

MASTER

Feedback control of a 2-product workstation with setups and one piecewise constant arrival rate

Smolders, A.C.J. (Roy)

Award date:
2007

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Feedback control of a 2-product workstation with setups and one piecewise constant arrival rate

A.C.J. Smolders

SE 420509

Master's thesis

Supervisor: Prof.dr.ir J.E. Rooda

Coaches: Dr.ir. A.A.J. Lefeber

Ir. J.A.W.M. van Eekelen

EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
SYSTEMS ENGINEERING GROUP

Eindhoven, March 2007

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

March 2006

Student Ing. A.C.J. Smolders
Supervisor Prof.dr.ir. J.E. Rooda
Advisors Dr.ir. A.A.J. Lefeber
 Ir. J.A.W.M. van Eekelen
Start March 2006
Finish March 2007
Title Feedback control of 2-product workstation with setups and piecewise-constant arrival rates

Subject

Flexible manufacturing systems can be considered as a network of workstations which serve different types of jobs. Switching between these types of jobs often requires a setup. Having non-negligible setup times leads to serial batching and results in a piecewise-constant flow of jobs leaving the workstation. In a network setting all workstations receive jobs at a piecewise-constant rate. To control such a system, we are interested in an optimal steady state process cycle for the system. Next, we would like to derive a feedback control that makes the system behavior converge towards this optimal steady state process cycle.

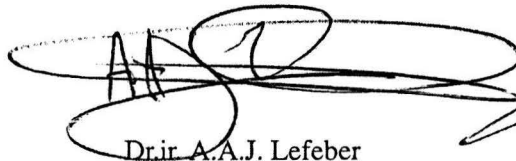
Assignment

To control a network of workstations (or servers), a new method for controlling switched linear systems is proposed in the paper "Feedback control of 2-product server with setups and bounded buffers" by Van Eekelen, Lefeber and Rooda. In this paper the basic ideas of how to design a controller are illustrated by considering a single workstation with constant arrival rates. As mentioned above, in a network of workstations which processes different types of jobs, a piecewise-constant arrival rate of jobs at a workstation is inevitable. To obtain better insight into the phenomena within such a network of workstations, the same basic ideas can be applied. Consider the smallest system possible: a single workstation which serves two different types of jobs. Extend the results presented in the above mentioned paper to the setting of piecewise-constant arrival rates. First determine an optimal steady state process cycle. Second, derive a feedback controller which steers the system towards the desired behavior.

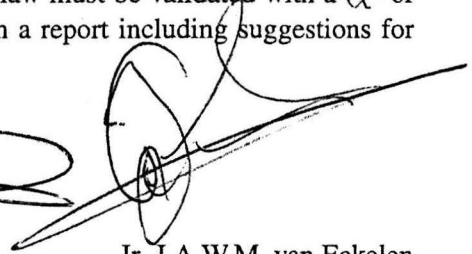
First, a literature review is required. Next, an optimal process cycle and feedback law must be looked for. Furthermore, the derived optimal process cycle and feedback law must be validated with a (χ - or Matlab-)model where useful. Finally, results must be presented in a report including suggestions for future work.



Prof.dr.ir. J.E. Rooda



Dr.ir. A.A.J. Lefeber



Ir. J.A.W.M. van Eekelen

Systems
Engineering



Department of Mechanical Engineering

Preface

Writing this preface makes me realize I almost finished my final thesis together with a five years curriculum in Mechanical Engineering.

I studied Mechanical Engineering at a School for Higher Vocational Education where I received my degree as a Bachelor in Mechanical Engineering in 2000. After graduation I was not satisfied with the knowledge a gained. I wanted to learn more about the wonders of Mechanical Engineering.

This curriculum started for me in August 2000. After a first week of partying and sleep as little as possible, my life as a student at the TU/e started. A shorter program was attended to finish my curriculum due to my earlier obtained degree. I started with second year courses but had to conclude too many wonders reached me too fast. In January 2001 I decided to quit the short program and pass through the regular curriculum instead.

After settling in Eindhoven during this period, it was hard to find a good harmony between study, friends, work and sports. But in the following years I really enjoyed the different wonders of Mechanical Engineering combined with a very nice student life. One of these wonders was the Analysis of Manufacturing Systems. I wanted to explore this area more thoroughly and decided to enter the Section Systems Engineering supervised by prof. Rooda. I enjoyed performing the courses and the final project I am about to conclude.

Many thanks go to my parents. They supported me during all these years. Also I would like to use this opportunity, to thank all people who made my student life a time never to forget. A well deserved thanks goes to Jelmer and Sebastiaan, with these guys I finished a lot of courses with satisfying results. I would like to thank Erjen Lefeber and Joost van Eekelen for their good coaching and prof. Rooda for his supervising. An additional word of gratitude goes to the students in the SE-lab and the PhD students who provided a pleasant working environment. Last but not least, I would like to thank Dirk for his support and wise twaddle.

Roy Smolders - acjismolders@gmail.com
Eindhoven, March 2007

Summary

The use of flexible manufacturing systems in production facilities has a large advantage. Different products can be manufactured with a limited number of machines. In order to have a good customer response the flow time must be short, what means the work in process (wip) levels have to be low and scheduling of tasks within the system becomes important. In the literature, models of queueing networks and fluid approximation models are used to analyse the optimal wip level within a given scheduling policy for a flexible manufacturing machine. In [Eek06a] an approach is discussed which does the opposite. First it determines the optimal wip level and second a controller is proposed which steers the system to this optimum. This approach finds, as long as the total utilization of the system is less than one, a minimal weighted time averaged wip level for a manufacturing machine with setups and processing two product types arriving at constant rates. Having non-negligible setup times leads to serial batching and results in a piecewise constant departure rate for each type of product. A second machine in series receives these products at a piecewise constant rate.

This study uses the approach of [Eek06a] to determine an optimal wip level, but with one constant arrival rate and one piecewise constant arrival rate. Two important conditions have to be met before applying the theory in this report. The first condition is a sum of time fractions that each product needs to be processed is less than one and second, the length of one process cycle has to match the periodic behavior of the piecewise constant arrival pattern. The length of the periodic behavior also has a lower bound which has to be satisfied. If these conditions are satisfied, a solution in finding a minimal weighted time averaged wip level is guaranteed. Different optimal process cycles are obtained. The computation of optimal process cycle depends on the relationship between the maximum arrival rate of the piecewise constant arrival pattern and the maximum process rate of type same product type. Two situations are possible after determination of the cycle. These situations depend on the same relationship between the maximum arrival rate and the maximum process rate. In the first situation, the maximum arrival rate is larger than the maximum process rate. In the second situation, where the maximum arrival rate is less than the maximum process rate, a distinction is made. The distinction is based on the slow mode(s) (time intervals where the actual process rate is lowered to the arrival rate) that occur in the optimal process cycle. Eventually all possible trajectories of the process cycles are divided in three situations. For each of these three situations a feedback control law is proposed which steers the

system to its specific optimal process cycle. An analytical proof is presented which shows that the controller always steers the system to its desired trajectory. These controllers have been tested in a simulation study. In this study the workstation is simulated by means of a hybrid fluid model and by means of a discrete event model. The system starts with arbitrary buffer levels, an arbitrary machine state and an arbitrary point in the cyclic piecewise constant arrival pattern. From this starting point a controller has to steer the system to the optimal process cycle. All simulation results confirm convergence of the system to the desired trajectories. The controllers are very robust. If disturbances occur but the parameter setting stays the same, the same controller always steers the system back to the desired trajectory. So as long as the system meets the two conditions mentioned above, a robust feedback control law steers the system to a process cycle with a minimal weighted time averaged wip level and keeps it there.

Summary (Dutch)

Het gebruik van flexibele productie systemen in fabricage systemen heeft een groot voordeel. Een beperkt aantal machines is nodig voor het produceren van verschillende producten. Om accuraat te kunnen handelen naar de wensen van de klant moeten doorlooptijden kort zijn. Dit betekent dat de hoeveelheid onderhanden werk in het systeem laag moet worden gehouden en het belangrijk wordt orders te gaan plannen. In de literatuur zijn wachtrij theorieën en vloeistof modellen gevonden voor flexibele productie systemen. De theorieën geven een analyse van de hoeveelheid onderhanden werk nadat een planningsstrategie is toegepast. In [Eek06a] is de aanpak juist andersom. Eerst wordt de hoeveelheid onderhanden werk geoptimaliseerd en daarna wordt een regelaar voorgesteld die het systeem naar dit optimum toe dirigeert. De aanpak is gehanteerd op een machine met omsteltijden en welke twee typen producten produceert die met een constante aankomst arriveren. Daarbij moet gelden dat de bezettingsgraad van de machine kleiner blijft dan één. De niet te verwaarlozen omsteltijden zorgen voor het produceren van dezelfde producten in serie. Dit resulteert in een stuksgewijs constant patroon van producten die de machine verlaten. Als twee machines in serie staan ontvangt een tweede machine deze producten met een stuksgewijs constant patroon.

Deze studie gebruikt de aanpak uit [Eek06a] om de optimale hoeveelheid onderhanden werk te bepalen voor een systeem met een stuksgewijs constant aankomstpatroon en een patroon met constante aankomst. Om de minimale hoeveelheid onderhanden werk te vinden, zijn eerst de systeemeigenschappen en de aankomstpatronen gedefinieerd. De theorie in dit verslag kan worden gebruikt als aan twee condities wordt voldaan. Ten eerste moet de totale bezettingsgraad kleiner zijn dan één en ten tweede, de lengte van één proces cyclus moet even lang zijn als het periodieke gedrag van het stuksgewijze constante aankomstpatroon. Tevens moet het periodieke gedrag ook voldoen aan een ondergrens. Als aan deze condities wordt voldaan, kan een gewogen gemiddelde hoeveelheid onderhanden werk over tijd worden bepaald. Hierbij zijn verschillende optimale proces cycli gevonden. De berekening van een optimal proces cyclus is afhankelijk van de relatie tussen de maximale aankomstsnelheid en de maximale productiesnelheid van hetzelfde type product. Na het bepalen van een cyclus zijn er twee situaties denkbaar. In de eerste situatie is de maximum aankomstsnelheid groter dan de maximale productiesnelheid. In het tweede geval, waar de maximale aankomstsnelheid lager is dan de productiesnelheid, is een onderverdeling gemaakt. Deze is gebaseerd op welke producten worden geproduceerd met een gematigde snelheid (de actuele productiesnelheid

verlaagd naar de aankomstsnelheid) gedurende de optimale cyclus. Uiteindelijk zijn alle mogelijke trajectories onderverdeeld in drie situaties. Voor alle drie de situaties is een regelaar met terugkoppeling voorgesteld, welke het systeem naar de optimale cyclus toe moet dirigeren. Met een analytisch convergentiebewijs is bewezen dat de regelaar het systeem altijd naar de bijbehorende gewenste trajectorie zal regelen. De regelaars zijn getest aan de hand van simulaties. Het systeem is gesimuleerd met een vloeistof model en met een model in een discrete event omgeving. Het systeem start met willekeurige buffer groottes, een willekeurige toestand waarin de machine verkeert en op een willekeurig punt ergens op het cyclisch gedrag van het stuksgewijze aankomstpatroon van de machine. Vanuit dit startpunt moet de regelaar het systeem naar de optimale cyclus sturen. Alle simulatie resultaten bevestigen convergentie van het systeem naar de gewenste trajectories. De regelaars zijn zeer robust. Als er verstoringen optreden waarbij de systeem parameters niet veranderen, zal de regelaar het systeem altijd terug regelen naar de gewenste cyclus. Kortom, zolang het systeem voldoet aan de twee eerdergenoemde condities, zal een robuuste regelwet met terugkoppeling het systeem altijd dirigeren naar de proces cyclus met daarin het minimale gewogen gemiddelde van de hoeveelheid onderhanden werk over tijd. Tevens weet de regelaar het systeem ook op deze optimale trajectorie te houden.

Contents

Assignment	i
Preface	iii
Summary	v
Summary (Dutch)	vii
1 Introduction	1
2 Analysis of flexible manufacturing systems	3
3 System specifications of a two product workstation with setups	9
3.1 Characteristics	9
3.2 State, input and dynamics	13
4 Optimal process cycle	19
4.1 General analysis of the process cycle	19
4.2 Analysis of separate buffer levels	21
4.2.1 Analysis of the buffer level with a constant arrival rate.	21
4.2.2 Analysis of the buffer level with a piecewise constant arrival rate.	24
4.3 Optimal steady state process cycle	30
4.3.1 Situation I ($\hat{\lambda}_1 \geq \mu_1$)	30
4.3.2 Situation II ($\hat{\lambda}_1 < \mu_1$)	31
4.4 Optimal steady state trajectories	33

4.4.1	Trajectory of situation I	34
4.4.2	Trajectories of situation II	35
5	Feedback control	43
5.1	Feedback control of situation I ($\hat{\lambda}_1 \geq \mu_1$)	44
5.2	Feedback control of situation II-a ($\hat{\lambda}_1 < \mu_1$)	47
5.3	Feedback control of situation II-b ($\hat{\lambda}_1 < \mu_1$)	51
6	Simulation experiments	55
6.1	Simulation models	55
6.1.1	Hybrid fluid model	55
6.1.2	Discrete event model	56
6.2	Simulation results of the controller for situation I	59
6.3	Simulation results of the controller for situation II-a	62
6.4	Simulation results of the controller for situation II-b	65
7	Conclusions and recommendations	69
7.1	Conclusions	69
7.2	Recommendations	70
	Bibliography	73
	A Fluid models	75
	B Discrete event models	85

Chapter 1

Introduction

Manufacturing companies want to satisfy as much costumers as possible. Each costumer has its own demands. Different demands may lead to different types of products. In order to prevent a large machinery, manufacturers want to process different types of products with a single machine. Before such a machine can process another type of product, it usually requires a setup. By processing different products in a network of flexible manufacturing machines, a manufacturer can use a limited number of machines which can process at a relatively high degree of capacity utilization. A disadvantage of processing multiple products at a flexible manufacturing system (FMS) is the complexity of scheduling. The difficulties lie in the different nonconstant flows between different workstations. If machines process all products of each product type one by one, almost every machine copes with a piecewise- and/or constant arrival rate. To control all product flows in the network, a global controller can schedule all tasks that need to be performed and send these tasks to each machine. A manufacturer can apply different optimization criteria in controlling the workstations within his network. Besides the criterion of a high throughput, manufacturers want a short flow time for fast customer response. This can be established with minimized work in process levels.

To get good insights of how such a controller must operate and steers a machine, the focus of this research is finding a suitable controller for a single workstation which operates in a network of flexible manufacturing systems.

Objective

To obtain a better insight into the phenomena within a network of workstations, the smallest network possible is considered. The goal is to analyze one workstation with setups that processes two types of products/jobs. One type of product arrives with a piecewise constant arrival pattern and the other product type arrives with a constant arrival pattern. For this system the optimization criterion is to find an optimal trajectory where the costs of products residing at one workstation are minimized. Next, a

feedback controller has to be obtained which steers the system to this optimal trajectory. Convergence to the desired trajectory must be proven analytically. In different simulation experiments the proposed controllers have to be tested. These tests show that the controller works in the setting of a hybrid fluid model and in a discrete event setting as well. If the initial conditions of both experiments is the same, the convergence of the system to the desired trajectory must be the same. Finally a conclusion has to be drawn about the optimal trajectories and proposed controllers and the recommendations must be discussed.

Valorization

For companies that process a lot of different products in a network of flexible manufacturing machines this research can be very valuable. The use of this theory minimizes work in process levels, what results in lower flow times for better customer service. When this theory is applied, a manufacturer has less capital in its production process and is able to provide its customers with a better service. Eventually this can lead to a payoff for both manufacturer and consumers.

Another advantage of less work in process is the reduction of the probability products pass their due date. The feedstock is used more efficient what results in less residue and lower production costs. On moral grounds, consumers and governments support companies who process in an environment-friendly manner. By keeping the residues low, a company produces less waste and therefore reduces the chance to suffer its loss of face to the public.

Approach

A lot of research is performed with respect to servers through which different types of jobs flow. A flexible manufacturing system is an example of a network of servers. Other examples in these studies are data flows between computers, call-centers or an urban road network of crossings through which cars flow. In these studies different policies are used to optimize single systems and entire networks. Most theories optimize the system afterwards. In this study the system is optimized the other way around. First the desired (optimal) trajectory is determined and second the policy that converges the system to its desired behavior. This approach is applied in [Eek06a] also and therefore very useful. In Chapter 2 earlier performed research is described which results in the reason for the research objective and approach in this report. In Chapter 3 the properties of a two product workstation with accompanying product arrival patterns are discussed. Here the criteria of the system are established. If the system satisfies the conditions of Chapter 3, the optimal trajectory can be established in Chapter 4. When the trajectory is determined, a matching feedback controller is determined in Chapter 5. Simulations are performed to validate convergence of the controller to its desired trajectory. The simulations, presented in Chapter 6, contain continuous and discrete event simulations. Finally, the report ends with conclusions and recommendations for further research.

Chapter 2

Analysis of flexible manufacturing systems

A single flexible manufacturing machine or a network of flexible manufacturing machines are systems through which different types of jobs flow. Other kind of systems through which different types of jobs flows are: an urban road network of crossings with traffic lights through which cars flow, or a network of computers or telecommunication systems through which different data flows are present. To obtain a good overview of the different approaches that have been used to analyse these systems, this chapter discusses earlier performed researches.

In this study a server or workstation is considered with one constant arrival rate at one buffer and a second buffer where products arrive with a cyclic piecewise constant arrival rate. The scheduling of the system, i.e. when to process which product, is based on minimization of the time averaged weighted work-in-process (wip) level. First an optimal process cycle is obtained. Then a feedback law is proposed which steers the system to this optimal cycle. Within this study the assumption is made that the duration of one process cycle equals the length of the periodic behavior of the piecewise constant arrival pattern. Furthermore, the setup costs are not taken into account.

