equal to the controller used in the previous subsections, so the controller design parameters are: $p = m = 8$, $w_u : w_y = 8 : 1$, and the conversion algorithm parameters are: $\alpha = 1$, $\epsilon_{\text{threshold}} = 0.3$ and the controller output is not constrained. The equal product types variant of the manufacturing system is observed, at target throughputs corresponding to utilizations of 50% to 90%. Again, for each throughput level, the mean flow time is measured from lot 3000 to lot 10000, and each experiment is repeated 30 times.

For modeling reasons, the buffer capacities are limited per product type. For example, a buffer capacity of 1 (Table 9.4) means that one lot of each product type can be stored in each buffer. For eight different product types and three buffers, a buffer capacity of 1 implies a total buffer capacity of 24 lots for the system. Table 9.4 show the results

buffer capacity 1			buffer capacity 2		
δ^*	δ	$\bar{w}$	δ^*	δ	$\bar{w}$
0.50	0.50	4.85	0.50	0.50	4.81
0.60	0.60	6.13	0.60	0.60	5.95
0.70	0.70	8.29	0.70	0.70	7.65
0.80	0.76	9.94	0.80	0.80	11.28
0.90	0.76	10.00	0.90	0.85	16.00

Table 9.4: Results of MPC experiments with finite buffer capacities

of the simulation experiments with limited buffer capacities. The column δ^* represents the target throughput, offered to the controller as a reference signal. The column δ represent the achieved throughput, resulting from the experiments. With a limited buffer capacity of 1 lot per product type per workstation, the MPC controlled flow line cannot achieve a throughput higher than 0.76 (lots/unit time), regardless of the reference throughput. Note that for a finite buffer capacity of 1, some outputs predicted by the continuous model are assumed to be an element of $\mathbb{R}$, while the output of the discrete event model is an element of only $0, 1$. The controller still performs well, despite the discrepancy between the continuous model and the discrete event model. The performance of the system is even slightly better than the performance of the system with infinite buffer capacity. When the throughput reference cannot be followed, the system remains stable and divides the production shortage over the different product types.

Chapter 10

Conclusions and discussion

10.1 Conclusions

The goal of this research project is to develop a method to apply a feedback controller for a manufacturing system. The performance of the feedback controlled manufacturing system is compared to the performance of several common control methods. The main criteria for performance in this research project are throughput and WIP level. A three machine multi-product manufacturing line is introduced as a case to test the performance of the control methods. Both the common control methods and the feedback controller are applied to this same case.

Common control methods

Four common control methods, *push*, *pull*, *conwip* and *POLCA*, are described. Simulation experiments are used to measure the performance of the control methods applied to the flow line. For each control method, the simulation experiments show that a higher throughput requires a higher inventory level. Furthermore, a more variable environment causes more queueing and therefore requires more inventory.

Hybrid systems as conwip and POLCA, using aspects, of both push and pull methods, outperforms the more rigid pure push and pure pull. Systems using kanban authorization cards suffer from *resident* WIP at lower utilization levels. For systems using product-specific kanban cards, the mix of kanbans should be adapted to the product mix. At last, the use of different customer demand models for the different control methods strongly influence the results of the experiments.

Feedback control.

Because the performance of a manufacturing system strongly depends on the control method, goal of this research project is to develop a more systematic approach for designing a control method. A possible approach is the use of feedback control. The method presented in this report to use feedback for controlling a manufacturing system consist of the following aspects:

A conversion algorithm.

A conversion algorithm is presented to convert a continuous input signal into an appropriate sequence of discrete events. The input is defined as a target throughput per sample, for each resource and product type. Compensating the conversion algorithm backlog error leads to better performance than leaving the compensation to the feedback controller.

A continuous model.

A continuous model of the system consisting of both the conversion algorithm and the discrete event model of the flow line is presented. System Identification is used to determine the structure of the linear state-space model, as well as to fit unknown model parameters. The model has a limited domain because the machines are subjected to a utilization constraint and the buffer contents can only have natural values.

A feedback controller.

The feedback controller uses Model Predictive Control (MPC). MPC determines an optimal controller output signal, that minimizes the output reference tracking errors, and the moves of the controller output for a number of samples ahead. The continuous model is used to predict the system response. The controller gain matrix for an unconstrained MPC controller is constant, and can therefore be determined off-line.

Experiments show that the MPC controlled model of the multi-product flow line performs well compared to the common control methods. At higher utilization levels, it requires less inventory than common control methods for equal throughput levels. Furthermore, it is less sensitive to variability, as is shown in the experiments with the non-equal product types. The performance remains good, also when a finite buffer capacity is introduced. This is remarkable, because the linear model does not correctly describe the non-linear response of output variables when the maximum capacity of a buffer is reached. Constraints on variables in the MPC controller are not necessary to successfully control the observed system.

10.2 Discussion

This research project describes a method to design a feedback controller for a discrete event model of a manufacturing system. The method is able to successfully control a simple case: a three machine multi-product flow line. A relevant question is whether the control method using feedback control can and should be applied to other manufacturing systems.

The described method using feedback control is not very complicated, but more complicated than the common control methods such as push, pull, POLCA or conwip. The implementation of the common methods requires only limited tools such as schedules or cards. Implementation of the feedback control method requires detailed information about the location, amount and flow of inventory. This information may only be present in sophisticated manufacturing environments. Furthermore, manufacturing systems only used at low utilization ratios, or having only little stochastic properties, may be equally served by a less complicated control method, as for example push control.

A feedback controller for a flow line is described. For manufacturing systems with highly flexible routing, such as a job shop, the described method is less suitable. Each product, route or recipe is a different product type for the continuous model. Each extra product type

introduces an extra set of input, output and state variables. Although the computations are not expensive, a large number of variables greatly increases the size of the problem.

When a continuous dynamic model is used for designing a feedback controller, the controller can only be as good as the continuous model. Not all phenomenons occurring in a manufacturing system may be easily captured in a continuous model. An example of such a phenomenon is the fact that buffer contents cannot be negative or greater than the buffer capacity. The discrepancy between discrete event model and the continuous dynamic model does not disturb the controller in the performed experiments. Nevertheless, the continuous model is an aspect of concern.

For this research project, the feedback controller is only tested in an unlikely environment. An infinite supply of raw materials and infinite customer demand is assumed. This assumption is made because one of the objectives of the research project is to compare the performance of common control strategies to feedback control. For this comparison, the maximum throughput for given WIP levels is determined, and the supply of materials or the arrival of customer demand should not limit the measured performance. In reality, the supply of raw materials and the customer demand is rarely unlimited.

In general, one of the objectives of feedback control is to suppress the influence of disturbances. The simulation experiments were performed for a relatively constant environments, with only stochastic process times. Presumably, the benefits of feedback control only really show in an environment subject to more disturbances. No experiments have yet been performed to verify this claim.

Chapter 11

Recommendations

As discussed in the previous chapter, describing the complex behavior of a discrete event model of a manufacturing system by a continuous dynamic model may introduce a discrepancy between the behavior of the discrete event model and the behavior predicted by the continuous model. The possibilities and limitations of modeling the behavior of the discrete event model with a continuous dynamic model should be an aspect of further research.