Queueing networks

As mentioned above, the system that is considered has an arrival rate which varies over time. The complexity of time-varying rate problems in queueing networks has resulted in less literature compared to equilibrium behavior of queues with constant arrival rates. Alnowibet and Perros [Aln06] and Massey [Mas02] regard the analysis of (tele-) communication models with time varying arrival rates in queueing networks. Massey analyzed different queueing theory models with time varying rates to maximize the profit of call centers for instance. The performance modeling of telecommunication

systems starts with an offered load model. An offered load describes the number of communication resources requested by arriving customers. The theory in combination with the system properties, the mean, variance, covariance and time lag between the peak arrivals and peak load for the system are measured. To deal with a system with finite resources, the offered load model is replaced with a loss queue model. The analysis of a loss queue model with time varying rates is performed with a pointwise stationary approximation (PSA) model (see [Gre91] and [Gre97]) and a modified offered load approximation (MOL) model (see [Jag75]). Depending on the shape of the time-varying arrival pattern, one of the approximation models provides the most accurate estimation of the performance of a system. In case of loss models the approximation models also estimate the probability of blocking. Blocking might occur due to the finite resources available. Alnowibet and Perros also analyze a nonstationary queueing network. The network contains multi-rate loss queues and population constraints, where the external arrival rates are a periodic function of time. The analysis is not based on PSA or MOL but based on the fixed-point approximation (FPA) method for a nonstationary queueing network with multi-rate loss queues. For further details on the FPA algorithm, the reader is referred to [Aln04]. The FPA algorithm calculates the mean number and the blocking probability of each class of product or customer in each queue, without the need to solve any differential equations. When the number of customers in a loss queue depends on the number of customers in other loss queues, customers may share common communication channels, what is known as a queueing network with population constraints. In case of the presence of population constraints, Alnowibet and Perros introduce an iterative approximation algorithm based on the FPA method. The result of the method is an approximate time-dependent blocking probability function for the system, obtained after a relative short CPU time in comparison with the simulations which are needed to obtain the same probability function.

Bekker et al. [Bek04] provides an analysis with a workload-dependent arrival rate in a single-server queue. The goal is to control the arrival of jobs to optimize server performance. Proportionality relations between the workload distribution of two queues with the same ratio of arrival rate and process rate are derived. With these relationships results of a whole class of models can be obtained from the analysis of one model.

All these approaches use time-dependent arrival rates, often represented as a Poisson distribution, but none of them uses piecewise constant arrival pattern as discussed at the start of this Chapter.

In this report the system is optimized to the time averaged weighted wip level. In literature several optimization problems are encountered for a queueing network. In Azaron et al. [Aza06] a multi-objective optimal control problem is developed. Azaron et al. use the longest path analysis in a queueing network. In the analysis the density functions are determined of the time spent in a service station and the queueing network is transformed into a stochastic network. Finally the distribution function is obtained of the longest path in the stochastic network. The distribution function is applied in the multi-objective flow time control problem which minimizes the average flow time, the variance of the flow time and the total costs of the system per period. To solve the multi-objective, nonlinear programming (NLP) problem, a variation of the

goal programming technique (see [Ign76]) is used. The downside of this theory is that the individual arrivals are independent Poisson processes with equal rates. So again no piecewise constant arrival pattern is included.

Fluid approximation models

Other studies performed the analysis of applications with multiple data flows with fluid approximation models. Ridley et al. [Rid03] proposed a fluid approximation model for a priority call center with time varying arrivals. The system has two customer classes, high priority calls and low priority calls (Figure 2.1). When a low priority call

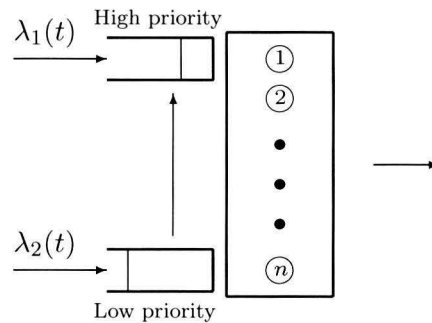


Figure 2.1: The two-customer class call center with n operators

is not completed within a given amount of time, the call switches from the low priority queue to the high priority queue. The arrival rate of the customers is exponentially distributed. With the model Ridley et al. estimate the mean number of costumers waiting. Eventually this number is used to estimate the overall staffing level (n). So the process rate of the server is adapted to the arrival rate. In this report this luxury is not available, the server has a fixed maximum capacity. Another fluid model is used by Lan and Olsen [Lan06]. They discuss a multi product, single server production system with setup times and costs. With a nonlinear programming model a lower bound on the long-run average production costs per unit time is established. Lan and Olsen show it is the lower bound of performance for a single stochastic server with Poisson arrivals. The theoretical lower bound can not always be reached. The lower bound is established for fluid systems. The more a deterministic system converges to a fluid system the closer the lower bound can be reached. Lan and Olsen provide heuristics for stochastic production systems in order to optimize the performance.

Deterministic systems

Except for the piecewise constant arrival pattern, Savkin and Somlo consider a similar type of problem for solving flexible manufacturing scheduling problems of a deterministic system [Som06]. They use a hybrid dynamical approach (HDA) to find time periods of periodic motions. The HDA is supported with tools provided by the qualitative theory of hybrid dynamical systems of Matveev and Savkin [Mat00]. The use of these tools makes the planning more effective and it widens the application field of HDA. The paper presents a model with the task to process different part types during a given time period on a fixed number of machine groups. The essence of the HDA is working on a part type and when the conditions to switch are reached, perform a setup and process another part until the next switch condition is reached. The hybrid dynamical approach is performed for a simple (two part types) problem and a complex (multi-part types, multi-machine groups). The results are periodic schedules which show performances close to the optimal. Although the theory provides schedules for the transient and periodic behavior, these schedules are predetermined and can not cope with disturbances within the system. Furthermore, the theory does not reduce the number of jobs in the system even when the buffers contain more jobs than necessary.

Most of the literature just mentioned, contains systems which are optimized after a given policy is applied. The most considered policies are: Clear-the-Largest-Buffer-Level (CLB) Policy, Clear-a-Fraction (CAF) Policies and Clear-the-Largest-Work (CLW) Policy [Per89]. In order to achieve a robust scheduling policy that can handle disturbances and is able to reach its desired trajectory at all times, first the optimal process cycle has to be obtained, and second a feedback control is derived which steers the system to the optimal process cycle. In [Lef06] this method is applied for a network of servers through which many types of jobs flow. In [Eek06a] this theory is applied for the smallest system possible: a single workstation which serves two types of jobs with type specific setup times. For the workstation an optimal process cycle is derived and a feedback law is proposed that steers the system to this optimal process cycle. In the optimal cycle 'slow modes' can occur (also referred to as 'idling' [Cha92] or 'cruising' [Lan06]). In a slow mode products are processed at a rate that equals the current arrival rate. The output of such a workstation is a piecewise constant departure rate. So a workstation behind this workstation receives products with a piecewise constant arrival rate. This report discusses the same workstation as described in [Eek06a] except it has one cyclic piecewise constant arrival rate.

Different approaches of solving a flexible manufacturing system have been presented. Each approach has its own advantages and disadvantages. The choice of approach for a system with one constant arrival rate at one buffer and a second buffer where products arrive with a piecewise constant arrival pattern is the same as used in [Lef06] and [Eek06a]. The approach holds to obtain first the optimal process cycle and second determine the feedback control for the optimal process cycle. This approach is used throughout the rest of this report. In the next chapter the system specifications of the

workstation are discussed. Afterwards the optimal process cycle, feedback control and simulations of the feedback control laws are discussed.

Chapter 3

System specifications of a two product workstation with setups

In a flexible manufacturing system, workstations are able to process different types of products. The way products are processed depends on the buffers and machines in a workstation, the arrival pattern of products and on the properties of the manufacturing machine itself. All these parts together form the system properties. With these properties preconditions are established. If the preconditions are met, the system can be optimized with respect to the time averaged weighted wip level and a way to control the system can be determined. To obtain these conditions the system properties have to be defined first. In this chapter the characteristics and dynamics of a workstation with two buffers are defined. First the characteristics of the system are discussed. Here the structure of the workstation, the arrival patterns and the time fractions that each product needs to be processed are considered. In the second section the dynamics of the workstation are discussed. With these dynamics come constraints that result in a system with a desired cyclic sequence.

3.1 Characteristics

In this section the characteristics of a manufacturing workstation with two buffers are presented. As mentioned in Chapter 1, the workstation serves two types of products where one product type arrives at a constant rate and one product type at a piecewise constant rate, the structure of the workstation is discussed first. Next, the arrival patterns and time fractions that each product needs to be processed in the system are discussed.

Structure of the workstation

When products (jobs) arrive at a workstation, these products can be of a different type. Each type is stored in a separate buffer. The capacity of the buffers are assumed infinite and work on parallel first-in-first-out (FIFO) basis. Products are processed like discrete events. Because it is easier to analyse a hybrid fluid model than a discrete event model of the workstation, this report discusses the analysis of a hybrid fluid model of the workstation. In Figure 3.1 such a workstation is presented. The number of products

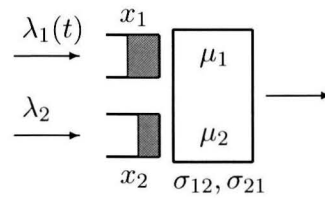


Figure 3.1: Manufacturing machine

in each buffer is denoted by x_i . The subscript i refers to the type of product. In this case two types of products arrive so: $i \in \{1, 2\}$. The workstation can process one product type at a time. This means the machine has to switch between both types to prevent large buffer levels. Switching between processing type 1 and 2, and vice versa, requires a setup with setup times of respectively: σ_{12} and σ_{21} hours. Without loss of generality, the time unit is set to 'hours' in this report. The arrival rate of products at the workstation is denoted with λ_1 and λ_2 (in products/hour). The rates can be time dependent. In this report one time dependent arrival rate $\lambda_1(t)$ for type 1 and a constant arrival rate λ_2 for type 2 is used. The workstation is able to process products of type 1 and 2 at rate between zero and a maximum rate of μ_1 and μ_2 products/hour respectively.

Using a hybrid fluid model instead of a discrete event model changes the character of the buffer levels. The buffer levels become: $x_i \in [0, N_i]$ where $i \in \{1, 2\}$ and N_i represents the maximum capacity of each buffer. The special situation with infinite buffer levels ($N_i = \infty$) is assumed in the rest of this report.

The parameters presented in Figure 3.1 are all parameters of the system. This includes the machine parameters, the buffer levels and the properties of the arrival patterns. Different aspects of the system and its parameters are discussed in the next parts.

Piecewise constant arrival pattern

As mentioned in the previous part, $\lambda_1(t)$ is time dependent. In a network of two product workstations, switching between the two product types is necessary to avoid very large buffer levels in one of the buffers. If the workstation has non-negligible setup times this

leads to serial batching, what results in a piecewise constant flow of one type of products leaving the workstation. For workstations behind this first workstation, products of each type arrive at a piecewise constant pattern. In this report the piecewise constant arrival pattern is assumed cyclic. The result of such arrival pattern takes on two values for $\lambda_1(t)$: 0 or $\hat{\lambda}_1$. The pattern is presented in Figure 3.2. The upper part of Figure 3.2

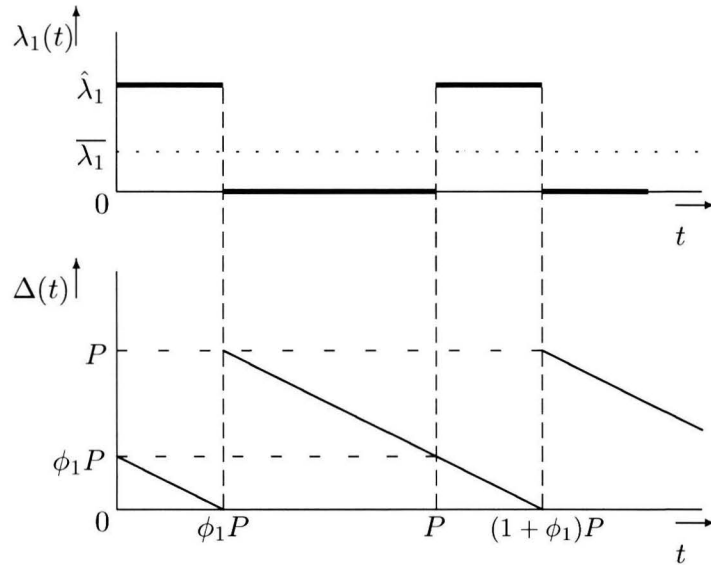


Figure 3.2: Definition of $\lambda_1(t)$ and $\Delta(t)$

presents the arrival pattern of type 1. The piecewise constant arrival pattern repeats itself after a period. Each period has a length denoted with P hours and a mean arrival rate denoted with $\bar{\lambda}_1$ products/hour, which is computed with:

$$\bar{\lambda}_1 = \frac{1}{P} \int_0^P \lambda_1(t) dt.$$

Buffer 1 receives products when $\lambda_1 = \hat{\lambda}_1$ where $\hat{\lambda}_1$ represents the rate of arrivals. The time span during which these products arrive is a fraction of one period. The time fraction is denoted as $0 < \phi_1 \leq 1$. This results in a mean arrival rate of:

$$\frac{1}{P} \int_0^P \lambda_1(t) dt = \frac{1}{P} \left(\int_0^{\phi_1 P} \hat{\lambda}_1 dt + \int_{\phi_1 P}^P 0 dt \right) = \frac{1}{P} (\phi_1 P \hat{\lambda}_1),$$

what results in:

$$\bar{\lambda}_1 = \phi_1 \hat{\lambda}_1. \quad (3.1)$$

Remark 3.1.1. Parameter $\phi_1 > 0$, otherwise no products arrive during a period and the machine has only 1 type of product to process. If $\phi_1 = 1$ the arrival pattern becomes constant. The case $\phi_1 = 1$ has been dealt with in [Eek06a].

The approach in [Eek06a] is used as a guide line for this report. Therefore, the appearance of slow modes is very common.

Definition 3.1.2. (Slow mode). A slow mode occurs when the machine processes a type of product at a rate equal to the arrival rate, under the condition that the arrival rate is less than the maximum process rate for that specific type of product.

Remark 3.1.3. If a slow mode appears in the first workstation, the arrival pattern, as presented in Figure 3.2, changes at the second workstation. In the rest of the analysis in this report this behavior is excluded, but it is discussed in Chapter 7.

In the remainder let Δ denote the remaining time until the piecewise constant arrival rate is turned off again. The resulting evolution of $\Delta(t)$ is presented in the lower part of Figure 3.2.

Process times of the workstation

The production capacity of a system has to be sufficient to process all products that arrive at the system. If the capacity is insufficient buffer levels increase in time. In general increasing buffer levels are undesirable. For that reason also a workstation which processes two types of jobs and performs setups must be stable. In order to obtain a stable system, the machine needs the ability to process as many products as products that arrive during a specific time span. The length of such specific time span is discussed in another part of this chapter. Each type of product needs a specific time fraction of a specific time span. To obtain a stable system, the sum of time fractions for a workstation without setup times is less than or equal to 1. For a 2-product workstation with setups, the sum of time fractions of each producttype has to be strictly less than 1. If the sum equals or becomes larger than 1, the system becomes unstable because there is no time to process products that arrive during a setup. An unstable workstation reveals itself by exploding buffer levels when time goes to infinity. To obtain the sum of time fractions of each product that needs to be processed, the time fractions of the separate types of products must be determined. These individual time fractions are defined as

$$\bar{\rho}_i = \frac{\bar{\lambda}_i}{\mu_i} \quad \text{where :} \quad i \in \{1, 2\}. \quad (3.2)$$

One of the arrival rates is time dependent, therefore equation 3.2 uses mean arrival rates ($\bar{\lambda}_i$) what results in time dependent time fractions ($\bar{\rho}_i$). If an arrival pattern is time independent then $\lambda_i \equiv \bar{\lambda}_i$ and $\rho_i \equiv \bar{\rho}_i$. The sum of time fractions that each product needs to be processed is indicated by ρ :

$$\rho = \sum_{i=1}^2 \bar{\rho}_i < 1. \quad (3.3)$$

At this point, all characteristics of the workstation are defined. In the next section these parameters are used to describe the dynamics of the system. At a later stage, the workstation has to be controlled. A controller imposes different tasks on the machine. The controller sends an input signal to the machine, in which it tells the machine what to do. The decision making is done by the controller. To make these decisions properly, the dynamics of the system has to be clear.

3.2 State, input and dynamics

The system, as described in previous section has to process different product types and has to perform setups. An input signal tells the machine which task has to be performed. In this section the state and input vector of the machine are explained. The state and input signals are used to describe the dynamics of the system. In the last part of this section the desired periodic behavior of a process cycle is defined.

The state of the system consists of different elements. Besides the buffer levels x_1 and x_2 and the value of Δ , also a remaining setup time x_0 and the mode m are present. The workstation has two modes denoted by $m \in \{1, 2\}$. The mode presents which type of product the machine is currently either serving or being setup for. These five variables determine the state of the system at time t :

$$\vec{x}(t) = [x_1(t) \quad x_2(t) \quad x_0(t) \quad \Delta(t) \quad m(t)]^T \in \mathbb{R}_+^4 \times \{1, 2\}. \quad (3.4)$$

The machine can process products of type 1 or 2, perform a setup or become idle. A controller which supervises the system determines, based on the state of the system, an input vector for the machine. The input vector consists of three signals. Signal u_1 and u_2 represent the rate at which the machine has to process. These input signals must be less than or equal to maximum process rate μ_1 and μ_2 respectively. The third input signal u_0 determines which activity must be performed. The possible activities are:

- $u_0 = \mathbf{1}$: setup to type 1
- $u_0 = \mathbf{1}$: process type 1
- $u_0 = \mathbf{2}$: setup to type 2
- $u_0 = \mathbf{2}$: process type 2

Also 'idling' forms an activity. This activity can be performed in combinations with other parameters of the input vector. The input vector of the machine is:

$$\vec{u}(t) = [u_0(t) \quad u_1(t) \quad u_2(t)]^T \in \{\mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{2}\} \times \mathbb{R}_+^2. \quad (3.5)$$

With the input vector and state of the system, the hybrid dynamics of the system are described in two parts. The first part contains the discrete event dynamics of the

system:

$$\begin{aligned}
x_0 &:= \sigma_{21}, \quad m := 1 \quad \text{if } u_0 = \mathbf{1}, \quad m = 2. \\
x_0 &:= \sigma_{12}, \quad m := 2 \quad \text{if } u_0 = \mathbf{2}, \quad m = 1. \\
\Delta &:= P \quad \quad \quad \text{if } \Delta = 0.
\end{aligned} \tag{3.6}$$

Jumps of variables x_0 and m can occur when the input signal u_0 changes. In that case, both the mode and the remaining setup change. Besides the possible jumps of variables when $u_0 \in \{\mathbf{1}, \mathbf{2}\}$, variable Δ always shows jumps. Variable Δ decreases when time goes by. If $\Delta = 0$ it is reset to $\Delta = P$. The second part of the hybrid dynamics contains the continuous dynamics.

$$\begin{aligned}
\dot{x}_0(t) &= \begin{cases} -1 & \text{for } u_0(t) \in \{\mathbf{1}, \mathbf{2}\} \\ 0 & \text{for } u_0(t) \in \{\mathbf{1}, \mathbf{2}\} \end{cases} \\
\dot{x}_1(t) &= \lambda_1(t) - u_1(t) \\
\dot{x}_2(t) &= \lambda_2 - u_2(t) \\
\dot{\Delta}(t) &= -1.
\end{aligned} \tag{3.7}$$

Furthermore, at each time instant the input is subject to the constraints:

$$\begin{aligned}
u_0 &\in \{\mathbf{1}, \mathbf{2}\}, \quad u_1 = 0, \quad u_2 = 0 && \text{for } x_0 > 0 \\
u_0 &\in \{\mathbf{1}, \mathbf{2}\}, \quad 0 \leq u_1 \leq \mu_1, \quad u_2 = 0 && \text{for } x_0 = 0, \quad x_1 > 0, \quad m = 1 \\
u_0 &\in \{\mathbf{1}, \mathbf{2}\}, \quad 0 \leq u_1 \leq \lambda_1(t), \quad u_2 = 0 && \text{for } x_0 = 0, \quad x_1 = 0, \quad m = 1 \\
u_0 &\in \{\mathbf{1}, \mathbf{2}\}, \quad u_1 = 0, \quad 0 \leq u_2 \leq \mu_2 && \text{for } x_0 = 0, \quad x_2 > 0, \quad m = 2 \\
u_0 &\in \{\mathbf{1}, \mathbf{2}\}, \quad u_1 = 0, \quad 0 \leq u_2 \leq \lambda_2 && \text{for } x_0 = 0, \quad x_2 = 0, \quad m = 2.
\end{aligned}$$

The input signal u_0 contains 2 activities. These are the activities that can be performed. Given the current state the first constraint represents the situation where a setup is being performed. As long as $x_0 > 0$ no products can be processed ($u_1 = 0$; $u_2 = 0$). During the setup, an intervention may cause a switch to an other product type. In that case the ongoing setup is interrupted and the new setup starts. The second and third constraint represent the processing of type 1. As long as the buffer is not empty, the machine meets the second constraint and processes at its maximum process rate. Else the third constraint holds and its maximum process rate decreases to the rate equal to the arrival rate. During processing type 1 the conditions must hold that the machine does not process type 2 ($u_2 = 0$). The last two constraints represent the situations where type 2 is processed. The difference between these two constraints is the maximum process rate. If the buffer level is larger than zero the rate can be up to μ_2 , if the buffer is empty the maximum process rate is equal to the arrival rate λ_2 . When processing type 2, type 1 can not be processed ($u_1 = 0$).

The sequence of the four tasks can be performed in the following order only:

$$\dots \rightarrow \text{process type 1} \rightarrow \text{setup to type 2} \rightarrow \text{process type 2} \rightarrow \text{setup to type 1} \rightarrow \dots$$

In the remainder of this report this cyclic sequence is called the process cycle. In the next part the length of the process cycle is defined.

Steady state process cycles

This report studies steady state process cycles. Steady state means that the variables of a dynamical system describing its behavior are periodic functions of time or constant. So a steady state process cycle shows periodic behavior. The periodic behavior manifests itself in cyclic behavior of the buffer levels. Finding a steady state process cycle of a two product workstation with setups is only possible if the sum of time fractions that each product needs to be processed in the system is less than 1 (3.3).

During one process cycle, the mean arrival rate of a type is $\bar{\lambda}_i$. The machine can process the products that arrive during one process cycle at a maximum rate of μ_i . To obtain a steady state process cycle, the machine has to process:

$$\frac{\bar{\lambda}_i}{\mu_i} T = \bar{\rho}_i T \quad [\text{hours}] \quad i \in \{1, 2\},$$

of each product type during one process cycle of T hours. The workstation processes the same number of products as the number that arrive during one cycle. So to make sure the system has a steady state process cycle, the system needs enough time to process type 1, type 2, a setup to 1 and a setup to 2:

$$\begin{aligned} T &\geq \bar{\rho}_1 T + \rho_2 T + \sigma_{12} + \sigma_{21} \\ \text{or:} & \\ T &\geq \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}. \end{aligned} \tag{3.8}$$

By using (3.3), the minimum length of the process cycle is established:

$$T_{\min} = \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}.$$

Two variables in time are discussed so far. Variable P represents the length of the cyclic behavior of the piecewise constant arrival pattern, and T represents the length of a process cycle. Both variables play an important role in the desired periodic behavior. The relationship of both variables in relation with the periodic behavior is sketched in Example 3.2.1.

Example 3.2.1. In a network of two machines in series, two types of products arrive at a constant rate. The network operates in steady state and the arrival of products at the first machine is constant. When the products are processed in the first machine, they are sent to a second machine. These products arrive with a piecewise constant arrival pattern at the second machine. Using the parameter setting in Figure 3.3, the following values for T and P are determined.

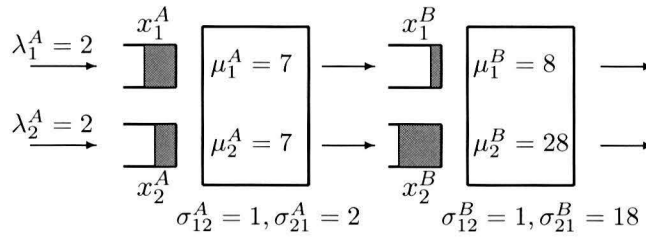


Figure 3.3: Manufacturing machine

- Machine 1:

No piecewise constant arrival pattern is present with respect to machine 1.

Using (3.8):

$$T_A = \frac{1 + 2}{1 - \frac{2}{7} - \frac{2}{7}} = 7 \text{ h.}$$

- Machine 2:

The piecewise constant arrival pattern of product types 1 and 2 is equal to the length of the process cycle of machine A: $P_B = T_A = 7 \text{ h.}$

The network operates in steady state. This means the number of products that arrive during time span T_A at machine 1 is equal to the number of products that leave the machine. So for the second machine holds that $\overline{\lambda}_i^B = \lambda_i^A$ where: $i \in \{1, 2\}$.

$$T_B = \frac{1 + 18}{1 - \frac{2}{8} - \frac{2}{28}} = 28 \text{ h.}$$

Machine 1 has periodic behavior each 7 hours. Machine 2 shows periodic behavior each 28 hours, because $4P_B = T_B$.

The example shows the situation when the period P_B is smaller than the process cycle ($P_B < T_B, \min$). In other situations it is possible that $P \geq T_{\min}$. In those situations the length of the periodic behavior becomes T .

In this report the assumption is made that the desired periodic behavior has a length of:

$$P = T > \frac{\sigma_{12} + \sigma_{21}}{1 - \rho_1 - \rho_2}. \quad (3.9)$$

In (3.9) the assumption is made that one process cycle T has the same time span as the length of one period P . The situation $T = T_{\min}$ is precluded. The exclusion is explained in Chapter 5 (Remark 5.3.5). When the assumption is met, the process cycle always has a steady state and a controller can be found that steers the system to this steady

state.

The characteristics, variables and dynamics of the system are discussed in this chapter. With these properties the length of the desired periodic behavior / steady state process cycle is determined.

In short, when a two product manufacturing system with one piecewise constant and one constant arrival rate satisfies the following conditions:

- $\bar{\rho}_1 + \rho_2 < 1$, and
- $P = T > \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}$,

the theory in this report is usable. The rest of this study focusses on the ‘best’ steady state process cycle possible, how to reach it and how to keep it optimal. The ‘best’ performance of a steady state process cycle can be translated into an optimization problem. In the next chapter such optimization problem is established and the steady state process cycle is optimized.

Chapter 4

Optimal process cycle

In the previous chapter the characteristics and dynamics of a switching system with one piecewise constant and one constant arrival rate have been discussed. The system has to meet two conditions:

- $\bar{\rho}_1 + \rho_2 < 1$.
- $P = T > \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}$.

The first conditions implies that the sum of time fractions that each product needs to be processed is never too large to find a steady state process cycle. The second condition presents the minimum length of a process cycle. If the process cycle is larger than this value, one can find a steady state process cycle. In this study the process cycle has to be optimized with respect to time averaged weighted work in progress (wip) level. The optimization is performed in different steps. In the first section the optimization objective is fully established. In the second section the individual buffer levels are described analytically for a time span of one process cycle. Finally the analytical representation of the total system is optimized in the third section.

4.1 General analysis of the process cycle

In this section, the optimization problem is defined. The objective is to optimize the steady state process cycle of the system with respect to the time averaged cumulative costs related to the wip levels of both buffers. As defined in the previous chapter, the length of one process cycle T equals one period P hours. Within this time span the process cycle is optimized. The costs of the system are defined as J . Where J is:

$$J = \frac{1}{P} \int_0^P c_1 x_1(s) + c_2 x_2(s) ds. \quad (4.1)$$

In (4.1) variables x_1 and x_2 are the buffer levels of respectively type 1 and type 2. The weighing factors c_1 and c_2 are assumed to be constant factors for type 1 and 2 respectively. So to minimize costs, the wip level has to be minimized. Minimizing the wip-levels of a two product manufacturing system leads to several statements. The lemmas which play a role for this system are discussed shortly. The proofs of these lemmas are given in [Eek06b].

Lemma 4.1.1. *When serving type i , optimal policies first serve at the highest possible rate, after which they might idle.*

Lemma 4.1.2. *For optimal steady state behavior of type i , buffer i is always emptied.*

The analysis in this report focuses on a system where the length of the steady state process cycle is equal to the length of the piecewise constant arrival pattern. To fit both lengths it is very likely the machine has to stay in a certain mode although the buffer is empty. In such a situations the machine has to become idle or the machine has to process in a ‘slow mode’ [Eek06a]. When the system processes in a slow mode, the process rate is equal to the arrival rate. If a slow mode becomes active, the machine does not use its full capacity but the machine keeps processing instead of become idle or performs a setup. The effects of idling and slow modes result in a two new statements.

Lemma 4.1.3. *Optimal policies do not idle.*

Proof. The system holds a constant arrival pattern for type 2 and a piecewise constant arrival pattern for type 1. When the machine finished processing type 1 and no products of type 1 arrive, the machine can idle. At the same time products of type 2 are arriving at buffer 2 continuously. To keep the buffer level of type 2 as low as possible, the machine must switch to process type 2 as soon as possible. Idling after type 2 is processed, is not an option because products keep arriving and the machine can process in a slow mode. \square

So the machine is not allowed to idle. This brings the following statement.

Lemma 4.1.4. *If the length of one steady state process cycle is longer than its minimum necessary length ($T > T_{min}$), the optimal steady state process cycle contains at least one slow mode.*

Proof. When the length of a steady state process cycle is longer than the minimum length, the machine has more capacity than needed. The machine has to meet Lemma 4.1.1 but the machine is not allowed to idle (Lemma 4.1.3). The only way to avoid idling if a machine has too much capacity is to keep processing lots at the same rate as its arrival rate (slow mode) after the buffer is emptied. \square

The occurrence of slow modes is inevitable when discussing process cycles with a length larger than T_{min} . In the next section such process cycles are discussed and optimized. Before the total system is optimized, the individual buffers are analyzed.

4.2 Analysis of separate buffer levels

In order to get a good insight how the buffer level evolves during one process cycle, the buffer levels are analyzed separately in this section. In general, during one process cycle, the buffer level increases and decreases over time. These fluctuations are presented and explained in this section. First the constant arrival rate of type 2 is discussed. Because the arrival pattern is time independent, the fluctuations of the buffer level depend only on the state of the system. This makes it possible to analyze both buffer levels separately. Therefore, the time span where type 1 is processed (τ_1), is assumed to be constant. Each trajectory is divided in four time spans found earlier, process type 1, setup to type 2, process type 2 and setup to type 1 (symbolic respectively: τ_1 , σ_{12} , τ_2 and σ_{21}). Within the process intervals a subdivision is made between processing with a maximum capacity of μ or at a lower rate equal to the arrival rate λ (slow mode). The subdivision leads to the following equality:

$$\tau_i = \tau_i^\mu + \tau_i^\lambda \quad i \in \{1, 2\} \quad (4.2)$$

The six time spans together form one process cycle. After the analysis, both trajectories of the buffer levels are combined and both buffer levels are optimized over τ_1 .

4.2.1 Analysis of the buffer level with a constant arrival rate.

First the buffer with a constant arrival rate is discussed. This situation refers to buffer 2 of the two product workstation. Because the analysis focusses on the behavior of a buffer with no time dependent arrival pattern, the arrival pattern can not influence the behavior of the buffer level. The goal is to find the mean wip level of the buffer. To reach this objective, the behavior of the buffer level must be determined during different process steps. During the whole process cycle the buffer receives products at a constant rate of λ_2 products/hour. The result is a linear increase when the machine is not processing type 2. After $\textcircled{2}$ the machine is able to process type 2 ($\textcircled{2}$) and the number of products decreases until the buffer is empty. This behavior is presented in Figure 4.1.

Figure 4.1 shows the optimal behavior within one process cycle with a given time span for τ_1 . The figure assumes steady state behavior where the buffer has to be emptied (Lemma 4.1.2) and the buffer level at the start of the process cycle is equal to the buffer level at the end of the process cycle. In Figure 4.1 the machine processes at two speeds. As long as the buffer is not empty the machine processes at a rate of μ_2 (Lemma 4.1.1). The decrease of products during the time span τ_2^μ equals $\mu_2 - \lambda_2$ products/hour. Afterwards, a slow mode of type 2 may occur. Appearance of a slow mode for type 2 depends on the parameter setting which is discussed in Section 4.3. If a slow mode occurs, the process rate equals the arrival rate λ_2 and has a time span τ_2^λ . The mean wip level can be computed with an equation like (4.1). A specific expression

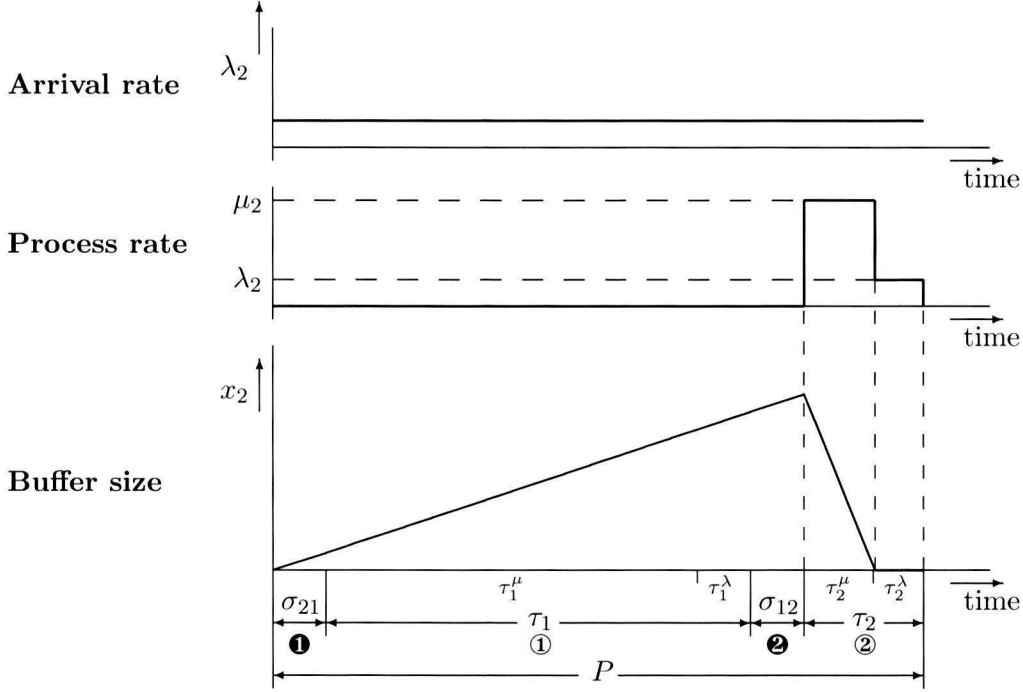


Figure 4.1: Trajectory of buffer with constant arrival pattern.

for the mean wip level of buffer 2 (\bar{w}_2) is:

$$\bar{w}_2 = \frac{1}{P} \int_0^P x_2(s) ds. \quad (4.3)$$

To obtain a relative simple optimization problem, the wip level of type 2 is formulated as a function of τ_1 . To describe the wip level of buffer 2, it means τ_2^μ and τ_2^λ become a function of τ_1 .