In the described simulation experiments, the feedback controlled discrete event model shows stable behavior. It is hard to define the conditions for an MPC controlled system to be stable. Further research to the issue of stability is recommended.

The surroundings of the system are very important for the performance of a system. Further research to feedback control for a manufacturing system might include an environment for the feedback controlled system, subject to disturbances in raw material supply and customer demand.

The use of MPC to control a manufacturing system seems especially beneficial for highly utilized systems. Furthermore, the feedback control problem for real manufacturing systems becomes very large because of the great number of variables. These are two reasons to use MPC to control part of a manufacturing system rather than the entire system. A common control method is suitable for lowly utilized parts of a manufacturing system, while feedback control can be used for higher utilized parts as the bottlenecks.

For this research project, the target of feedback control was to achieve a high throughput and a low WIP level. An additional control target, especially when the feedback controlled system is part of a larger manufacturing system, is to reduce the variability of the lots leaving the feedback controlled system. Less variability in the inter departure times at the end of the feedback controlled system causes less queueing in the downstream system.

Bibliography

[Alt97] T. Altiok. *Performance Analysis of Manufacturing Systems.* Springer, 1997.

[Buz93] J.A. Buzacott and J.G. Shantikumar. *Stochastic Models of Manufacturing Systems.* Prentice Hall, 1993.

[Hei97] C. Heij and P.M.J. van den Hof. *System Identification.* Dutch Institute of Systems and Control, 1997.

[Hop00] W.J. Hopp and M.L. Spearman. *Factory Physics: foundations of manufacturing management.* McGraw-Hill, 2000.

[Ken51] D.G. Kendall. Some problems in the theory of queues. *Journal of the Royal Statistical Society*, 1951.

[Kri00] A. Krishnamurthy and R. Suri. Re-examining the performance of push, pull and hybrid material control strategies for multi-product flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 2000.

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

[Mon83] Y. Monden. *Toyota Production System.* Industrial Engineering and Management Press, 1983.

[Roo01] J.E. Rooda and J.J.T. Kleijn. χ *Manual.* Systems Engineering Group, Eindhoven University of Technology, 2001.

[Ste84] G. Stephanopoulos. *Chemical process control: An introduction to theory and practice.* Prentice Hall, 1984.

[Sur00] R. Suri. *Quick Response Manufacturing: A Companywide Approach to Reducing Lead Times.* Productivity Press, 2000.

[Wul02] F.J.J. Wullems. Data collection for simulation of flow lines with blocking; the role of machine failure in throughput loss. Master's thesis, Systems Engineering Group, Eindhoven University of Technology, November 2002.

Appendix A

Implementation of Model Predictive Control

In this appendix, relations for computing the MPC controller gain matrices are described. For that purpose, first an augmented state space representation is presented. At last, a basic method is given for constructing the state in case it cannot be measured directly.

An augmented state representation

In chapter 7, state space models of the following form are derived:

$$x(k+1) = Ax(k) + Bu(k), \tag{A.1}$$

$$y(k) = Cx(k). \tag{A.2}$$

The model describes the system in terms of state variable x as a function of input u. In some cases, it may be beneficial to describe the system as a function of Δu instead of u. For example, a commonly used cost criterion for an MPC controller (8.4) is expressed in Δu. An alternative representation of the model may be obtained by choosing different state variables. The augmented state representation described in this subsection is used in the functions of the MATLAB® MPC Toolbox. Define the augmented state variable ξ:

$$\xi(k) = \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \qquad \text{with } \Delta x(k) = x(k) - x(k-1). \tag{A.3}$$

The model described by Equations A.1 and A.2 can be rewritten using the augmented state variable ξ:

$$\xi(k+1) = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \xi(k) + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u(k), \tag{A.4}$$

$$y(k) = \begin{bmatrix} 0 & I \end{bmatrix} \xi(k). \tag{A.5}$$

This model represents the same system as the model of (A.1) and (A.2). The system's matrix may be labelled using the index A for augmented:

$$\xi(k+1) = A_A \xi(k) + B_A \Delta u(k), \tag{A.6}$$

$$y(k) = C_A \xi(k). \tag{A.7}$$

The augmented state space representation described in this section is used in the next sections to compute the optimal controller gain matrix with respect to (8.4) and to predict $e^{p}_{\Delta u=0}$: the output error for a constant system input.

A.1 The MPC Controller gain matrix

Before the MPC controller gain matrix can be defined, few variables must be introduced. Define two matrices, $\bar{A}$ and $\bar{C}$, containing the model dynamics:

$$\bar{A} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB+B & B & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \sum_{i=0}^{m-1} A^i & \cdots & AB+B & B \\ \sum_{i=0}^{m} A^i & \cdots & \ddots & AB+B \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=0}^{p-1} A^i & \cdots & \cdots & \sum_{i=0}^{p-m} A^i \end{bmatrix}, \tag{A.8}$$

and

$$\bar{C} = \begin{bmatrix} C & & 0 \\ & C & \\ & & \ddots \\ 0 & & C \end{bmatrix}, \tag{A.9}$$

and define the product of matrices $\bar{A}$ and $\bar{C}$ as Y:

$$Y = \bar{C}\bar{A}. \tag{A.10}$$

Assume the system has n_x state variables, n_y outputs and n_u inputs. The prediction horizon is denoted as p and the control horizon as m. The dimension of $\bar{A}$ is $p\ n_x \times m\ n_u$ and the dimension of $\bar{C}$ equals $p\ n_y \times p\ n_x$ Therefore, Y has dimensions $p\ n_y \times m\ n_u$. Q and R are diagonal matrices (sized $p\ n_y$ and $m\ nu$ square) containing the weighing factors for output deviations e and controller output steps Δu respectively.

Introduce a notation to store the predicted values for state variables, output variables

and input variables in vector form:

$$x_s^p(k+1) = \begin{bmatrix} x_1(k+1|k) \\ \vdots \\ x_{nx}(k+1|k) \\ \cdots \\ x_1(k+p|k) \\ \vdots \\ x_{nx}(k+p|k) \end{bmatrix}, \tag{A.11}$$

$$y_s^p(k+1) = \begin{bmatrix} y_1(k+1|k) \\ \vdots \\ y_{ny}(k+1|k) \\ \cdots \\ y_1(k+p|k) \\ \vdots \\ y_{ny}(k+p|k) \end{bmatrix}, \tag{A.12}$$

$$u_s^m(k+1) = \begin{bmatrix} u_1(k|k) \\ \vdots \\ u_{nu}(k|k) \\ \cdots \\ u_1(k+m-1|k) \\ \vdots \\ u_{nu}(k+m-1|k) \end{bmatrix}. \tag{A.13}$$

Note that both state and output variables are predicted until prediction horizon p, but the controller output only until control horizon m. Similarly, the predicted deviations e of the system output y_s from the reference trajectory y_r are defined as $e^p(k+1) = y_s^p(k+1) - y_r^p(k+1)$.