When the machine is in mode ①, ① or ②, buffer 2 receives product at a rate of λ_2 for a time span of $(\sigma_{21} + \tau_1 + \sigma_{12})$ hours. During the time span τ_2^μ the buffer level decreases with an effective rate of $(\mu_2 - \lambda_2)$. Because the total number of products that arrive must equal the number of processed products, the following equality holds:

$$\lambda_2(\sigma_{21} + \tau_1 + \sigma_{12}) = (\mu_2 - \lambda_2)\tau_2^\mu.$$

Rewriting the equation and introducing the individual time fractions of type 1 and 2 result in:

$$\begin{aligned} \tau_2^\mu &= \frac{\lambda_2}{\mu_2 - \lambda_2} (\sigma_{21} + \tau_1 + \sigma_{12}) \\ &= \frac{\rho_2}{1 - \rho_2} (\sigma_{21} + \tau_1 + \sigma_{12}). \end{aligned} \quad (4.4)$$

In these equations the arrival rate is constant, this implies $\lambda_2 \equiv \bar{\lambda}_2$ and $\rho_2 \equiv \bar{\rho}_2$. The last step of the determination of the trajectory of the buffer level of type 2 is to find an

expression for τ_2^λ . In Figure 4.1 variable τ_2^λ is the last time interval before the period ends. Or:

$$\tau_2^\lambda = P - (\sigma_{21} + \tau_1 + \sigma_{12} + \tau_2^\mu),$$

and after substitution with (4.4)

$$\begin{aligned} \tau_2^\lambda &= P - \left[1 + \frac{\rho_2}{1-\rho_2}\right](\sigma_{21} + \tau_1 + \sigma_{12}) \\ &= P - \frac{1}{1-\rho_2}(\sigma_{21} + \tau_1 + \sigma_{12}). \end{aligned} \tag{4.5}$$

With the expressions for the intervals τ_2^μ and τ_2^λ , the mean wip level is determined. Physically (4.3) computes the area underneath the contour of the number of products in the buffer (in Figure 4.1) and divides it by P . The triangular area underneath the contour is equal to half the length of the base multiplied with the height. The result is:

$$\begin{aligned} \overline{w_2} &= \frac{\frac{1}{2}(P-\tau_2^\lambda) \cdot \text{height}}{P} \\ &= \frac{1}{2P} \frac{1}{1-\rho_2} (\sigma_{21} + \tau_1 + \sigma_{12}) \cdot \lambda_2 (\sigma_{21} + \tau_1 + \sigma_{12}) \\ &= \frac{\lambda_2}{2P(1-\rho_2)} (\sigma_{21} + \tau_1 + \sigma_{12})^2 \\ &= \frac{1}{2P} \frac{\mu_2 \rho_2}{1-\rho_2} (\sigma_{21} + \tau_1 + \sigma_{12})^2 \end{aligned} \tag{4.6}$$

Remark 4.2.1. During processing type 2 the machine needs to process the same number of product as arrive during one process cycle. Therefore the length of τ_1 can not be too large. The minimum length of time (τ_2^{\min}) to process all products of type 2 arriving during one period is to process all products at the machines maximum capacity:

$$\tau_2^{\min} = \frac{\lambda_2 P}{\mu_2} = \rho_2 P$$

With the presence of a minimum length for τ_2 and a fixed length for a process cycle, a maximum length or upper bound for τ_1 is established. Using $P = \tau_1 + \sigma_{12} + \tau_2 + \sigma_{21}$ and $\tau_2 \geq \rho_2 P$, the upper bound for τ_1 becomes:

$$\tau_1 \leq (1 - \rho_2)P - (\sigma_{12} + \sigma_{21}) \tag{4.7}$$

Equation 4.7 forms the upper bound for τ_1 .

At this moment the mean wip level of type 2 is determined and an upper bound for τ_1 is established. In the next section the upper bound is used as a constraint of the optimization problem.

4.2.2 Analysis of the buffer level with a piecewise constant arrival rate.

This part contains the derivation of the mean wip level of a buffer which receives products at a piecewise constant rate. The mean wip level is based on a time span of one period (P). This situation refers to buffer 1 of the two product workstation. In contrast with the constant arrival pattern, when a time dependent arrival pattern is used, a relationship exists between the time of processing type 1 and the time interval where products of type 1 arrive. Thereby, the fraction of a period where products arrive plays an important role in the determination of the wip level also. As mentioned in the previous part, τ_1 is the candidate to be optimized over. So in advance each length where $\tau_1 > 0$ is possible as long as constraint (4.7) is met. Like the established upper bound in (4.7) for τ_1 , also a lower bound can be determined. During one process cycle the machine needs enough time to process all products that arrive during one period. For the length of τ_1 this means:

$$\begin{aligned}\tau_1 &\geq \frac{\bar{\lambda}_1 P}{\mu_1} \\ &\geq \bar{\rho}_1 P\end{aligned}\tag{4.8}$$

Equation 4.8 forms the second constraint for τ_1 .

Remark 4.2.2. A situation where the lower bound is larger than the upper bound is not possible without violating the conditions mentioned at the start of this chapter. If $\bar{\rho}_1 P > (1 - \rho_2)P - (\sigma_{12} + \sigma_{21})$:

$$(\sigma_{12} + \sigma_{21}) > (1 - \bar{\rho}_1 - \rho_2)P.$$

While condition (3.9) must hold which implies:

$$(\sigma_{12} + \sigma_{21}) < (1 - \bar{\rho}_1 - \rho_2)P.$$

These two inequalities are reconcilable with each other. So a situation where τ_1 can not meet both upper and lower bound is not discussed in this report.

When τ_1 lies between the upper and lower bound, sufficient time is available to process enough products of each type in one process cycle. So with these constraints the wip level that has to be determined is:

$$\bar{w}_1 = \frac{1}{P} \int_0^P x_1(s) ds.\tag{4.9}$$

The trajectory of the buffer level depends heavily on the arrival rate. If many products arrive in a short amount of time ($\hat{\lambda}_1 \geq \mu_1$), a new lemma needs to be introduced to define the trajectory of the buffer level during one period. In the second situation where $\hat{\lambda}_1 < \mu_1$ the trajectory depends, besides the Lemmas 4.1.1 and 4.1.2, on other relationships too. So two situations are distinguished:

Situation I : $\hat{\lambda}_1 \geq \mu_1$.
 Situation II : $\hat{\lambda}_1 < \mu_1$.

During the remainder of this section these two situations are optimized with respect to the relation between the piecewise arrival pattern and the moment in time where type 1 has to be processed. After the determination of these relations the mean wip levels are determined. In the next section the mean wip levels of buffer 1 and 2 are combined for all different situations and optimized with respect to τ_1

Remark 4.2.3. In case $\hat{\lambda}_1$ or ϕ_1 is unknown, a relation exists between these two parameters. This relationship is used in the course of this chapter. The relationship is performed by substitution of (3.1) and (3.2) for type 1:

$$\bar{\lambda}_1 = \phi_1 \hat{\lambda}_1 \quad \text{and} \quad \bar{\rho}_1 = \frac{\bar{\lambda}_1}{\mu_1}$$

substituted:

$$\hat{\lambda}_1 = \frac{\bar{\rho}_1 \mu_1}{\phi_1}. \tag{4.10}$$

Situation I

Situation I represents the situation where $\hat{\lambda}_1 \geq \mu_1$. This means that many products arrive in a short amount of time. To obtain a minimum mean wip level in the buffer, a relation is established between the start of processing type 1 and the point in time where products of type 1 start to arrive.

Lemma 4.2.4. *In the situation where a buffer receives lots at a piecewise constant arrival rate and the rate of arrivals is higher than the process rate, minimizing the wip level of the buffer during one process cycle means a coincidence of the start of processing and the point in time where lots start to arrive.*

Proof. When a system receives lots at a rate that is higher than the process rate, an increase of the buffer level is inevitable. To minimize the number of products in the buffer, the rate of increasing must be kept to a minimum. By starting processing when lots start to arrive (= as soon as possible) the rate of an increasing buffer level is kept to a minimum. The minimum increase results in the lowest value possible for the maximum buffer level after all products have arrived. Also a lower maximum buffer level results in a shorter processing time after the lots stopped arriving. Processing lots as soon as possible, as performed here, leads to a minimization of the wip level of type 1.

Note, this proof uses the constant arrival pattern of type 2. □

Using Lemma 4.2.4, the start of processing type 1 has to coincide with the start of arrivals of type 1. During the processing of type 1 the buffer level increases until the

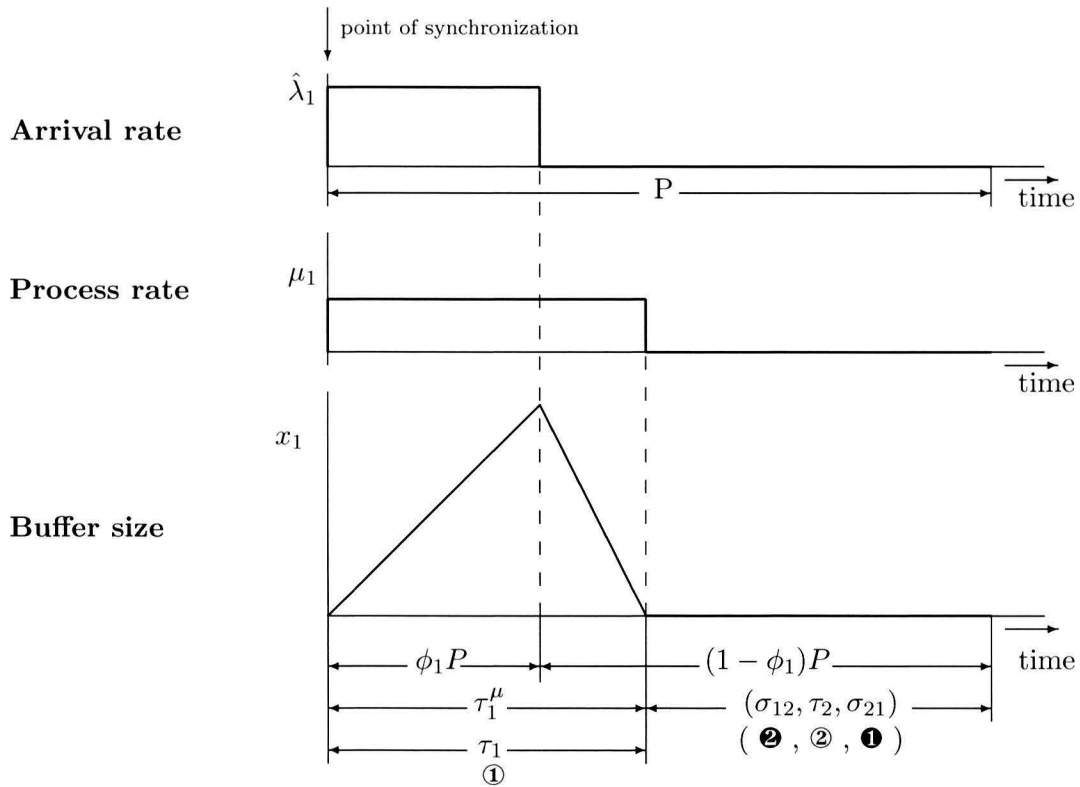


Figure 4.2: **Situation I:** Trajectory of buffer with piecewise constant arrival pattern.

products stop arriving. Then the buffer level decreases at a rate of μ_1 . In Figure 4.2 this behavior is presented.

In Figure 4.2 the optimal trajectory of a buffer with piecewise constant arrival pattern is depicted. As one can see, the machine processes type 1 at its maximum rate as long as possible (τ_1^μ). When the buffer is empty the machine can stay in $\textcircled{1}$ (still τ_1 has to be optimized in combination with buffer 2). But from the moment the buffer is empty no products of type 1 arrive nor have to be processed. As mentioned in Lemma 4.1.3, idling is not permitted within an optimal process cycle, so indisputably $\tau_1^\mu = \tau_1$ and no slow mode is active ($\tau_1^\lambda = 0$).

To determine the wip level of type 1, the triangular area underneath the contour has to be determined and divided by the length of the period P . The base of this triangle is $\tau_1^\mu = \bar{\rho}_1 P$. The height of the triangle is equal to the number of products that arrived during time span $\phi_1 P$ minus the number of products that are processed during the same time interval. The height becomes:

$$\hat{\lambda}_1 \phi_1 P - \mu_1 \phi_1 P = (\hat{\lambda}_1 - \mu_1) \phi_1 P.$$

Substitution of $\hat{\lambda}_1$ with (4.10) results in:

$$\text{height} = (\bar{\rho}_1 - \phi_1) \mu_1 P. \quad (4.11)$$

With the use of $\tau_1^\mu = \bar{\rho}_1 P$ and (4.11) the wip level of situation I becomes:

$$\begin{aligned} \overline{w_1^{(I)}} &= \frac{\frac{1}{2}(\tau_1^\mu) \cdot \text{height}}{P} \\ &= \frac{\frac{1}{2}\bar{\rho}_1 P(\bar{\rho}_1 - \phi_1)\mu_1 P}{P} \\ &= \frac{1}{2}\mu_1 \bar{\rho}_1 (\bar{\rho}_1 - \phi_1) P. \end{aligned} \tag{4.12}$$

As long as the constraint holds that $\hat{\lambda}_1 \geq \mu_1$, τ_1 has a fixed length and is processing type 1 in a slow mode not possible, what results in a fixed trajectory for type 1 also. So now the mean wip level of type 1 is determined for situation I. In the next part the wip level for situation II is discussed.

Situation II

In situation II the products of type 1 arrive at a rate less than μ_1 , or $\hat{\lambda}_1 < \mu_1$. Using (4.10) the following relation can be established:

$$\frac{\hat{\lambda}_1}{\mu_1} = \frac{\bar{\rho}_1}{\phi_1}. \tag{4.13}$$

Equation 4.13 implies that in this situation if $\hat{\lambda}_1 < \mu_1$ means that $\phi_1 > \bar{\rho}_1$. The rate at which products arrive is smaller than the maximum process rate. As in situation I exists a relationship between the arrival rate and processing type 1. Again the optimal situation must be obtained.

Lemma 4.2.5. *Optimizing situation II where the highest arrival rate of a piecewise constant arrival pattern is less than the process rate, the end of arrivals must coincide with the end of processing that type of job.*

Proof. With a given time span for τ_1 and a constant arrival rate of the jobs of type 2, the systems has to meet different constraints. During one process cycle the buffer level has to become zero (Lemma 4.1.2), the machine is not allowed to idle (Lemma 4.1.2), has to process $\bar{\lambda}_i P$; $i \in \{1, 2\}$ jobs and perform two setups. Within these constraints the wip level of the job type with piecewise constant arrival pattern has to be minimized. To obtain the lowest mean wip level possible, the buffer has to be kept empty as long as possible. To achieve this objective the buffer has to be empty during the time interval where no jobs arrive. When no jobs arrive, the machine has to process the other job type and perform setups. When processing the job type with piecewise constant arrival pattern starts, processing this job starts with the lowest buffer level possible. So to obtain the largest interval with an empty buffer, the machine has to process the job type with piecewise constant arrival pattern until the jobs stop to arrive. \square

In one steady state process cycle the number of processed products must equal the number of arrived products during one process cycle. So at the end of processing type 1 a fixed number of products have to be processed. To prevent idling, the machine has to finish processing type 1 if the buffer is empty and no products arrive. So the point of synchronization of the arrival pattern and process cycle is at the end of processing type 1. This leads to a trajectory presented in Figure 4.3.

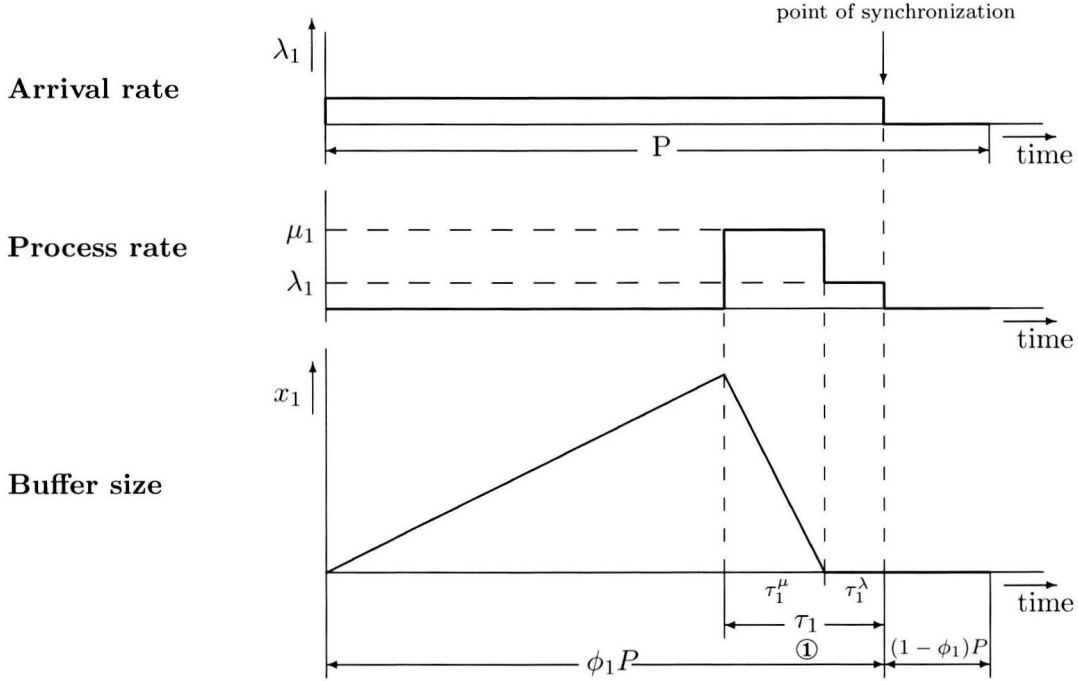


Figure 4.3: **Situation II:** Trajectory of buffer with piecewise constant arrival pattern.

Figure 4.3 shows the number of products in the buffer increases when type 1 is not processed. The decrease starts when the machine starts ① at a rate of μ_1 . When the buffer is empty and interval $\phi_1 P$ is not finished yet, the machine process in a slow mode until the interval $\phi_1 P$ ends.

The derivation of the mean wip level is similar to situation I, again the area underneath the contour has a triangular shape. The width of the base is the length of the interval where products arrive ($\phi_1 P$) minus the interval τ_1^λ . The height of the triangle is equal to the length of $\phi_1 P$ minus the length of τ_1 multiplied by the arrival rate. So the height becomes:

$$\text{height} = (\phi_1 P - \tau_1) \hat{\lambda}_1 = (\phi_1 P - \tau_1) \frac{\mu_1 \bar{\rho}_1}{\phi_1} \quad (4.14)$$

The equation of the width of the base contains the variable τ_1^λ . This parameter needs to be replaced by a function of τ_1 to simplify the total optimization problem in the next

section. A steady state behavior implies that in this situation the number of arrivals must equal the number of processed products. Or:

$$\mu_1 \tau_1^\mu + \hat{\lambda}_1 \tau_1^\lambda = \hat{\lambda}_1 \phi_1 P. \quad (4.15)$$

The relation between τ_1 and τ_1^λ is

$$\tau_1 = \tau_1^\mu + \tau_1^\lambda. \quad (4.16)$$

Combining (4.15) and (4.16):

$$\mu_1(\tau_1 - \tau_1^\lambda) + \hat{\lambda}_1 \tau_1^\lambda = \hat{\lambda}_1 \phi_1 P$$

$$\tau_1^\lambda = \frac{\mu_1 \tau_1 - \hat{\lambda}_1 \phi_1 P}{\mu_1 - \hat{\lambda}_1}.$$

Finally, substitution of $\hat{\lambda}_1$ with the use of (4.13) results in:

$$\tau_1^\lambda = \frac{(\tau_1 - \bar{\rho}_1 P) \phi_1}{\phi_1 - \bar{\rho}_1}. \quad (4.17)$$

For the width of the base follows:

$$(\phi_1 P - \tau_1^\lambda) = \frac{\phi_1 P - \tau_1}{\phi_1 - \bar{\rho}_1} \phi_1. \quad (4.18)$$

The expression for τ_1^μ is found by substitution of (4.17) in (4.16):

$$\tau_1^\mu = \frac{(\phi_1 P - \tau_1) \bar{\rho}_1}{\phi_1 - \bar{\rho}_1}. \quad (4.19)$$

With the found expressions for τ_1^λ , τ_1^μ and the use of (4.14) the mean wip level for situation II becomes:

$$\begin{aligned} \overline{w_1^{(II)}} &= \frac{\frac{1}{2}(\phi_1 P - \tau_1) \cdot \text{height}}{P} \\ &= \frac{1}{2P} \frac{\phi_1 P - \tau_1}{\phi_1 - \bar{\rho}_1} \phi_1 \cdot (\phi_1 P - \tau_1) \frac{\mu_1 \bar{\rho}_1}{\phi_1} \\ &= \frac{1}{2P} \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \bar{\rho}_1} (\phi_1 P - \tau_1)^2. \end{aligned} \quad (4.20)$$

With (4.20) a mean wip level is found for situation II, for a given τ_1 as long as its length is defined between the lower and upper bound mentioned in (4.8) and (4.7).

At this point a mean wip level is determined for both situations. During the determination of the trajectories with a piecewise constant arrival pattern, the first step to optimization is to synchronize the arrival pattern and the processing of type 1. The next step is to combine situations I and II with buffer 2 and optimize the total system with respect to τ_1 . In Section 4.3 these combinations are optimized.

4.3 Optimal steady state process cycle

In the previous section the trajectories of buffer 1 and 2 have been discussed. A fixed time span for processing type 1 (τ_1) is used during the analysis. With the fixed time span, mean wip levels for both buffers are defined as a function of τ_1 . In this section, the trajectories of buffer 1 and 2 are combined and the sum of mean wip levels is minimized by optimizing the length of τ_1 . In the previous section, two situations for type 1 have been discussed. In this section these two situations return. The total optimization mentioned in 4.1 is rephrased in minimizing the time averaged weighted wip level:

$$\begin{aligned} \min_{\tau_1} \quad & J_{(r)}(\tau_1) = c_1 \overline{w_1^{(r)}} + c_2 \overline{w_2} \quad \text{with:} \quad r \in \{\text{I, II}\} \\ \text{s.t.} \quad & \text{Constraints on } \tau_1 \end{aligned} \tag{4.21}$$

With the minimization problem come the constraints for τ_1 . Besides the lower and upper bound, (4.8) and (4.7) respectively, the time interval where products arrive ($\phi_1 P$) plays an important role also. In this section the two situations of the previous section are optimized and their optimal steady state behavior is presented.

4.3.1 Situation I ($\hat{\lambda}_1 \geq \mu_1$)

The behavior of buffer 1 in situation I has been discussed in the previous section, the result is a fixed trajectory for buffer 1 where no slow mode is possible because idling is forbidden by Lemma 4.1.3. When this behavior of type 1 is combined with the trajectory of buffer 2, trajectory of type 2 is captured also. After finishing type 1, the machine is not allowed to idle due to Lemma 4.1.3. The machine performs a setup to process type 2 and processes at its maximum capacity (Lemma 4.1.1) until the buffer is empty. Still Lemma 4.1.3 has to be met when buffer 2 is emptied (Lemma 4.1.2). As Lemma 4.1.4 showed, at least one slow mode is active during one process cycle. Because no slow mode can exist of type 1 a slow mode for type 2 has to exist.

Remark 4.3.1. The slow mode of type 2 always exists in the optimal process cycle for the situation where $\hat{\lambda}_1 \geq \mu_1$.

Proof. The length of $\tau_2^\lambda = P - \frac{1}{1-\rho_2}(\sigma_{21} + \overline{\rho_1}P + \sigma_{12})$, for the constraint of a period holds (3.9). Equation (3.9) can be modified into:

$$-\sigma_{12} + \sigma_{21} > -P(1 - \overline{\rho_1} - \rho_2).$$

Substitution with τ_2^λ leads to the following inequality:

$$\begin{aligned} P - \frac{1}{1-\rho_2}(\sigma_{21} + \overline{\rho_1}P + \sigma_{12}) &> P - \frac{1}{1-\rho_2}(P(1 - \overline{\rho_1} - \rho_2) + \overline{\rho_1}P) \\ &> P - \frac{1}{1-\rho_2}P(1 - \rho_2) \\ &> 0. \end{aligned} \tag{4.22}$$

Due to (4.22) the optimal process cycle always contains a time interval where a slow mode of type 2 is active. \square

The mean wip level of the optimal trajectory is determined by substitution of (4.12) and (4.6) in the optimization problem (4.21) where $\tau_1 = \tau_1^\mu = \bar{\rho}_1 P$.

$$J_1 = c_1 \frac{1}{2} \mu_1 \bar{\rho}_1 (\bar{\rho}_1 - \phi_1) P + c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12})^2. \quad (4.23)$$

Equation (4.23) presents the lowest time averaged weighted mean wip level possible for situation I when one period is equal to one process cycle.

4.3.2 Situation II ($\hat{\lambda}_1 < \mu_1$)

Like situation I, situation II also has to be optimized. In this situation the products of type 1 arrive at rate which is less than the maximum process rate of type 1. The consequence of $\hat{\lambda}_1 < \mu_1$, as mentioned in (4.13) of the previous section, is that $\phi_1 > \bar{\rho}_1$.

In situation II Lemma 4.2.5 holds. It means when the products of the piecewise constant arrival pattern stop, the machine has to switch to the other type of products.

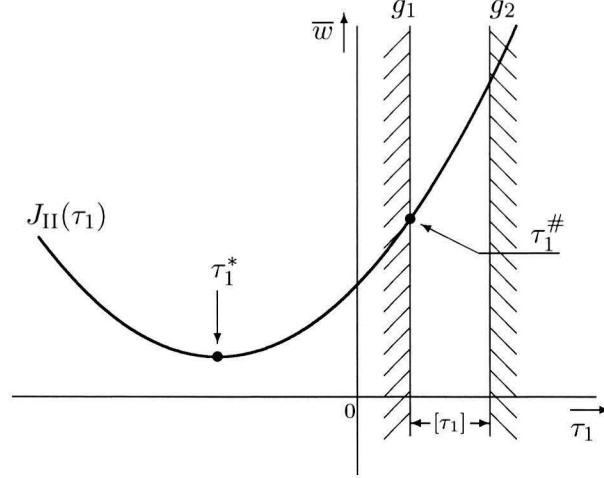
Remark 4.3.2. Besides the coincidence of the end of processing type 1 and the stop of arrivals, the length of the process time of type 1 (τ_1) can never exceed the length of the time span where products arrive ($\phi_1 P$). This is only possible if the process rate is less than the arrival rate. Due to Lemma 4.1.1 this is not allowed.

The optimization problem for situation II is written below:

$$\begin{aligned} \min_{\tau_1} \quad & J_{II}(\tau_1) = c_1 \frac{1}{2P} \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \bar{\rho}_1} (\phi_1 P - \tau_1)^2 + c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \tau_1 + \sigma_{12})^2 \\ \text{s.t.} \quad & \mathbf{g}_1(\tau_1) \quad \bar{\rho}_1 P - \tau_1 \leq \mathbf{0} \\ & \mathbf{g}_2(\tau_1) \quad \tau_1 - (1 - \rho_2)P + (\sigma_{21} + \sigma_{12}) \leq \mathbf{0}. \end{aligned} \quad (4.24)$$

Figure 4.4 shows an objective function of the optimization problem (4.24). The constraints (g_1 and g_2) determine the optimum of τ_1 in this case. The (constraint) optimum in this example is $\tau_1^\# = \bar{\rho}_1 P$, while the absolute optimum is τ_1^* .

In optimization problem (4.24) the first term of the objective is semi-positive definite because $\phi_1 > \bar{\rho}_1$. The second term is positive definite, what makes the sum of both terms positive definite also. The optimization of a positive definite quadratic function results in a parabola which top is a minimum. The constraints form the absolute lower and upper bound for τ_1 . For the determination of the absolute optimum τ_1^* , the derivative

Figure 4.4: Optimum for τ_1

of the objective function has to be zero:

$$\frac{dJ_{II}(\tau_1)}{d\tau_1} = 2A\tau_1 - 2A\phi_1P + 2B\tau_1 + 2B(\sigma_{21} + \sigma_{12}) = 0$$

where:

$$A = c_1 \frac{1}{2P} \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \rho_1}$$

$$B = c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2}$$

This leads to an optimum of:

$$\begin{aligned} \tau_1^* &= \frac{A\phi_1P - B(\sigma_{21} + \sigma_{12})}{(A+B)} = \frac{\left(c_1 \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \rho_1} (\phi_1 P) - c_2 \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \sigma_{12}) \right)}{c_1 \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \rho_1} + c_2 \frac{\mu_2 \rho_2}{1 - \rho_2}} \\ &= \frac{c_1 \mu_1 \bar{\rho}_1 (1 - \rho_2) (\phi_1 P) - c_2 \mu_2 \rho_2 (\phi_1 - \bar{\rho}_1) (\sigma_{21} + \sigma_{12})}{c_1 \mu_1 \bar{\rho}_1 (1 - \rho_2) + c_2 \mu_2 \rho_2 (\phi_1 - \bar{\rho}_1)}. \end{aligned} \quad (4.25)$$

Note: The expression τ_1^ represents the optimal time span needed to process all products of type 1 without taking the constraints of (4.24) into account.*

Taking the constraints into account, three possible scenarios are possible:

1. $\tau_1^* < g_1$

Solution: $\tau_1^\# = \bar{\rho}_1 P$

$$2. \mathbf{g}_1 < \tau_1^* < \mathbf{g}_2$$

$$\text{Solution: } \tau_1^\# = \tau_1^*$$

$$3. \tau_1^* > \mathbf{g}_2$$

$$\text{Solution: } \tau_1^\# = (1 - \rho_2)P - (\sigma_{12} + \sigma_{21})$$

The first scenario represents the situation where the optimal length of processing type 1 is less than the lower bound. This can be interpreted that it is important to keep the mean wip-level of type 2 as low as possible (no slow mode for type 1). Scenario two represents a optimal solution where both types have the same importance approximately (slow modes for type 1 and type 2). The last scenario represents the situation where processing type 1 is requires a lot of time to keep its mean wip level low (no slow mode for type 2).

Remark 4.3.3. The optimal τ_1 has to be smaller than the time span where type 1 arrives ($\phi_1 P$) as mentioned in Remark 4.3.2. Due to this requirement scenario 3 can only take place if $\phi_1 P$ is larger than the upper bound of τ_1 .

Remark 4.3.4. A special case is the situation $\hat{\lambda}_1 = \mu_1$. This situation occurs on the boundary of situation I and situation II. In this situation (4.24) and (4.23) must provide the same solution.

Proof. Due to (4.13) variable $\phi_1 = \bar{\rho}_1$ if $\hat{\lambda}_1 = \mu_1$. Furthermore, $\tau_1 = \bar{\rho}_1 P$ holds in both situations. After substitution of $\phi_1 = \bar{\rho}_1$ and $\tau_1 = \bar{\rho}_1 P$ the result is:

$$J_I = c_1 \frac{1}{2} \mu_1 \bar{\rho}_1 (\bar{\rho}_1 - \bar{\rho}_1) P + c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12})^2$$

$$J_{II} = c_1 \frac{1}{2P} \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \bar{\rho}_1} (\bar{\rho}_1 P - \bar{\rho}_1 P)^2 + c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12})^2$$

$$J_I = J_{II} = c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12})^2.$$

The transition of situation I into situation II is good because at the boundary of both situations $J_I = J_{II}$. \square

By updating the optimal value for τ_1 with applying the constraints, the optimal trajectory is defined for all possible parameter settings which can occur in situation II.

In the next section the different shapes of possible optimal trajectories are discussed.

4.4 Optimal steady state trajectories

All optimal steady state process cycles as described in Section 4.3 have their own characteristics. In this section the results of the optimal steady state process cycles are translated into the trajectories of the buffer levels during one process cycle. All situations are discussed, including the three possible scenarios for situation II.

4.4.1 Trajectory of situation I

Situation I distinguishes itself, besides the fact that $\hat{\lambda}_1 \geq \mu_1$, by the coincidence of the start of arrivals of type 1 with the start of processing type 1. The result is the optimal trajectory of the steady state process cycle of situation I in Figure 4.5. The trajectories

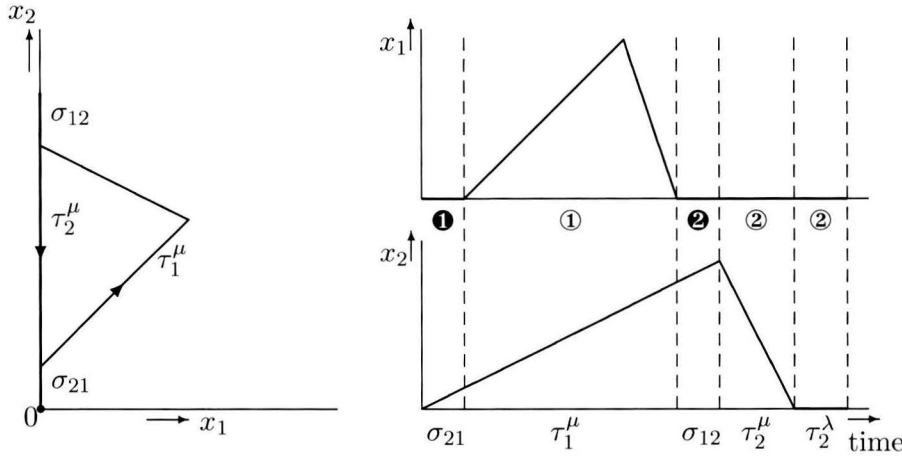


Figure 4.5: Trajectory of situation I. Left: Periodic orbit. Right: Buffer levels over time.

in Figure 4.5 start at the begin of setup **1**. The right-hand graphs show the buffer levels of type 1 and 2 over time. On the left-hand side the buffer levels are plotted against each other. The course of the buffer levels form a counter-clockwise trajectory. After setup **1** is performed, the machine starts processing type 1 at a rate of μ_1 . At the same time products of type 1 start to arrive at a higher rate ($\hat{\lambda}_1$). With regard to the trajectory of the periodic orbit, the result is a vector of $[\hat{\lambda}_1 - \mu_1, \lambda_2]^T$ as long as products of type 1 arrive. When the products stop arriving the new vector becomes $[-\mu_1, \lambda_2]^T$ until $x_1 = 0$. Then **2** is performed immediately ($\tau_1^\lambda = 0$) and afterwards type 2 is processed where the vector is $[0, -\mu_1]^T$ until the point is reached where both buffers are empty. The machine continues processing type 2 in a slow mode (in point $(0,0)$!). The length of the time intervals of the process rates are determined with (4.4), (4.5) and the determination of τ_1 by using the earlier presented lemmas:

$$\begin{aligned}
 \tau_2^\mu &= \frac{\rho_2}{1-\rho_2}(\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}) \\
 \tau_2^\lambda &= P - \frac{1}{1-\rho_2}(\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}) \\
 \tau_1^\mu &= \bar{\rho}_1 P \\
 \tau_1^\lambda &= 0.
 \end{aligned} \tag{4.26}$$

4.4.2 Trajectories of situation II

In Section 4.3 situation I and II are optimized. The determination of the optimal length for processing type 1 has been discussed in that section. The result of the optimization problem in situation II are three possible scenarios. Each scenario has its own characteristics, these are visible in three different trajectories which are discussed here.