The controller gain matrices for linear, unconstrained MPC can be determined off-line. In the unconstrained case, the quadratic performance criterion of Equation 8.4 reduces to a least squares problem.

Because of the superposition principle, the responses to a constant input value and deviations from this constant value may be superposed. Let $e^p_{\Delta u=0}$ denote the deviations e in case $\Delta u = 0$ (u is kept at a constant value). The unconstrained MPC control law provides the deviations in controller output ($\Delta u(k) = u(k) - u(k-1)$) as a linear combination of the predicted deviations e of the system output y_s from the reference trajectory y_r:

$$\Delta u_i^m = -[Y^T Q Y + R]^{-1} Y^T Q \; e^p_{0,\Delta u=0}, \tag{A.14}$$

$$\Delta u_i^m = K_{MPC} \; e^p_{0,\Delta u=0}. \tag{A.15}$$

Because computing the inverse in Equation A.14 is numerically inefficient, an alternative formulation should be used. For example, in MATLAB®, the *left divide* operator '\' solves the system by Gaussian elimination.

$$\Delta u_i^m = \begin{bmatrix} QY \\ R \end{bmatrix} \setminus \begin{bmatrix} Q e^p_{0,\Delta u=0} \\ 0 \end{bmatrix}, \tag{A.16}$$

or:

$$K_{MPC} = \begin{bmatrix} QY \\ R \end{bmatrix} \setminus \begin{bmatrix} Q \\ 0 \end{bmatrix}. \tag{A.17}$$

This section provided a method to determine the MPC controller gain matrix K_{MPC}. The optimal controller output Δu is obtained by multiplying the controller gain matrix by $e^p_{\Delta u=0}$ (see Chapter 8), the expected deviations of the system output from the reference signal in case u was kept at a constant value. The next subsection provides a method to compute $e^p_{\Delta u=0}$.

A.2 Expected output deviation $e^p_{\Delta u-0}$

The previous subsection gave the optimal controller output as a function of the expected deviation of the output and the reference signal in case the input was kept at a constant value ($e^p_{\Delta u=0}$). This subsection provides a method to compute the values of $e^p_{\Delta u=0}$. The error $e^p_{\Delta u=0}$ is the difference between predicted output and reference output until the prediction horizon:

$$e^p_{\Delta u=0} = y^p_{\Delta u=0} - y^p_r. \tag{A.18}$$

Recall the augmented state space representation from equations (A.4) and (A.5). The augmented state representation may be used to easily compute the expected outputs for $\Delta u = 0$, $y^p_{\Delta u=0}$. Matrices of the augmented representation are denoted with the index A.

$$\xi(k+1) = \Phi_A \xi(k) + \Gamma_A \Delta u(k), \tag{A.19}$$

$$y(k) = C_A \xi(k). \tag{A.20}$$

Because $\Delta u = 0$ and the augmented state representation only contains Δu instead of u, the prediction for y_s for a fixed input value can be easily computed:

$$\xi(k+n) = \Phi^n_A \xi(k), \tag{A.21}$$

$$y_A(k+n) = C_A\, \Phi^n_A \xi(k). \tag{A.22}$$

Substitute (A.22) in (A.18):

$$e^p_{\Delta u=0} = \begin{bmatrix} C_A\Phi_A \\ C_A\Phi^2_A \\ \vdots \\ C_A\Phi^p_A \end{bmatrix} [\xi] - y^p_r. \tag{A.23}$$

A.3 Internal state reconstruction

The MATLAB® implementation of the MPC controller in the MPC Toolbox requires a value for the state of the system for determining the controller output (8.7). In general, the (augmented) state of a system cannot be measured directly. For those cases, an observer may be designed to reconstruct the state, using the measured outputs and the system input. This section describes the standard (augmented) state reconstruction used by the MATLAB® MPC Toolbox.

The augmented state space model is represented by (A.4) and (A.5). At each sample time, the reconstruction of the state is updated. The updated state reconstruction depends on the previous reconstruction, the known controller output and the measured system output. The element y of the augmented state variable, representing the system output, is updated using the actual measurement from the controlled system. Let $\hat{\xi}$ denote the reconstruction of the controlled system state, and Δu_p the change of controlled system input.

$$\hat{\xi}(k+1) = \begin{bmatrix} A & 0 \\ CA & 0 \end{bmatrix} \hat{\xi}(k) + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u_p(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} y_p(k). \tag{A.24}$$

At each sample instant, the system output y_s is measured and sent to the controller. The controller uses the internal model of the controlled system, (containing matrices A, B and C to reconstruct the augmented state of the system). The controller output y_c is obtained by multiplying manufacturing system output y_s, output reference y_r and augmented state reconstruction $\hat{\xi}$ with controller gain matrices.

Appendix B

Chi implementation code

Appendix B provides the code of the implementation of the models in the specification language χ. Note that the systems share equal components.

B.1 Push model

```
// model for three machine multi product flow line
// PUSH controlled
// H. Ploegmakers, 2002
// reads orders from file orders.in
// generated by genlist.chi
// usage: ./push kd kp D

const N: nat = 8
type order=nat#nat          // serial nr, product type
, lot=nat#nat#real          // serial nr, product type, release time
, mix=real#real#real#real   // lambda1, lamba2, tau1, tau2

// function to calculate demands and process times (lambda and tau) for both
// product types, function of mix parameters kd, kp and total demand D
func kdkp2lt(kd, kp, D: real) -> mix =
|[ l1, l2, t1, t2: real
 | l2:= 2*D/(N*(kd + 1))
 ; l1:= kd*l2
 ; t2:= (l1 + l2)/(l1*kp + l2)
 ; t1:= t2*kp
 ; ret <l1, l2, t1, t2>
]|

// generator
proc G(fo: ?file, a: !lot, D: real)=
|[ o: order, e: -> real
 | e:= exponential(1/D)
 ; *[true -> fo?o; a!<o.0, o.1, time>; delta sample e]
]|

// buffer, infinite capacity
proc B(a: ?lot, b: !lot)=
|[xs: lot*, x: lot
 |xs:= []
 ; *[           a?x      -> xs:= xs++[x]
    | len(xs)>0; b!hd(xs) -> xs:= tl(xs)
    ]
]|


// machine
```

```
proc M(a: ?lot, b: !lot, kd, kp, D: real)=
|[ e: (-> real)^2, x: lot, m: mix
 | m:= kdkp2lt(kd, kp, D)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true
      -> a?x
       ; delta sample e.(2*x.1 div N)
       ; b!x
    ]
]|

// exit
proc E(a: ?lot, startup, tot: nat)=
|[x: lot, ct,tp,avgct: real, nr: nat
 | avgct:= 0.0; nr:= 0; tp:= 0.0
 ;*[ true
    -> a?x
     ; [x.0 <  startup -> skip
       |x.0 >= startup
        ->
nr:= nr+1; ct:= time-x.2
         ; avgct:= ((nr-1)*avgct+ct)/nr
         ; tp:= (nr+startup)/time
         ; [x.0 =  tot -> !avgct,"\t",x.0,"\t",tp,"\t",tp*avgct,"\n"
                        ; terminate
           |x.0 /= tot -> skip
           ]
      ]
   ]
]|

syst LINE(kd, kp, D: real)=
|[ gb1, b1m1, m1b2, b2m2, m2b3, b3m3, m3e: -lot
 | G(filein("orders.in"), gb1, D)
|| B(gb1, b1m1)
|| M(b1m1, m1b2, kd, kp, D)
|| B(m1b2, b2m2)
|| M(b2m2, m2b3, kd, kp, D)
|| B(m2b3, b3m3)
|| M(b3m3, m3e, kd, kp, D)
|| E(m3e, 5000, 10000)
]|

xper(kd, kp, D: real) = |[ LINE(kd, kp, D) ]|
```