Scenario 1

In scenario 1 the mean wip level of type 2 is relatively important. This behavior reveals itself in a trajectory where processing type 1 in a slow mode is not admitted in the optimal process cycle. Such behavior keeps the process time of type 1 as short as possible, so the machine can process type 2 as soon as possible and is able to process type 2 in a slow mode (Lemma 4.1.4). The optimal trajectory is presented in Figure 4.6. On the right-hand side Figure 4.6 shows the trajectories of both buffer levels against

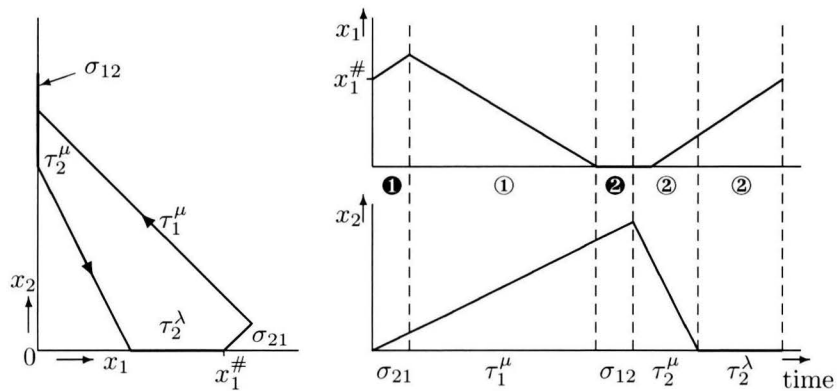


Figure 4.6: Trajectory for scenario 1

time. At the left-hand side the buffer levels are plotted against each other. When buffer 1 reaches its critical value $x_1^\#$, the machine switches from processing type 2 to type 1 (❶). When the machine starts processing, it results in a vector of $[\hat{\lambda}_1 - \mu_1, \lambda_2]^T$. The machine keeps processing type 1 until the buffer is empty. The moment the buffer is empty, the arrivals of type 1 stop. After ❷ the machine processes type 2, first at its maximum capacity and later in a slow mode, until the critical value for type 1 is reached again. The optimization of τ_1 resulted in this scenario in the same length as in situation I, also in both situations only a slow mode for type 2 exists. Therefore the accompanying time intervals τ_1^μ , τ_1^λ , τ_2^μ and τ_2^λ can be determined with the same equations as mentioned in (4.26).

Scenario 2

In this scenario the optimized length for τ_1 is positioned between the lower bound $\tau_1 \geq \bar{\rho}_1 P$ and upper bound $\tau_1 \leq (1 - \rho_2)P - (\sigma_{21} + \sigma_{12})$. The result are process times for type 1 and type 2 that are longer than the minimal required lengths ($\rho_i P$ with: $i \in \{1, 2\}$). So for both types a slow mode exists in the optimal trajectory. Such behavior is presented in Figure 4.7. On the right-hand side Figure 4.7 shows the

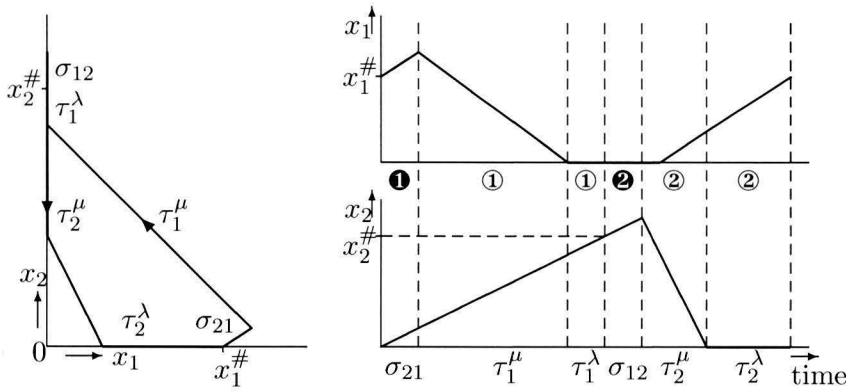


Figure 4.7: Trajectory for scenario 2

trajectories against the time. At the left-hand side the buffer levels are plotted against each other. The trajectory of scenario 2 matches the trajectory of scenario 1 except for the existence of a slow mode. When buffer 1 is empty the machine keeps processing type 1 in a slow mode as long as products keep arriving. The arrivals stop when $x_2 = x_2^\#$ is reached. The length of the time intervals are in accordance with (4.4), (4.5), (4.19) and (4.17), where τ_1 is substituted by (4.25):

$$\begin{aligned}\tau_2^\mu &= \frac{\rho_2}{1-\rho_2}(\sigma_{21} + \tau_1 + \sigma_{12}) \\ \tau_2^\lambda &= P - \frac{1}{1-\rho_2}(\sigma_{21} + \tau_1 + \sigma_{12}) \\ \tau_1^\mu &= \frac{(\phi_1 P - \tau_1)\bar{\rho}_1}{\phi_1 - \bar{\rho}_1} \\ \tau_1^\lambda &= \frac{(\tau_1 - \bar{\rho}_1 P)\phi_1}{\phi_1 - \bar{\rho}_1}\end{aligned}$$

where:

$$\tau_1 = \frac{c_1 \mu_1 \bar{\rho}_1 (1 - \rho_2) (\phi_1 P) - c_2 \mu_2 \rho_2 (\phi_1 - \bar{\rho}_1) (\sigma_{21} + \sigma_{12})}{c_1 \mu_1 \bar{\rho}_1 (1 - \rho_2) + c_2 \mu_2 \rho_2 (\phi_1 - \bar{\rho}_1)}.$$

Scenario 3

The third scenario holds the situation where no slow mode for type 2 exists, but because of Lemma 4.1.4 a slow mode of type 1 has to be present. The accompanying trajectory is presented in Figure 4.8. In the graph on the left-hand side of Figure 4.8 the buffer

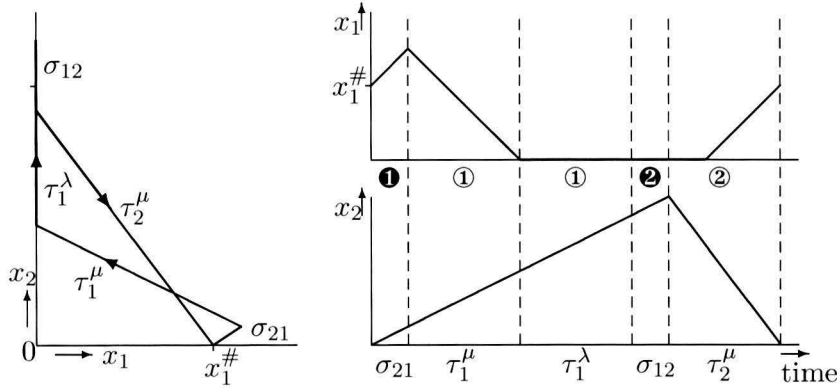


Figure 4.8: Trajectory for scenario 3

levels are plotted against each other. After the setup **1**, the machine starts to process type 1 what results in the vector $[\hat{\lambda}_1 - \mu_1, \lambda_2]^T$. When the buffer is empty it keeps processing type 1 in a slow mode until products of type stop arriving. Then setup **2** is performed and the machine starts to process type 2. When buffer 2 is empty, the machine switches immediately to perform setup **1** again. The accompanying time intervals in this situation are:

$$\begin{aligned}
 \tau_2^\mu &= \rho_2 P. \\
 \tau_2^\lambda &= 0. \\
 \tau_1^\mu &= \frac{((\phi_1 + \rho_2 - 1)P + (\sigma_{21} + \sigma_{12}))\bar{\rho}_1}{\phi_1 - \bar{\rho}_1}. \\
 \tau_1^\lambda &= \frac{((1 - \bar{\rho}_1 - \rho_2)P - (\sigma_{21} + \sigma_{12}))\phi_1}{\phi_1 - \bar{\rho}_1}.
 \end{aligned} \tag{4.27}$$

Equations 4.27 confirm the minimization of the process time of type 2. As long as type 2 is processed, the machines processes at its maximum rate and processing type 2 in a slow mode is not possible.

Scenario $\phi_1 = 1$

A special scenario is when $\phi_1 = 1$. If $\phi_1 = 1$ the piecewise constant arrival pattern disappears what implies that $\hat{\lambda}_1 = \lambda_1$ and $\bar{\rho}_1 = \rho_1$. These properties correspond with

the theory as described in [Eek06a]. To proof the scenario is similar, the time averaged weighted wip level (J) in this scenario has to be the same as presented in [Eek06a]. To obtain this proof the following recipe has to be performed:

1. Optimize the active unconstraint problem to τ_1 and P .
2. If the solution found the previous step is not possible, implement each constraint for τ_1 separately and optimize the constraint problem to P .
3. Check when each constraint can be active using the assumption $c_1\lambda_1 > c_2\lambda_2$ (made in [Eek06a]).
4. Verify if the time averaged weighted wip level (4.24) corresponds with the time averaged weighted wip level (7) in [Eek06a].

For the sake of convenience ($\sigma_{21} + \sigma_{12}$) is written as σ . Now the different steps of the recipe are discussed.

1. Optimizing the unconstrained problem.

First the active situation in this scenario is determined. Variable $\phi_1 = 1$ in combination with (4.13) results in $\hat{\lambda}_1 < \mu_1$. This means situation II is active when $\phi_1 = 1$. If $\phi_1 = 1$ (4.24) becomes:

$$J(\tau_1, P) = c_1 \frac{1}{2P} \frac{\lambda_1}{1 - \rho_1} (P - \tau_1)^2 + c_2 \frac{1}{2P} \frac{\lambda_2}{1 - \rho_2} (\tau_1 + \sigma)^2 \quad (4.28)$$

Next, the unconstraint problem (4.28) is optimized to τ_1 :

$$\frac{\delta J}{\delta \tau_1}(\tau_1^*, P) = 0$$

what results in: (4.29)

$$\tau_1^*(P) = \frac{c_1 \lambda_1 (1 - \rho_2) P - c_2 \lambda_2 (1 - \rho_1) \sigma}{c_1 \lambda_1 (1 - \rho_2) + c_2 \lambda_2 (1 - \rho_1)}.$$

The value for τ_1^* has to be substituted in (4.28) and P is optimized now:

$$\frac{dJ}{dP}(\tau_1(P_{unc}^*), P_{unc}^*) = 0$$

what results in: (4.30)

$$P_{unc}^* = -\sigma \quad \text{or} \quad P_{unc}^* = \sigma.$$

These values for the unconstrained P_{unc}^* form solutions which are invalid. Both solutions do not meet the condition (3.9). This means the optimization problem has to be a constrained problem.

2. Optimizing the constrained problem where one constraint is active.

Each constraint has to be checked separately. First the lower bound constraint is discussed. Optimizing (4.28) with an active lower bound $\tau_1 = \rho_1 P$ subjected to P results in:

$$\frac{dJ}{dP}(P_{LB}^*) = 0$$

what results in:

$$\begin{aligned} P_{LB}^* &= -\frac{c_2 \lambda_2 \sigma}{\sqrt{(c_1 \lambda_1 (1 - \rho_1)(1 - \rho_2) + c_2 \lambda_2 \rho_1^2) c_2 \lambda_2}} \\ \text{or:} \\ &= \frac{c_2 \lambda_2 \sigma}{\sqrt{(c_1 \lambda_1 (1 - \rho_1)(1 - \rho_2) + c_2 \lambda_2 \rho_1^2) c_2 \lambda_2}}. \end{aligned} \quad (4.31)$$

Second, the upper bound is active. Optimizing (4.28) with $\tau_1 = P(1 - \rho_2) - \sigma$ subjected to P results in:

$$\frac{dJ}{dP}(P_{UB}^*) = 0$$

what results in:

$$\begin{aligned} P_{UB}^* &= -\frac{c_1 \lambda_1 \sigma}{\sqrt{(c_2 \lambda_2 (1 - \rho_1)(1 - \rho_2) + c_1 \lambda_1 \rho_2^2) c_1 \lambda_1}} \\ \text{or:} \\ &= \frac{c_1 \lambda_1 \sigma}{\sqrt{(c_2 \lambda_2 (1 - \rho_1)(1 - \rho_2) + c_1 \lambda_1 \rho_2^2) c_1 \lambda_1}}. \end{aligned} \quad (4.32)$$

Variable P describes a time span, because the first solutions of both P_{LB}^* and P_{UB}^* are less than zero, the solutions are rejected.

3. Check which constraint is active.

In [Eek06a] the assumption is made that $c_1 \lambda_1 > c_2 \lambda_2$. Furthermore, the minimum length of a steady state process cycle is set to $P = \frac{\sigma}{1 - \rho_1 - \rho_2}$ (see also (3.8)). To check which constraint is active the following calculations for P_{LB}^* are performed:

$$\begin{aligned} P_{LB}^* &> \frac{\sigma}{1 - \rho_1 - \rho_2} \\ (P_{LB}^*)^2 &> \frac{\sigma^2}{(1 - \rho_1 - \rho_2)^2} \end{aligned}$$

Substitution of the second solution of (4.31) results in:

$$\frac{c_2 \lambda_2 \sigma^2}{c_1 \lambda_1 (1 - \rho_1)(1 - \rho_2) + c_2 \lambda_2 \rho_1^2} > \frac{\sigma^2}{(1 - \rho_1 - \rho_2)^2}$$

For the denominators hold that:

$$\begin{aligned} \frac{c_1 \lambda_1}{c_2 \lambda_2} ((1 - \rho_1)(1 - \rho_2)) + \rho_1^2 &< (1 - \rho_1 - \rho_2)^2 \\ \frac{c_1 \lambda_1}{c_2 \lambda_2} &< \frac{(1 - \rho_1 - \rho_2)^2}{((1 - \rho_1)(1 - \rho_2)) + \rho_1^2} = \frac{1 - 2\rho_1 - \rho_2}{1 - \rho_1} \end{aligned}$$

Due to the assumption $c_1\lambda_1 > c_2\lambda_2$, $\frac{c_1\lambda_1}{c_2\lambda_2} > 1$ so:

$$1 < \frac{1 - 2\rho_1 - \rho_2}{1 - \rho_1}$$

Finally the following inequality has to hold:

$$1 < 1 - \rho_1 - \rho_2 \quad (4.33)$$

Equation (4.33) can never be satisfied. The constraint is inactive. When a similar computation is performed for P_{UB}^* . The result is:

$$1 - \rho_1 - \rho_2 < 1 \quad (4.34)$$

Equation (4.34) is always satisfied. This makes the upper bound, in case of the assumption that $c_1\lambda_1 > c_2\lambda_2$ the active constraint. The results are:

$$\begin{aligned} \tau_1^\# &= P(1 - \rho_2) - \sigma \\ P = P_{UB}^* &= \frac{c_1\lambda_1\sigma}{\sqrt{(c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2)c_1\lambda_1}}. \end{aligned} \quad (4.35)$$

4. Verification of the time averaged weighted wip level.

The values for τ_1 and P have been determined in the previous step. With the earlier made assumptions and substitution of (4.35) the following time averaged weighted wip level is determined:

$$\begin{aligned} J &= c_1 \frac{1}{2P} \frac{\mu_1 \bar{\rho}_1}{\phi_1 - \rho_1} (\phi_1 P - \tau_1)^2 + c_2 \frac{1}{2P} \frac{\mu_2 \rho_2}{1 - \rho_2} (\sigma_{21} + \tau_1 + \sigma_{12})^2 \\ &= c_1 \frac{1}{2P} \frac{\lambda_1}{1 - \rho_1} (P - \tau_1)^2 + c_2 \frac{1}{2P} \frac{\lambda_2}{1 - \rho_2} (\tau_1 + \sigma)^2 \\ &= \frac{1}{2P} \left(\frac{c_1\lambda_1}{1 - \rho_1} (\rho_2 P + \sigma)^2 + c_2\lambda_2(1 - \rho_2)P^2 \right). \end{aligned}$$

Using Matlab for the substitution of P results in:

$$J = \sigma \frac{c_1\lambda_1\rho_2\sqrt{C} + C}{(1 - \rho_1)\sqrt{C}} \quad \text{where:} \quad C = (c_2\lambda_2(1 - \rho_1)(1 - \rho_2) + c_1\lambda_1\rho_2^2)c_1\lambda_1.$$

The cost function (7) in [Eek06a] has to equal the found expression for J . The theory in this report deals with systems which contain always a slow mode (Lemma 4.1.4). For the theory in [Eek06a] this means determination of variable α with the positive real root of equation (9). Next, the variable α is substituted in the equation of the time averaged weighted wip level (7) in [Eek06a]:

$$\alpha = \frac{-\left(c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2\right) - \sqrt{(1-\rho_1-\rho_2)^2 C}}{\left(c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2\right)(1-\rho_1)}.$$

Substitution of α in the cost function (performed in Matlab) results in:

$$J_{\text{ACC}} = \sigma \frac{c_1 \lambda_1 \rho_2 \sqrt{C} + C}{(1 - \rho_1) \sqrt{C}}$$

$$\text{where: } C = (c_2 \lambda_2 (1 - \rho_1) (1 - \rho_2) + c_1 \lambda_1 \rho_2^2) c_1 \lambda_1.$$

Both cost functions deliver the same result ($J = J_{\text{ACC}}$) if $c_1 \lambda_1 > c_2 \lambda_2$ holds and only a slow mode of type 1 occurs in the optimal steady state process cycle.

Intermezzo: Multiple cycles in one period

The optimal trajectories found in this chapter, are the result of equalize lengths for period P and the duration of one process cycle T . During one period type 1 and type 2 are processed once. The goal of the optimization is to minimize the weighted mean wip level of the total buffer levels within one period.

If the sum of time fractions that each product needs to be processed in a workstation is low, it is possible enough time is available to process type 1 and type 2 twice during one period in order to minimize the total mean wip level of the system even further. To keep a stable system the following inequality has to be met:

$$P > N \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}. \quad (4.36)$$

Equality 4.36 is based on (3.9). New parameter $N \in \mathbb{N}$ represents the number of cycles that are performed in one period, (without violating the inequality). The advantage of processing each type N times are lower buffer levels. The disadvantage is the workstation setups a factor N longer. An example is introduced to show the influence of the weighted mean wip level of the system when multiple cycles in one period occur.

Example 4.4.1. Assume a workstation with the following parameter setting:

$\bar{\rho}_1$:	0.3	-	c_1 :	1	-
ρ_2 :	0.25	-	c_2 :	1	-
μ_1 :	1	lots/hr.	P :	1000	hrs.
μ_2 :	1	lots/hr.	ϕ_1 :	0.5	-
σ_{12} :	50	hrs.			
σ_{21} :	50	hrs.			

Table 4.1: Parameter setting.

When computing the weighted mean wip level of the workstation in this parameter setting with a continuous approximation model. The model is simulated in Matlab where the time averaged weighted wip level of the steady state process cycle is determined.

The time averaged weighted wip levels result in:

$$J_{(N=1)} = 49.1$$

$$J_{(N=2)} = 43.3.$$

Note: The computed value for $J_{(N=2)}$ is not proven to be optimal.

The example shows that multiple cycles during one period can lead to a better time averaged weighted wip level of the system.

Further analysis of systems with multiple cycles during one period is beyond the scope of this report. But it forms an interesting research area in the search for lower time averaged weighted wip levels during one period.

At this point all situations/scenarios have been discussed with respect to a two product workstation with one constant and one piecewise constant arrival pattern. Different shapes for all the situations and scenarios have been discussed and explained that process type 1 and type 2 during one period P .

The essential differences between the trajectories depend on the position of the slow modes. Or more precisely, which product is processed in a slow mode. Situation I and II can be classified in to three groups by sorting the slow modes:

1. For situation I only a slow mode for type 2 always exists. (*Situation I*)
2. For situation II where type 2 is processed relatively long, the optimal process cycle has at least a slow mode for type 2. (*Situation II-a*)
3. For situation II where type 1 is processed relatively long, the optimal process cycle has at least a slow mode for type 1. (*Situation II-b*)

With this sorting the three scenarios of situation II and situation I are reduced to three different optimized problems where in both situation II-a and II-b two slow modes can occur. In the next chapter feedback controllers are defined for these three possible situations. These feedback controllers have to steer the system from an arbitrary point, with respect to buffer levels and time, to its desired/optimal trajectory.

Chapter 5

Feedback control

A two product workstation with one constant arrival rate and one piecewise constant arrival rate has been discussed in Chapter 4. The result is an optimal steady state process cycle for a system which meets the conditions:

- $\bar{\rho}_1 + \rho_2 < 1$.
- $P = T > \frac{\sigma_{12} + \sigma_{21}}{1 - \bar{\rho}_1 - \rho_2}$.

For all possible parameter settings that satisfy these inequalities, Chapter 4 provides an optimal steady state process cycle with respect to minimal weighted mean wip level. Each optimal process cycle describes the ‘desired behavior’ of the buffer levels in a workstation. It is this behavior which results in a minimal wip level for the workstation. In practice workstations are exposed to disruptions. Disturbances like changing arrival rates, machine breakdowns or other in- or external factors can influence the buffer levels of the workstation. Therefore, it is unlikely a workstation starts processing on its desired trajectory and stays on this trajectory. By using a state feedback controller the system is steered to the desired trajectory, regardless of the initial buffer levels or point in time. The trajectories for all possible parameter settings are classified in to three groups as mentioned at the end of Chapter 4.

1. Situation I (always a slow mode for type 2).
2. Situation II-a (at least a slow mode for type 2)
3. Situation II-b (at least a slow mode for type 1)

Dependent on the parameter setting one of these three situations is valid. In this chapter a state feedback controller is proposed for all three situations. In each section of this chapter a controller is presented that deals with one of these three situations.

In this chapter, each section starts with a description of the desired trajectory. The

state of the system (3.4) is determined for different points along the trajectory. These states are the reference where the controller has to converge to. Next, a controller is proposed that has to steer to the optimal trajectory. Finally a proof of convergence is obtained that shows convergence of a system which starts with arbitrary buffer levels at any point in time.

5.1 Feedback control of situation I ($\hat{\lambda}_1 \geq \mu_1$)

A two product workstation in a factory has to deal with undesired behavior caused by disturbances. In general a workstation does not process at the desired trajectory. Therefore, a controller is needed that steers the system to this desired (optimal) trajectory. In this case a controller is proposed for situation I. The most important feature of the situation is the short time span where products of type 1 arrive at a high rate ($\hat{\lambda}_1 \geq \mu_1$). The controller has to steer the system to a trajectory as presented in Figure 5.1 in the previous chapter. The state of this system at different points on the desired trajectory are determined after processing type 1, the setup to type 2, processing type 2 at μ_2 , processing type 2 at λ_2 and after the setup to type 1, respectively Mode 1–5:

$$\begin{aligned}
 \text{After Mode 1:} & \quad \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ \lambda_2(\sigma_{21} + \bar{\rho}_1 P) \\ P \end{bmatrix} \\
 \text{After Mode 2:} & \quad \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ \lambda_2(\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}) \\ P - \sigma_{12} \end{bmatrix} \\
 \text{After Mode 3:} & \quad \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ P - \sigma_{12} - \frac{\lambda_2(\sigma_{21} + \bar{\rho}_1 P + \sigma_{12})}{\mu_2 - \lambda_2} \end{bmatrix} \\
 \text{After Mode 4:} & \quad \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \phi_1 P + \sigma_{21} \end{bmatrix} \\
 \text{After Mode 5:} & \quad \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ \lambda_2 \sigma_{21} \\ \phi_1 P \end{bmatrix}.
 \end{aligned}$$

Note, x_0 and m are not taken into account in the state of the system. Due to the dynamics, discussed in Chapter 3, both parameters always have the same value after each mode and do not influence the path to the desired trajectory. In Figure 5.1 the states of the system after each mode are visualized.

When the system runs and after one of its five modes the accompanying values for x_1 , x_2 and Δ do not have the desired value, the system does not operate on its desired trajectory. To steer from an arbitrary point in time and arbitrary buffer levels to the desired trajectory, the following controller is proposed:

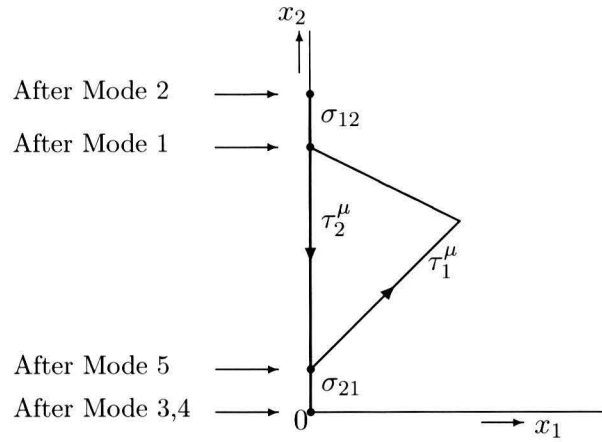


Figure 5.1: Periodic orbit of situation I.

Proposition 5.1.1. *The following state feedback control law brings the system with $\hat{\lambda}_1 \geq \mu_1$ to the desired trajectory.*

$$(u_0, u_1, u_2) = \begin{cases} (\mathbf{1}, 0, 0) & \text{if } m = 1, x_0 > 0 \\ (\mathbf{1}, \mu_1, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0 \\ (\mathbf{2}, 0, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0 \\ (\mathbf{2}, 0, 0) & \text{if } m = 2, x_0 > 0 \\ (\mathbf{2}, 0, \mu_2) & \text{if } m = 2, x_0 = 0, x_2 > 0, \Delta > \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21} \\ (\mathbf{2}, 0, \lambda_2) & \text{if } m = 2, x_0 = 0, x_2 = 0, \Delta > \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21} \\ (\mathbf{1}, 0, 0) & \text{if } m = 2, x_0 = 0, \Delta \leq \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21} \end{cases} \quad (5.1)$$

Remark 5.1.2. An informal description of this controller is:

- Mode 1: $\mathbf{1}$ as long as $x_1 > 0$; go to Mode 2.
- Mode 2: perform $\mathbf{2}$, after σ_{12} go to Mode 3.
- Mode 3: $\mathbf{2}$ at μ_2 as long as both $x_2 > 0$ and $\Delta > \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$; go to Mode 4.
- Mode 4: $\mathbf{2}$ at λ_2 as long as both $x_2 = 0$ and $\Delta > \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$; go to Mode 5.
- Mode 5: perform $\mathbf{1}$, after σ_{21} go to Mode 1.

Dependent on the state of the system, the controller starts in one of the five modes. The initial state always suits one of the modes. From that mode the controller starts. Mode 3 and Mode 4 might have a duration of zero initially.

Assume the n^{th} start after \bullet . The superscript (n) represents the number of the process cycle that takes place.

Before proving that the controller in Proposition 5.1.1 converges the system towards the desired behavior, first two lemmas are formulated. The first step is to prove that the system will process type 2. The next step is to prove convergence of buffer level 2 to its desired trajectory.

Lemma 5.1.3. *The system eventually processes type 2.*

Proof. To process type 2, $\Delta > \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$ after setup \bullet (using Mode 3 of Proposition 5.1.1). Assume that $\Delta < \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$ after setup \bullet in each loop.

When Mode 1 is finished for the first time: $x_1^{(1)} = 0$ and $\Delta^{(1)} > \phi_1 P$. As long as $\Delta < \phi_1 P$ products of type 1 arrive at a higher rate than they can be processed, so $x_1 = 0$ can occur only if no products arrive. After setup \bullet , two situations can occur:

1. $x_1 = \hat{\lambda}_1(\phi_1 P - \max(\Delta - \sigma_{12}, 0))$ if $\Delta^{(1)} < \phi_1 P + \sigma_{12}$.
2. $x_1 = 0$ if $\Delta^{(1)} \geq \phi_1 P + \sigma_{12}$.

In both situations $\Delta^{(1)} < \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$ after setup \bullet can still hold. The duration of one loop without processing type 2 is $\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}$. This means the updated $\Delta^{(2)}$ after one loop is:

$$\Delta^{(2)} = \Delta^{(1)} + P - (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}).$$

Note that (3.9) shows: $P - (\sigma_{21} + \bar{\rho}_1 P + \sigma_{12}) > 0$. This results in a linear increase for Δ as long as the iteration count of loops increases. Therefore, it becomes impossible to keep $\Delta^{(n)} < \phi_1 P + \frac{x_1}{\mu_1} + \sigma_{21}$. So type 2 has to be processed. \square

The second step is to prove that the buffer level of type 2 reaches its desired trajectory.

Lemma 5.1.4. *Eventually type 2 is processed in a slow mode and the desired trajectory is reached.*

Proof. When the system processes type 2, the controller already has synchronized the arrival pattern of type 1 with the processing of type 1. This implies that after setup \bullet variables Δ and x_1 follow the desired trajectory:

$$([x_1, x_2, \Delta]^T)^{(n)} = ([0, x_2, P - \sigma_{12}]^T)^{(n)}.$$

Variable x_2 is still unknown. Because x_1 and Δ are at their desired trajectory, the machine has τ_2 hours to process type 2 during each loop. Each loop the machine needs $\rho_2 P$ hours to process type 2. When the buffer level of type 2 is high enough and a slow

mode for type 2 exists, the machine is able to process $\tau_2^\lambda(\mu_2 - \lambda_2)$ products more than needed each loop. The result is a buffer level that decreases until it becomes zero:

$$x_2^{(n+1)} = \max(x_2^{(n)} - \tau_2^\lambda(\mu_2 - \lambda_2), 0). \quad (5.2)$$

Equation (5.2) shows a decrease in the buffer level of type 2 if the number of loops increases. Eventually the buffer level has to become zero due to Lemma 4.1.4. The lemma demands at least one slow mode in the optimal process cycle. A slow mode for type 1 is not possible which implies $\tau_2^\lambda > 0$. From the moment the buffer level becomes zero, the machine starts processing type 2 in a slow mode and the desired trajectory is reached. Note, if the buffer level of type 2 is initially low and the machine starts processing type 2, the machine can process type 2 in a slow mode as long as $\Delta \leq \phi_1 P + \sigma_{21}$ holds. Although a slow mode for type 2 occurs, the machine has not reached its desired trajectory yet. The desired trajectory is reached when a slow mode for type 2 is reached after type 1 is processed for at least one time. \square

In short, the controller steers the system from an arbitrary point in time with arbitrary buffer levels to the desired (optimal) trajectory. It first processes type 1 until its buffer is empty. Then type 2 is processed until a setup is needed to let the start of the arrivals of type 1 coincides with the start of processing type 1 (Lemma 4.2.4).

5.2 Feedback control of situation II-a ($\hat{\lambda}_1 < \mu_1$)

The optimal trajectory in this situation has at least a slow mode for type 2. Depending on the parameter setting, processing type 1 in a slow mode is possible. With the option of possibly two slow modes in the desired trajectory, the controller has six modes. These modes are respectively processing type 1 at μ_1 , processing type 1 at $\hat{\lambda}_1$, a setup to type 2, processing type 2 at μ_2 , processing type 2 at λ_2 and a setup to type 1. The state of the Modes 1–6 are:

$$\begin{aligned}
\text{After Mode 1:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ (\sigma_{21} + \tau_1^\mu) \lambda_2 \\ \phi_1 P - \sigma_{21} - \tau_1^\mu \end{bmatrix} \\
\text{After Mode 2:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ (\sigma_{21} + \tau_1^\mu (1 - \frac{\mu_1}{\lambda_1}) + \phi_1 P) \lambda_2 \\ P \end{bmatrix} \\
\text{After Mode 3:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} \max(0, \phi_1 P - P + \sigma_{12}) \hat{\lambda}_1 \\ (\sigma_{21} + \tau_1^\mu (1 - \frac{\mu_1}{\lambda_1}) + \phi_1 P + \sigma_{12}) \lambda_2 \\ P - \sigma_{12} \end{bmatrix} \\
\text{After Mode 4:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} \max \left(0, \phi_1 P - P + \sigma_{12} + \frac{(\sigma_{21} + \tau_1^\mu (1 - \frac{\mu_1}{\lambda_1}) + \phi_1 P + \sigma_{12}) \lambda_2}{\mu_2 - \lambda_2} \right) \hat{\lambda}_1 \\ 0 \\ P - \sigma_{12} - \frac{(\sigma_{21} + \tau_1^\mu (1 - \frac{\mu_1}{\lambda_1}) + \phi_1 P + \sigma_{12}) \lambda_2}{\mu_2 - \lambda_2} \end{bmatrix} \\
\text{After Mode 5:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} x_1^\# \\ 0 \\ \phi_1 P - \frac{x_1^\#}{\lambda_2} \end{bmatrix} \\
\text{After Mode 6:} \quad & \begin{bmatrix} x_1 \\ x_2 \\ \Delta \end{bmatrix} = \begin{bmatrix} x_1^\# + \hat{\lambda}_1 \sigma_{21} \\ \lambda_2 \sigma_{21} \\ \phi_1 P - \frac{x_1^\#}{\lambda_1} - \sigma_{21} \end{bmatrix}.
\end{aligned}$$

Here τ_1^μ is the length the machine processes type 1 at a rate of μ_1 and $x_1^\#$ is the critical value for the number of products of type 1 in the buffer:

$$\begin{aligned}
\tau_1^\mu &= \frac{x_1^\# + \hat{\lambda}_1 \sigma_{21}}{\mu_1 - \hat{\lambda}_1}, \text{ and} \\
x_1^\# &= \frac{(\phi_1 P - \tau_1^\#) \bar{\rho}_1}{\phi_1 - \bar{\rho}_1} (\mu_1 - \hat{\lambda}_1) - \hat{\lambda}_1 \sigma_{21}.
\end{aligned}$$

where: $\tau_1^\#$ is the optimal process time of type 1 after the constrained optimization of (4.24).

The position of the states after each mode in the periodic orbit are presented in Figure 5.2.

Proposition 5.2.1. *The following state feedback control law brings the system with $\hat{\lambda}_1 < \mu_1$ and a slow mode for type 2 to the desired process cycle.*

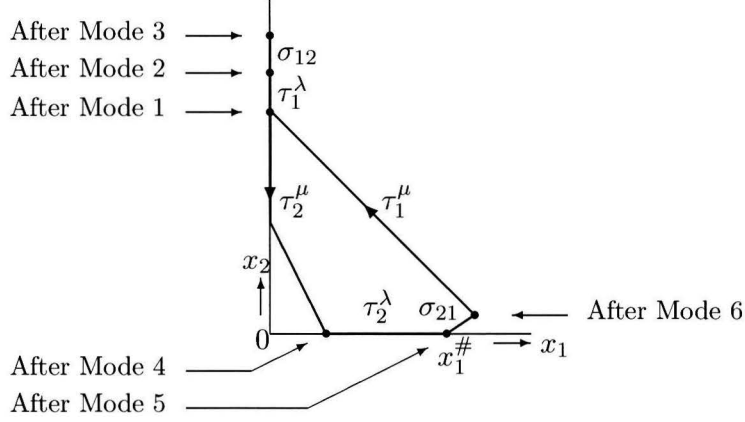


Figure 5.2: Periodic orbit of situation II-a.

$$(u_0, u_1, u_2) = \begin{cases} (\mathbf{1}, 0, 0) & \text{if } m = 1, x_0 > 0 \\ (\mathbf{1}, \mu_1, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0 \\ (\mathbf{1}, \lambda_1, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, \Delta \leq \phi_1 P \\ (\mathbf{2}, 0, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, \Delta > \phi_1 P \\ (\mathbf{2}, 0, 0) & \text{if } m = 2, x_0 > 0 \\ (\mathbf{2}, 0, \mu_2) & \text{if } m = 2, x_0 = 0, x_1 < x_1^\#, x_2 > 0 \\ (\mathbf{2}, 0, \lambda_2) & \text{if } m = 2, x_0 = 0, x_1 < x_1^\#, x_2 = 0 \\ (\mathbf{1}, 0, 0) & \text{if } m = 2, x_0 = 0, x_1 \geq x_1^\# \end{cases} \quad (5.3)$$

Remark 5.2.2. An informal description of this controller is:

- Mode 1: $\mathbf{1}$ as long as $x_1 > 0$; go to Mode 2.
- Mode 2: $\mathbf{1}$ as long as both $x_1 = 0$ and $\Delta \leq \phi_1 P$; go to Mode 3.
- Mode 3: perform $\mathbf{2}$, after σ_{12} go to Mode 4.
- Mode 4: $\mathbf{2}$ at μ_2 as long as both $x_2 > 0$ and $x_1 < x_1^\#$; go to Mode 5.
- Mode 5: $\mathbf{2}$ at λ_2 as long as both $x_2 = 0$ and $x_1 < x_1^\#$; go to Mode 6.
- Mode 6: perform $\mathbf{1}$, after σ_{21} go to Mode 1.