B.2 Pull model

```
// model for three machine multi product flow line
// PULL controlled
// H. Ploegmakers, 2002
// reads orders from file orders.in
// generated by genlist.chi
// usage: ./pull kd kp nk
// note: orders.in must be generated with equal kd!

const N: nat = 8
type order=nat#nat          // nr, product type
, lot=nat#nat#real#real     // nr, product type, tr, t(entering buffer)
, mix=real#real#real#real   // lambda1 lamba2 tau1 tau2

// function to initialize kanban buffers
func initkanban (a: nat) -> nat* =
|[ i: nat, ks: nat*
 | ks:= []
 ; *[ a > 0
    -> i:= N-1
     ; *[ i >= 0 -> ks:= ks++[i]; i:= i-1
 ]
     ; a:= a-1
   ]
 ; ret ks
]|

// function to calculate process times (tau) for both
 product types
func kdkp2lt(kd, kp, D: real) -> mix =
|[ l1, l2, t1, t2: real
 | l2:= 2*D / (N*(kd + 1))
 ; l1:= kd*l2
 ; t1:= (kd + 1) * kp/(kd * kp + 1)
 ; t2:= (kd + 1) / (kd * kp + 1)
 ; ret <l1, l2, t1, t2>
]|

// function that determines which kanban to send to machine: selects
// the longest waiting kanban, for which a product is available
func chooseproduct(ks: nat*, nb: nat^N, i: nat) -> nat =
|[ k: nat
 | [ len(ks)  <  i -> ret 0
   | len(ks) >=  i -> k:=hr(take(ks, i))
                    ; [ nb.k > 0 -> ret i
                      | nb.k = 0 -> ret chooseproduct(ks, nb, i+1)
                      ]
   ]
]|

// machine: comp. MIS p.45
proc M(a: (?lot)^N, b: (!lot)^N, c: ?nat, kd, kp: real)=
|[ e: (-> real)^2, x: lot, kanban: nat, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true
    -> c?kanban
     ; a.kanban?x
     ; delta sample e.(2*kanban div N)
     ; b.kanban!x
   ]
]|

// generator
proc G(a: !lot, id: nat)=
|[ n: nat
 | n:= 0
 ; *[ true -> a!<n, id, time, 0.0>; n:=n + 1]
]|

// buffer
```

```
proc B(a: ?lot, b: !lot, c: !nat, id: nat)=
|[ x: lot, xs: lot*
 | xs:=[]
 ; *[            a?x      -> xs:=xs ++ [<x.0, x.1, x.2, time>]
    | len(xs)>0 ; b!hd(xs) -> xs:=tl(xs); c!id
    ]
]|

// buffer: updates content to KBb
proc Bb(a: ?lot, b: !lot, c: !nat, d: !nat, id: nat)=
|[ x: lot, xs: lot*
 | xs:=[]
 ; *[             a?x      -> xs:=xs++[x]; d!1
    | len(xs)> 0 ; b!hd(xs) -> xs:=tl(xs); c!id; d!0
    ]
]|

// kanban-buffer
proc KB(a: (?nat)^N, b: !nat, k: nat)=
|[ ks: nat*
 | ks:= initkanban(k)
 ; *[ j: nat <- 0..N:  a.j?k    -> ks:=ks++[k]
    | len(ks)>0      ; b!hd(ks) -> ks:=tl(ks)
    ]
]|

// kanban buffer, monitoring the number of products for each type in buffer
proc KBb(a: (?nat)^N, b: !nat, k: nat, d: (?nat)^N)=
|[ks: nat*, i, bc: nat, nb: nat^N
 | nb:= <0, 0, 0, 0, 0, 0, 0, 0>
 ; ks:= initkanban(k)
 ; *[ j: nat <- 0..N: a.j?k
    -> ks:=ks ++ [k]
    | chooseproduct(ks, nb, 1) > 0; b!hr(take(ks, chooseproduct(ks, nb, 1)))
    -> i:=chooseproduct(ks, nb, 1)
     ; ks:= take(ks, i-1) ++ drop(ks, i)
    | j: nat <- 0..N: d.j?bc
    -> [bc = 0 -> nb.j:=nb.j - 1
       |bc = 1 -> nb.j:=nb.j + 1
       ]
    ]
]|

// exit process
proc E(a: (?lot)^N, fo: ?file, startup, tot: nat)=
|[ o: order, x: lot, ot, nr, nrr: nat, ct, avgct, tp: real
 | avgct:=0.0; nr:=0; nrr:=0
 ; *[ true
    -> fo?o
     ; ot:= (o.1)
     ; a.ot?x
     ; nrr:=nrr+1
     ; [nrr < startup -> skip
       |nrr >=startup
       -> nr:=nr+1; ct:=time-x.2
        ; avgct:=((nr-1)*avgct+ct)/nr
        ; tp:=(nr+startup)/time
        ; [nrr = tot
          ->
!avgct, "\t", nrr, "\t", tp, "\t", tp*avgct, "\n"
           ; terminate
          |nrr /= tot -> skip
          ]
       ]
    ]
]|

syst LINE(kd, kp: real, nk: nat)=
|[ gm1, m1b1, b1m2, m2b2, b2m3, m3b3, b3e: (-lot)^N,
   b1kb1, b2kb2, b3kb3, b1kb2, b2kb3: (-nat)^N,
   kb1m1, kb2m2, kb3m3: -nat
 | i: nat <- 0..N: G(gm1.i, i)
||                 M(gm1, m1b1, kb1m1, kd, kp)
|| i: nat <- 0..N: Bb(m1b1.i, b1m2.i, b1kb1.i, b1kb2.i, i)
```

```
||                  M(b1m2, m2b2, kb2m2, kd, kp)
|| i: nat <- 0..N: Bb(m2b2.i, b2m3.i, b2kb2.i, b2kb3.i, i)
||                  M(b2m3, m3b3, kb3m3, kd, kp)
|| i: nat <- 0..N: B(m3b3.i, b3e.i, b3kb3.i, i)
||                  E(b3e, filein("orders.in"), 5000, 10000)
||                  KB(b1kb1, kb1m1, nk)
||                  KBb(b2kb2, kb2m2, nk, b1kb2)
||                  KBb(b3kb3, kb3m3, nk, b2kb3)
]|

xper(kd, kp: real, nk: nat) = |[ LINE(kd, kp, nk) ]|
```

B.3 (Hybrid) Conwip model

```
// model for three machine multi product flow line
// controlled by pull-conwip
// H. Ploegmakers, 2002
// reads orders from file orders.in

const N: nat = 8
type order=nat#nat          // number, product-type
, lot=nat#nat#real          // number, product-type, release-time
, mix=real#real#real#real   // lambda1 lamba2 tau1 tau2

// function to initialize kanban buffers
// note: generic kanbans are used
func initkanban(a: nat) -> nat* =
|[ ks: nat*
 | ks:= []
 ; *[a > 0  ->  ks:= ks ++ [1]; a:= a-1 ]
 ; ret ks
]|

// function to calculate demands and process times (lambda and tau) for both
// product types, function of mix parameters kd, kp and total demand D
func kdkp2lt(kd, kp, D: real) -> mix =
|[ l1, l2, t1, t2: real
 | l2:= 2*D/(N*(kd + 1))
 ; l1:= kd*l2
 ; t2:= (l1 + l2)/(l1*kp + l2)
 ; t1:= t2*kp
 ; ret <l1, l2, t1, t2>
]|

//generator, reads product type from file
proc G(fo: ?file, a: !lot)=
|[ o: order
 | *[ fo?o; a!<o.0, o.1, time> ]
]|

// buffer: infinite capacity
proc B(a: ?lot, b: !lot)=
|[ xs: lot*, x: lot
 | xs:= []
; *[           a?x      -> xs:= xs ++ [x]
   | len(xs)>0; b!hd(xs) -> xs:= tl(xs)
   ]
]|

// buffer, infinite capacity, releasing kanbans
proc BR(a: ?lot, b: !lot, c: !nat)=
|[ x: lot, xs: lot*
 | xs:= []
 ; *[              a?x       -> xs:= xs ++ [x]
    | len(xs)>0 ;   b!hd(xs)  -> xs:= tl(xs); c!1
    ]
]|

// kanban-buffer
proc KB(a: ?nat, b: !nat, k: nat)=
|[ ks: nat*
 | ks:= initkanban(k)
 ; *[             a?k      -> ks:= ks ++ [k]
    | len(ks) > 0; b!hd(ks) -> ks:= tl(ks)
    ]
]|

// machine authorized by kanbans
proc MA(a: ?lot, b: !lot, c: ?nat, kd, kp: real)=
|[ e: (-> real)^2, x: lot, kanban: nat, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true ->
       c?kanban
```

```
      ; a?x
      ; delta sample e.(2*x.1 div N)
      ; b!x
    ]
]|

// machine
proc M(a: ?lot, b: !lot, kd, kp: real)=
|[ e: (-> real)^2, x: lot, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true ->
        a?x
      ; delta sample e.(2*x.1 div N)
      ; b!x
    ]
]|

proc E(a: ?lot, startup, tot: nat)=
|[x: lot, nr, nrr: nat, ct, avgct, tp: real
 | avgct:= 0.0; nr:= 0
 ; *[true ->
       a?x
     ; nrr:= nrr + 1
     ; [ nrr <  startup -> skip
       | nrr >= startup ->
           nr:= nr + 1
         ; ct:= time-x.2
         ; avgct:= ((nr-1)*avgct + ct)/nr; tp:= (nrr)/time
         ; [ nrr = tot -> !avgct, "\t", nrr, "\t", tp, "\t", tp*avgct, "\n"
                          ; terminate
           | nrr /= tot -> skip
           ]
       ]
   ]
]|

syst LINE(kd, kp: real, nk: nat)=
|[ gm1, m3b3, b3e, m1b1, b1m2, m2b2, b2m3: -lot, b3kb, kbm1: -nat
 | G(filein("orders.in"), gm1)
|| MA(gm1, m1b1, kbm1, kd, kp)
|| B(m1b1, b1m2)
|| M(b1m2, m2b2, kd, kp)
|| B(m2b2, b2m3)
|| M(b2m3, m3b3, kd, kp)
|| BR(m3b3, b3e, b3kb)
|| E(b3e, 5000, 10000)
|| KB(b3kb, kbm1, nk)
]|

xper(kd, kp: real, nk: nat) = |[ LINE(kd, kp, nk) ]|
```

B.4 (Pull) Conwip model

```
// model for three machine multi product flow line
// controlled by pull-conwip
// H. Ploegmakers, 2002
// reads orders from file orders.in

const N: nat = 8
type order = nat#nat           // number, product-type
, lot = nat#nat#real           // number, product-type, release-time
, mix = real#real#real#real    // lambda1 lamba2 tau1 tau2

func initkanban (a: nat) -> nat* =
|[ i: nat, ks: nat*
 | ks:= []
 ; *[a > 0
    -> i:= N-1
     ; *[ i >= 0 -> ks:= ks ++ [i]; i:= i - 1 ]
     ; a:= a-1
    ]
 ; ret ks
]|

// function to calculate process times (tau) for both product types
func kdkp2lt(kd, kp, D: real) -> mix =
|[ l1, l2, t1, t2: real
 | l2:= 2*D / (N*(kd + 1))
 ; l1:= kd*l2
 ; t1:= (kd + 1) * kp/(kd * kp + 1)
 ; t2:= (kd + 1) / (kd * kp + 1)
 ; ret <l1, l2, t1, t2>
]|


// generator
proc G(a: !lot, id: nat)=
|[ n: nat
 | n:= 0
 ; *[ a!<n, id, time> -> n:= n + 1]
]|

// buffer, infinite capacity
proc B(a: ?lot, b: !lot)=
|[ xs: lot*, x: lot
 | xs:= []
 ; *[            a?x      -> xs:= xs++[x]
    | len(xs)>0; b!hd(xs) -> xs:= tl(xs)
    ]
]|

// buffer, infinite capacity, releasing kanbans
proc BR(a: ?lot, b: !lot, c: !nat, id: nat)=
|[ x: lot, xs: lot*
 | xs:= []
 ; *[             a?x         -> xs:= xs++[x]
    | len(xs) > 0; b!hd(xs)    -> xs:= tl(xs); c!id
    ]
]|

// kanban-buffer
proc KB(a: (?nat)^N, b: !nat, k: nat)=
|[ ks: nat*
 | ks:= initkanban(k)
 ; *[ j: nat <- 0..N: a.j?k    -> ks:= ks++[k]
    | len(ks)>0;      b!hd(ks) -> ks:= tl(ks)
    ]
]|

// machine
proc M(a: ?lot, b: !lot, kd, kp: real)=
|[ e: (-> real)^2, x: lot, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
```

```
  ; e.1:= exponential(m.3)
  ; *[ true
     -> a?x
      ; delta sample e.(2*x.1 div N)
      ; b!x
     ]
]|

// machine authorized by kanbans
proc MA(a: (?lot)^N, b: !lot, c: ?nat, kd, kp: real)=
|[ e: (-> real)^2, x: lot, kanban: nat, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true
     -> c?kanban
      ; a.kanban?x
      ; delta sample e.(2*kanban div N)
      ; b!x
     ]
]|

// machine: last machine, sends through N channels
proc ME(a: ?lot, b: (!lot)^N, kd, kp: real)=
|[ e: (-> real)^2, x: lot, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[ true
     -> a?x
      ; delta sample e.(2*x.1 div N)
      ; b.(x.1)!x
     ]
]|

proc E(a: (?lot)^N, fo: ?file, startup, tot: nat)=
|[o: order, x: lot, ot, nrr, nr: nat, ct, avgct, tp: real
 | avgct:= 0.0; nr:= 0; nrr:= 0
 ; *[ true
     -> fo?o
      ; ot:= o.1
      ; a.ot?x
      ; nrr:= nrr+1
      ; [nrr < startup -> skip
        |nrr >= startup
        -> nr:= nr+1
         ; ct:= time-x.2
         ; avgct:= ((nr-1)*avgct+ct)/nr
         ; tp:= nrr/time
         ; [ nrr= tot
           -> !avgct, "\t", nrr, "\t", tp, "\t", tp*avgct, "\n"
            ; terminate
           | nrr /= tot -> skip
           ]
        ]
     ]
]|

syst LINE(kd, kp: real, nk: nat)=
|[ gm1, m3b3, b3e: (-lot)^N, m1b1, b1m2, m2b2, b2m3: -lot,
   b3kb: (-nat)^N, kbm1: -nat
 | i: nat <- 0..N:  G(gm1.i, i)
||                  MA(gm1, m1b1, kbm1, kd, kp)
||                  B(m1b1, b1m2)
||                  M(b1m2, m2b2, kd, kp)
||                  B(m2b2, b2m3)
||                  ME(b2m3, m3b3, kd, kp)
|| i: nat <- 0..N:  BR(m3b3.i, b3e.i, b3kb.i, i)
||                  E(b3e, filein("orders.in"), 5000, 10000)
||                  KB(b3kb, kbm1, nk)
]|

xper(kd, kp: real, nk: nat) = |[ LINE(kd, kp, nk) ]|
```

B.5 POLCA model

```
// model for three machine multi product flow line
// controlled by POLCA
// H. Ploegmakers, 2002
// reads orders from file orders.in

type tps=real^3     // release-time M1, rtM2, rtM3
, lot = nat#nat#tps // nr, product type, release times
, mix = (real)^4    // lambda1 lamba2 tau1 tau2
, order = nat#nat   // number, product-type
const N: nat = 8

// function to initialize kanban buffers
// note: generic kanbans are used
func initkanban(a: nat) -> nat* =
|[ ks: nat*
 | ks:= []
 ; *[ a > 0 -> ks:= ks++[1]; a:= a-1 ]
 ; ret ks
]|

// function to calculate process times (tau) for both product types
func kdkp2lt(kd, kp, D: real) -> mix =
|[ l1, l2, t1, t2: real
 | l2:= 2*D / (N*(kd + 1))
 ; l1:= kd*l2
 ; t1:= (kd + 1) * kp/(kd * kp + 1)
 ; t2:= (kd + 1) / (kd * kp + 1)
 ; ret <l1, l2, t1, t2>
]|

// function inserting lots in buffer
// sorted by MRP release time
func insbuf(y: lot, xs: lot*, i: nat) -> lot* =
|[ ys: lot*
 |  [ len(xs) = 0                                -> ys := [y]
    | len(xs) > 0 and (y.2).i > ((hr(xs)).2).i -> ys:= xs++[y]
    | len(xs) > 0 and (y.2).i <=((hr(xs)).2).i ->
        [ (y.2).i <= ((hd(xs)).2).i -> ys := [y]++xs
        | (y.2).i >  ((hd(xs)).2).i -> ys := [hd(xs)] ++ insbuf(y, (tl(xs)), i)
        ]
    ]
; ret ys
]|

// returns waiting time in queue
func phiQ(ca2, ce2, u: real) -> real =
|[ te: real
 | te:= 1.0
 ; ret (ca2+ce2)*u*te/(2*(1-u))
]|

// function: linking equation
// returns cd^2 as a function of ca^2, ce^2, u
func cd2(ca2, ce2, u: real) -> real =
|[ ret (1-u^2)*ca2 + u^2*ce2 ]|

// function providing effective sigma^2 for two exponential distributions
// function of lambda1, lambda2, fraction of first product type (50% -> 0.5)
func sigma(l1, l2, c: real) -> real =
|[ ret (2*c-c^2)*(l1^2) + (1-c^2)*(l2^2) - 2*c*(1-c)*(l1*l2) ]|

// generator, reads product type from file
proc G(fo: ?file, kd, kp, D: real, a: !lot)=
|[ o: order, m: mix, sigmatau, phiQ1, phiQ2: real, i: nat
 | m:= kdkp2lt(kd, kp, D)
 ; sigmatau:= sigma(m.2, m.3, m.0 / (m.0+m.1))
 ; phiQ1:= phiQ(1.0, sigmatau, D)
 ; phiQ2:= phiQ(cd2(1.0, sigmatau, D), sigmatau, D)
 ; *[true
    -> fo?o
     ; i:= 2*o.1 div N + 1
```

```
      ; a!<o.0, o.1, <time, time+phiQ1+m.i, time+phiQ1+phiQ2+ 2*m.i>>
    ]
]|

// buffer: inserts lots using insbuf
proc B(a: ?lot, b: !lot, isort: nat)=
|[ x: lot, xs: lot*, r: bool
 | xs:= []
 ; *[ true; a?x
    -> xs:= insbuf(x, xs, isort)
     ; r:= false
    | not(r) and len(xs) > 0; delta max(0.0,hd(xs).2.isort - time)
    -> r:= true
    | r; b!hd(xs)
    -> xs:= tl(xs)
     ; r:= false
    ]
]|

// buffer releasing kanbans: inserts lots using insbuf
proc BR(a: ?lot, b: !lot, c: !nat, isort: nat)=
|[ x: lot, xs: lot*, r: bool
 | xs:= []
 ; *[ true; a?x
    -> xs:= insbuf(x, xs, isort)
     ; c!1
     ; r:= false
    | not(r) and len(xs) > 0; delta max(0.0,hd(xs).2.isort - time)
    -> r:= true
    | r; b!hd(xs)
    -> xs:= tl(xs)
     ; r:= false
    ]
]|

// machine; authorized by kanbans
proc M(a: ?lot, b: !lot, c: ?nat, kd, kp: real)=
|[ e: (-> real)^2, x: lot, kanban: nat, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[true
    -> c?kanban
     ; a?x
     ; delta sample e.(2*x.1 div N)
     ; b!x
    ]
]|


// 'push'-machine: without kanbans
proc ME(a: ?lot, b: !lot, kd, kp: real)=
|[ e: (-> real)^2, x: lot, m: mix
 | m:= kdkp2lt(kd, kp, 1.0)
 ; e.0:= exponential(m.2)
 ; e.1:= exponential(m.3)
 ; *[true
    -> a?x
     ; delta sample e.(2*x.1 div N)
     ; b!x
    ]
]|

// kanban-buffer
proc KB(a: ?nat, b: !nat, nk: nat)=
|[ ks: nat*, k: nat
 | ks:= initkanban(nk)
 ; *[              a?k      -> ks:= ks ++ [k]
    | len(ks) > 0; b!hd(ks) -> ks:= tl(ks)
    ]
]|

// exit
proc E(a: ?lot, startup, tot: nat)=
```

```
|[ x: lot, ct, tp, avgct: real, nr, nrr: nat
 | avgct:= 0.0; nr:= 0; nrr:= 0
 ; *[ true
    -> a?x
     ; nrr:= nrr + 1
     ; [ nrr <  startup -> skip
       | nrr >= startup
       -> nr:= nr + 1
        ; ct:= time-(x.2).0
        ; avgct:= ((nr-1)*avgct+ct)/nr
        ; tp:= nrr/time
        ; [ nrr = tot  -> !avgct, "\t", nrr, "\t", tp, "\t", tp*avgct, "\n"
                        ; terminate
          | nrr /= tot -> skip
          ]
       ]
    ]
]|

syst LINE(kd, kp, D: real, nk1, nk2: nat)=
|[ gm1, m1b2, b2m2, m2b3, b3m3, m3b4, b4e: -lot, b3kb1, kb1m1, b4kb2, kb2m2: -nat
 | G(filein("orders.in"), kd, kp, D, gm1)
|| M(gm1, m1b2, kb1m1, kd, kp)
|| B(m1b2, b2m2, 1)
|| M(b2m2, m2b3, kb2m2, kd, kp)
|| BR(m2b3, b3m3, b3kb1, 2)
|| ME(b3m3, m3b4, kd, kp)
|| BR(m3b4, b4e, b4kb2, 2)
|| E(b4e, 5000, 10000)
|| KB(b3kb1, kb1m1, nk1)
|| KB(b4kb2, kb2m2, nk2)
]|

xper(kd, kp, D: real, nk1, nk2: nat) = |[ LINE(kd, kp, D, nk1, nk2) ]|
```

B.6 MPC controlled model

```
// Discrete Event System: 3 machines, N products
// system is controlled by MPC controller (in Matlab)
// compile by: c76 DESlimbuf.chi pythoncontrol.ext
// Usage: "DESlimbuf kd kp throughput wipref bufsize hor weightu tend"
// kd: product group demand ratio
// kp: product group process time ratio
//
// external functions (specified in pythoncontrol.py):
// ext initmatlab(kd:real, h: nat, wu, ut, wref: real)->bool
// ext controlaction(y:real^24)-> real^24
// ext closematlab()->bool

const N: nat = 8        // only pair numbers
, m: nat = 4            // # machines + generator
, mb: nat = 5           // # buffers: m + 1
, N4: nat = 32          // N4=m*N
type lot = nat#nat#real // type, number, release-time
, na = nat^N
, ra = real^N
, ra4 = real^N4

// function returns index corresponding to largest element (>0) in rest, b>0
func select(rest: ra, b: na) -> bool#nat =
|[ bestsofar: nat#real#bool, threshold: real, i: nat
 | threshold:= 0.3
 ; bestsofar:= <0, threshold, false>; i:= 0
 ; *[ i < N
    -> [ b.i  > 0 ->
           [rest.i > bestsofar.1 -> bestsofar:= <i,rest.i,true>
           |rest.i <=bestsofar.1 -> skip
           ]
       | b.i <= 0 -> skip
       ]
     ; i:= i + 1
    ]
 ; ret <bestsofar.2, bestsofar.0>
]|

// function intialises natural arrays (buffer contents) at n
func initna(n: nat) -> na =
|[ i: nat, a: na
 | i:= 0
 ; *[ i < N -> a.i:= n; i:= i + 1 ]
 ; ret a
]|

// function intialises real arrays (size N) at n
func initra(n: real) -> ra =
|[ i: nat, a: ra
 | i:= 0
 ; *[ i < N -> a.i:= n; i:= i + 1 ]
 ; ret a
]|

// function intialises real arrays (size 4*N) at n
func initra4(n: real) -> ra4 =
|[ i: nat, a: ra4
 | i:= 0
 ; *[ i < N4 -> a.i:= n; i:= i + 1 ]
 ; ret a
]|

// function multiplies real array elements with (real) constant
func multra(p: ra, n: real) -> ra =
|[ i: nat, a: ra
 | i:= 0
 ; *[ i < N -> a.i:=  n * p.i ; i:= i + 1 ]
 ; ret a
]|

// function takes the sum of elements of nat array
```

```
func sumna(a: na) -> nat =
|[ i: nat, s: nat
 | i:= 0; s:= 0
 ; *[ i < N -> s:= s + a.i; i:= i + 1 ]
 ; ret s
]|

// function takes the sum of elements of real array
func sumra(a: ra) -> real =
|[ i: nat, s: real
 | i:= 0; s:= 0.0
 ; *[ i < N -> s:= s + a.i; i:= i + 1 ]
 ; ret s
]|

func minusra(p1,p2: ra) -> ra =
|[ i: nat, a: ra
 | i:= 0
 ; *[ i < N -> a.i:=  p1.i - p2.i; i:= i + 1 ]
 ; ret a
]|

func plusra(p1,p2: ra) -> ra =
|[ i: nat, a: ra
 | i:= 0
 ; *[ i < N -> a.i:=  p1.i + p2.i; i:= i + 1 ]
 ; ret a
]|

// function combines an array of m na's into 1 ra4
func na2ra4(na4: na^m) -> ra4 =
|[ yra4: ra4, i, j, index: nat
 | i:= 0; j:= 0
 ; *[ j < m
     -> *[ i < N
           -> index:= j*N + i
            ; yra4.index:= n2r(na4.j.i)
            ; i:= i + 1
        ]
      ; j:= j + 1
      ; i:= 0
   ]
 ; ret yra4
]|

//function splits an array of m*N reals (ra4) into an array of m arrays of N reals
func ra42ra(xra4: ra4) -> ra^m =
|[ y4ra: ra^m, ratemp: ra, i, j, index: nat
 | j:= 0; i:= 0
 ; *[ j < m
     -> *[ i < N -> index:= j*N + i
                  ; ratemp.i:= xra4.index
                  ; i:= i + 1
        ]
      ; y4ra.j:= ratemp
      ; j:= j + 1
      ; i:= 0
   ]
 ; ret y4ra
]|

// function generating real array with zeros, except for 1.0 for element n
func event2I(n: nat) -> ra =
|[ a: ra
 | a:= initra(0.0)
 ; a.n:= 1.0
 ; ret a
]|

func saturate(y: ra4, llim, ulim: real) -> ra4 =
|[ ysat: ra4, i: nat
 | i:= 0
 ; *[ i < N4 -> ysat.i:= min(max(y.i,llim),ulim); i:= i + 1 ]
 ; ret ysat
```

```
]|

func plotna(r: na) -> bool =
|[ i: nat
 | *[ i < N-1 -> !r.i,"\t"; i:= i + 1 ]
 ; !r.i
 ; ret true
]|

func plotra(r: ra) -> bool =
|[ i: nat
 | *[ i < N-1 -> !r.i,"\t"; i:= i + 1 ]
 ; !r.i
 ; ret true
]|

// generator: equals machine with only one channel (out) and no service times
proc G(b: (!lot)^N, c: ?nat)=
|[ r,nr: nat
 | nr:= 0
 ; *[ true -> c?r; b.r!<r,nr,time>; nr:= nr + 1 ]
]|

// machine
proc M(a: (?lot)^N, b: (!lot)^N, c: ?nat, kd, kp: real)=
|[ x: lot, r: nat, te: (-> real)^2
 | te.0:= exponential((kd + 1)*kp/(kd*kp + 1))  // parameter: mean te
 ; te.1:= exponential((kd + 1)/(kd*kp + 1))
 ; *[ true
    -> c?r
     ; a.r?x
     ; delta sample te.(2*r div N)
     ; b.r!x
    ]
]|

// buffer: sends 1 or 0 to controller (lot received/sent)
proc B(a: ?lot,b: !lot,c: !nat, bmax:nat)=
|[ xs: lot*,x: lot
 | xs:= []
 ; *[ len(xs) < bmax; a?x      -> xs:= xs++[x]; c!1
   | len(xs) > 0;    b!hd(xs) -> xs:= tl(xs); c!0
   ]
]|

// exit
proc E(a: (?lot)^N, c: !real, tse: nat)=
|[ x: lot, ft, mft: real, nr: nat
 | mft:= 0.0; nr:= 0
 ; *[i: nat <- 0..N: a.i?x
     -> [ x.1 >  tse -> ft:=time - x.2
                      ; nr:= nr + 1
                      ; mft:= (ft + mft*(nr-1))/nr
        | x.1 <= tse -> skip
        ]
    | true; c!mft -> skip
    ]
]|

// filter: used by controller in continuous-discrete transformation
proc F(a: ?ra, b: !ra, lambda: real)=
|[ xrest,yrest: ra
 | [ abs(lambda - 0.5) >  0.5 -> !"alpha incorrect (0 < alpha < 1)\n"; terminate
   | abs(lambda - 0.5) <= 0.5 -> skip
   ]
 ; *[ true -> a?xrest; yrest:= multra(xrest,lambda); b!yrest ]
]|

// controller
proc C(a: (!nat)^m, b: ((?nat)^N)^m, af: (!ra)^4, bf: (?ra)^4, d: ?real,
       ts, tend, llim, ulim, kd, wu, ut, wref: real, h: nat) =
|[ j, n, nr: nat, avgWIP, tsample, mft: real, MLrunning, foo: bool, nextlot: (bool#nat)^m
 , bb: na^mb, cprod: na, e, u, ul: ra^m, yc: ra4, midle: bool^m
 | yc:= initra4(0.0)
```

```
; j:= 0
; mft:= 0.0
; *[ j < m
    -> e.j:= initra(0.0)
     ; nextlot.j:= <false, 0>
     ; midle.j:= true
     ; bb.(j+1):= initna(0)
     ; j:= j + 1
   ]
; bb.0:=initna(1); cprod:= initna(0)
; avgWIP:= 0.0; tsample:= 0.0; nr:= 0
; MLrunning:= initmatlab(kd,h,wu,ut,wref)
; *[ k: nat <- 0..m: nextlot.k.0 and midle.k; a.k!nextlot.k.1
    -> midle.k:= false
   | k: nat <- 0..m, i: nat <- 0..N: true; b.k.i?n
     -> [ n=0 -> bb.(k+1).i:=  bb.(k+1).i - 1
        | n=1 -> bb.(k+1).i:=  bb.(k+1).i + 1
                 ; [ k =  3 -> cprod.i:=  cprod.i + 1
                   | k /= 3 -> [ midle.(k+1)      -> nextlot.(k+1):=  select(e.(k+1),bb.(k+1))
                               | not(midle.(k+1)) -> skip
                               ]
                   ]
                 ; e.k:= minusra(e.k,event2I(nextlot.k.1))
                 ; midle.k:= true
                 ; nextlot.k:=  select(e.k,bb.k)
        ]
   | true; delta tsample - time
     -> tsample:=  time + ts
      ; nr:= nr + 1
      ; avgWIP:= (n2r(sumna(bb.1)+sumna(bb.2)+sumna(bb.3))+avgWIP * (nr-1))/nr
      ; j:= 0
      ; *[ j < m
         -> af.j!e.j; bf.j?e.j
          ; ul.j:= u.j
          ; j:= j + 1
         ]
      ; yc:= controlaction(na2ra4(<bb.1,bb.2,bb.3,cprod>))
      ; u:= ra42ra(saturate(yc,llim,ulim))
      ; j:= 0
      ; *[ j < m
         -> e.j:= plusra(e.j,multra(minusra(multra(u.j,3.0),ul.j),0.5*ts))
          ; nextlot.j:=  select(e.j,bb.j)
          ; j:= j + 1
         ]
      ; [ time >= tend -> MLrunning:= closematlab()
                        ; d?mft
                        ; !avgWIP,"\t",mft*sumna(cprod)/time,"\t",sumna(cprod)/time,"\n"
                        ; terminate
        | time <  tend -> skip
        ]
   ]
]|

syst LINE(kd, kp, alpha, tend: real, bm, hor: nat, weightu, ut, wipref: real)=
|[ gb1, b1m1, m1b2, b2m2, m2b3, b3m3, m3b4, b4e: (-lot)^N
 , cm: (-nat)^m, bc: ((-nat)^N)^m, cf, fc: (-ra)^4, ec: -real
 |                G(gb1,cm.0)
|| i: nat <- 0..N: B(gb1.i,b1m1.i,bc.0.i,bm)
||                M(b1m1,m1b2,cm.1, kd, kp)
|| i: nat <- 0..N: B(m1b2.i,b2m2.i,bc.1.i,bm)
||                M(b2m2,m2b3,cm.2, kd, kp)
|| i: nat <- 0..N: B(m2b3.i,b3m3.i,bc.2.i,bm)
||                M(b3m3,m3b4,cm.3, kd, kp)
|| i: nat <- 0..N: B(m3b4.i,b4e.i,bc.3.i,1000)
||                E(b4e, ec, 2000)
||                C(cm, bc, cf, fc, ec, 10.0, tend, -5.0, 5.0, kd, weightu, ut, wipref, hor)
|| i: nat <- 0..m: F(cf.i,fc.i, alpha)
]|

xper(kd, kp, alpha, throughput, wipref: real, bufsize, hor: nat, weightu, tend: real) =
|[ LINE(kd, kp, alpha, tend, bufsize, hor, weightu, throughput, wipref) ]|
```