A new parameter is introduced in Proposition 5.2.1 and Remark 5.2.2. New parameter $x_1^\#$ denotes the critical value for the number of products of type 1 in the buffer. The value is determined with the use of the optimal length for τ_1 and Lemma 4.2.5. When $m = 2$ and this value is reached, the workstation has to start $\mathbf{1}$ immediately. Its value is determined with:

$$x_1^\# = \tau_1^\mu(\mu_1 - \hat{\lambda}_1) - \hat{\lambda}_1 \sigma_{21} \quad (5.4)$$

To prove convergence first the buffer level of type 1 is steered to its desired trajectory.

Lemma 5.2.3. *When type 1 is processed for the second time, the buffer level of type 1 follows the desired trajectory.*

Proof. When type 1 is processed for the first time and the buffer is empty, two situations can occur:

1. $\Delta \leq \phi_1 P$; Switch to Mode 2 and processes type 1 in slow mode until $\Delta = 0$.
2. $\Delta > \phi_1 P$; Switch to Mode 3.

In the first situation the arrival pattern of type 1 can be synchronized with the processing of type 1 (Lemma 4.2.5). In this situation type 1 follows its desired trajectory already after the first time its processed.

In the second situation the controller switches to Mode 3 immediately. After the setup in Mode 3, the controller switches to process type 2 (Mode 4 and 5). The controller stays in one of these modes (depending on the buffer level x_2) as long as $x_1 < x_1^\#$. When $x_1 \geq x_1^\#$, the controller switches to ① and then to ②, where the end of arrivals of type 1 coincides with the end of processing type 1. After the second time type 1 is processed, x_1 and Δ are at their desired trajectory because the timing with $x_1^\#$ is based on Lemma 4.2.5. \square

After ② is completed for the second time and no slow mode for type 2 has occurred yet, type 1 is processed on the desired trajectory but type 2 may not.

Lemma 5.2.4. *The desired trajectory is reached if type 1 is steered to its desired trajectory and type 2 is processed in slow mode.*

Proof. In Lemma 5.2.3 type 1 is steered to its desired trajectory. This means Δ and x_1 follow the desired path. Only x_2 is still unknown. The characteristic of situation II-a is the presence of a slow mode of type 2. The desired trajectory contains a slow mode what implies the machine is able to process more products than needed when the buffer level for type 2 is high. During one time interval of τ_2 the machine is able to process more products for the duration of the slow mode (τ_2^λ). The machine can process $\tau_2^\lambda(\mu_2 - \lambda_2)$ products more than needed during one desired trajectory. The buffer level of type 2 decreases each loop with $\tau_2^\lambda(\mu_2 - \lambda_2)$ products, unless the buffer is emptied earlier:

$$x_2^{(n+1)} = \max(x_2^{(n)} - \tau_2^\lambda(\mu_2 - \lambda_2), 0).$$

If the buffer is empty, the workstation keeps on processing type 2 in slow mode until the critical value for type 1 ($x_1 = x_1^\#$) is reached and the workstation needs to switch to type 1. After the first time a slow mode for x_2 occurs ($x_2 = 0$), all variables (x_1, x_2, Δ)

follow their desired trajectory what makes the system follow the desired trajectory. If the initial buffer level of type 2 is low, the machine processes type 2 in a slow mode as long as $x_1 < x_1^\#$. \square

In short, the controller lets the workstation empty the buffer of type 1. Next, the workstation processes type 2 until the buffer level of type 1 reaches the value $x_1^\#$. If the value is reached the workstation switches to \bullet immediately. During the time interval where $x_1 < x_1^\#$ the workstation has to process type 2.

5.3 Feedback control of situation II-b ($\hat{\lambda}_1 < \mu_1$)

The third controller to be discussed, steers a workstation which has at least a slow mode for type 1. The approach of finding the controller is similar to that of the first two controllers. First the desired trajectory is determined. The desired trajectory is reached if buffer levels and Δ after leaving the Modes 1–6 have the same values as mentioned in the feedback control of situation II-a.

The position of the state of the system after each mode in the periodic orbit are presented in Figure 5.3.

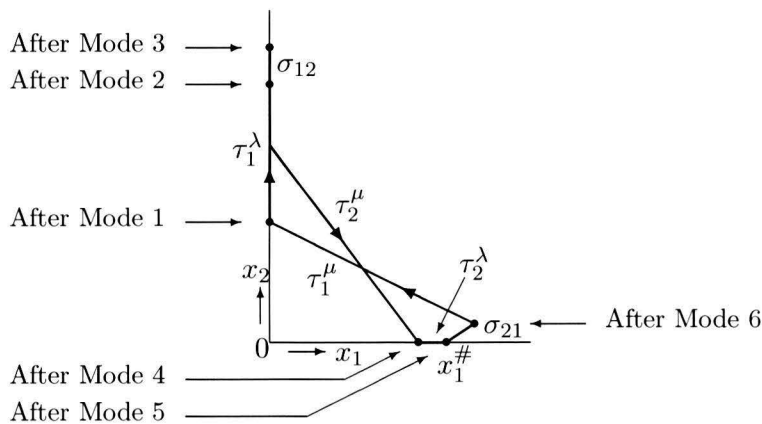


Figure 5.3: Periodic orbit of situation II-b.

Proposition 5.3.1. *The following state feedback control law brings the system with $\hat{\lambda}_1 < \mu_1$ and at least a slow mode for type 1 to the desired process cycle.*

$$(u_0, u_1, u_2) = \begin{cases} (\mathbf{1}, 0, 0) & \text{if } m = 1, x_0 > 0 \\ (\mathbf{1}, \mu_1, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0, \Delta \leq \phi_1 P \\ (\mathbf{1}, \lambda_1, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, \Delta \leq \phi_1 P \\ (\mathbf{2}, 0, 0) & \text{if } m = 1, x_0 = 0, \Delta > \phi_1 P \\ (\mathbf{2}, 0, 0) & \text{if } m = 2, x_0 > 0 \\ (\mathbf{2}, 0, \mu_2) & \text{if } m = 2, x_0 = 0, x_2 > 0 \\ (\mathbf{2}, 0, \lambda_2) & \text{if } m = 2, x_0 = 0, x_1 < x_1^\#, x_2 = 0 \\ (\mathbf{1}, 0, 0) & \text{if } m = 2, x_0 = 0, x_1 \geq x_1^\#, x_2 = 0 \end{cases} \quad (5.5)$$

Remark 5.3.2. An informal description of this controller is:

- Mode 1: $\mathbf{1}$ as long as both $x_1 > 0$ and $\Delta \leq \phi_1 P$; go to Mode 2.
- Mode 2: $\mathbf{1}$ as long as both $x_1 = 0$ and $\Delta \leq \phi_1 P$; go to Mode 3.
- Mode 3: perform $\mathbf{2}$, after σ_{12} go to Mode 4.
- Mode 4: $\mathbf{2}$ at μ_2 as long as $x_2 > 0$; go to Mode 5.
- Mode 5: $\mathbf{2}$ at λ_2 as long as both $x_2 = 0$ and $x_1 < x_1^\#$; go to Mode 6.
- Mode 6: perform $\mathbf{1}$, after σ_{21} go to Mode 1.

To prove the convergence for a system with at least a slow mode for type 1, the proof starts with synchronization of the piecewise constant arrival rate of type 1 with processing type 1.

Lemma 5.3.3. *The system eventually processes type 1.*

Proof. When the machine starts processing type 2 it stops when buffer 2 is empty and the buffer level of type 1 is larger or equal to $x_1^\#$. When a setup to type 1 is performed, two situations can occur:

1. $\Delta < \phi_1 P$; Type 1 can be processed and the Lemma is proven.
2. $\Delta \geq \phi_1 P$; Type 1 is not processed because no products arrive. The machine immediately switches, performs a setup to type 2 and starts processing type 2. After a time span of $\rho_2(\sigma_{21} + \sigma_{12})$ buffer 2 is empty again and a setup to type 1 is performed.

When option 2 is performed the value of Δ determines if type 1 is processed or option 2 has to be performed again. It is only when the time span of option 2 equals a period P that option 2 stays valid. In that specific situation Δ has the same value each loop. To prove option 1 has to occur, the time span in option 2 has to be unequal with P . This is always true because the maximum time span for processing type 2 and perform two setups is always smaller than the length of P . Eventually option 1 occurs. \square

The machine always has a situation where at the start of processing type 1, $\Delta \leq \phi_1 P$. The end of processing type 1 is reached if products of type 1 stop arriving ($\Delta = 0$). It is possible buffer 1 is not empty but the machine performs a setup anyway. At this point in time, the end of the arrival of products of type 1 is synchronized with the end of processing type 1. The next step is to prove that both buffer levels end up at the desired trajectory.

Lemma 5.3.4. *The desired trajectory is reached after a slow mode of type 2 has occurred.*

Proof. In Lemma 5.3.3 the processing of type 1 is synchronized with the arrival rate of type 1. One of the characteristics of situation II-b is the presence of a slow mode for type 1. The presence of the slow mode of type 1 in the desired trajectory means that the system can process more products during one process cycle if the system has initially a high buffer level for type 1. The extra number of products that can be processed during one process cycle equals the length of the slow mode in the desired trajectory multiplied with the process rate minus the arrival rate ($\tau_1^\lambda(\mu_1 - \hat{\lambda}_1)$). If the initial buffer level of type 1 is low, the machine processes in slow mode after emptying the buffer level. After buffer 2 is emptied once and type 1 is processed in slow mode for the first time, the exact buffer level of type 2 is still unknown. The system performs setup ② and empties buffer 2. During processing type 2, products of type 1 arrive. When buffer level 2 is empty, three situations can occur:

- $x_1^{(n)} < x_1^\#$; The machine processes type 2 in slow mode until $x_1^{(n)} = x_1^\#$. The desired trajectory is reached.
- $x_1^{(n)} = x_1^\#$; In this special situation no slow mode for type 2 occurs. Nevertheless the desired trajectory is reached.
- $x_1^{(n)} > x_1^\#$; The machine processed type 2 ‘too long’ with respect to the desired trajectory. The machine performs setup ① and processes type 1. Because type 2 was processed ‘too long’, less time is available before $\Delta = 0$. Although there is less time, buffer 1 is emptied before $\Delta = 0$ (Due to the fact both buffers are emptied before and $\tau_1 + \tau_2 > (\bar{\rho}_1 + \rho_2)P$). When $\Delta = 0$, a setup ② is performed. Type 1 needed ‘less time’ to be processed with respect to the desired trajectory because the machine stayed less time processing in a slow mode. The result is a lower buffer level at the start of processing type 2 again. When the buffer level of type 2 is empty, the buffer level of type 1 has decreased with respect to the previous cycle $x_1^{(n+1)} < x_1^{(n)}$.

If the third situation occurs, the system repeats this behavior until eventually $x_1^{(n+1)} \leq x_1^\#$ occurs. Then one of the first two situation holds, meaning the desired trajectory is reached. \square

Remark 5.3.5. For the proof the convergence of the system with the proposed controllers the synchronization of the arrival rate of type 1 and processing type 1 plays an important role. When the situation occurs when no slow modes occur ($P = T = T_{\min}$) it becomes difficult to steer the system. If buffer levels are high a decrease of the buffer levels in combination with keeping the synchronization is only possible when the non steady state process cycle has a length that equals a multiple duration of P .

For all possible parameter settings within the scope of this thesis, a state feedback control law is obtained. All controllers steer the system from an arbitrary point in time and arbitrary buffer levels, to the desired trajectory. In the next chapter, each controller is tested in two simulations. The first simulation is the controller in a hybrid fluid approximation model. The second simulation is performed in a discrete event model with stochastic inter-arrival times and stochastic process times.

Chapter 6

Simulation experiments

In the previous chapter three feedback control laws are proposed and shown to converge to the desired trajectory as determined in Chapter 4. To display the convergence of the system to its desired trajectory, a simulation is performed. The controller is implemented in a hybrid fluid model. With the use of this model, the optimal process cycles and trajectories are computed and compared with the optimized trajectories of Section 4.4. The results will show the controller steers the system exactly to the optimal steady state process cycle as determined in Chapter 4. However, in a real production system the inter-arrival times and process times are never constant. For that reason a discrete event simulation is performed. The simulation model includes stochastic behavior on the inter-arrival times of both product types and both process times. The controller needs to show that it is able to deal with these disturbances. The simulation results of the three controllers presented in Chapter 5 are discussed in this chapter. In the first section the simulation models are presented. In the other sections the results for each controller are discussed separately.

6.1 Simulation models

6.1.1 Hybrid fluid model

The controllers are designed with a hybrid fluid model. The controllers presented in Chapter 5 are described in five or six different modes. By following these modes, the controller steers the system to the desired trajectory. Translation of these modes into a simulation model results in a model, (made with Matlab version 2006b), with five or six cases. Each case represents a mode of the controller. Initially the system has an arbitrary state. The values of this state determine in which case the machine starts. When a case becomes active, Δ and both buffer levels are analyzed. With this information, the model calculates the time the machine is able to stay in the mode its in. The model then calculates Δ and the buffer levels at which the mode will be left. Then the

controller switches to the next case and the process is repeated. The constraints each case hold are similar to the constraints mentioned in the proposition of each controller in Chapter 5. A more detailed explanation of the simulation in Matlab is presented in Appendix A.

6.1.2 Discrete event model

The second test that is performed uses a discrete event simulation of the workstation with the controller discussed in the previous chapter. In case of a discrete event manufacturing workstation, a discrete event model gives a better representation of a manufacturing system than the hybrid fluid approximation model. Within the discrete event model mentioned here, stochastic variables are introduced for process times and inter-arrival times. The arrivals and processing products are assumed to follow a Poisson process [Mon99]. This characteristic makes it possible to apply an exponential distribution for the inter-arrival times between products and all process times. The discrete event model is modeled in χ 0.8 [Bee00], [Hof02]. In the model different processes are defined. The iconic representation of the system is visualized in Figure 6.1. In Figure 6.1 all capital letters represent a process. Here, the generators are G_1 and G_2 .

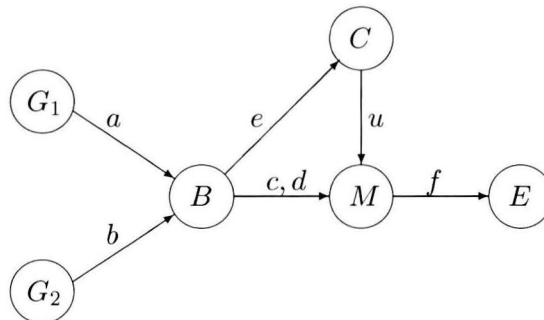


Figure 6.1: Iconic model of a two product workstation

These processes simulate the arrival patterns of type 1 and type 2 respectively. Process B is the buffer which receives products from the generators over channels a and b . The buffer contains two buffers where type 1 and type 2 are stored separately. The buffer sends the products to the machine M . The machine M receives products of type 1 or 2 from the buffer over channel c or d . The controller C receives the number of products from the buffer. The controller uses the buffer level information to establish an input signal u and sends that to the machine. When the machine has processed a product, the product leaves the system at the exit process E . Although different controllers are applied, the structure as presented in Figure 6.1 and all processes are maintained during all simulations. All processes (except the controllers) are discussed now. The controllers are discussed for each situation separately together with the results. The total χ -codes

of all three discrete event models in ASCII is presented in Appendix B.

type lot = nat (X-1)

```

proc G1(a: !lot, λ1, φ1, P: real) =
  || r, s : real, t : →real
  | t := exponential(1/λ1)
  ; * [ true → r := rmod(τ, P); s := σt
        ; [ r + s < φ1P → Δs; a!1
            || r + s ≥ φ1P → Δ(P - r)
          ]
        ]
  ]
  ]

```

(X-2)

Process G_1 (X-2) generates a piecewise constant arrival pattern. The process determines first its current point in time within one process cycle (r). Next, a sample for the inter-arrival time is taken from an exponential distribution and added to r . If the sum of these time spans is less than the time interval where products can leave the generator, the generator waits the time span of the sample and then sends it to the buffer. If the sum of time spans is larger than the time interval where lots can leave, the generator waits until a new period starts where it can send lots again.

Generator G2 (X-3) sends lots of type 2 to the buffer:

```

proc G2(a: !lot, λ2: real) =
  || t : →real
  | t := exponential(1/λ2)
  ; * [ true → a!2; Δσt ]
  ]

```

(X-3)

The generator sends a lot to the buffer and waits the time span of a sample taken from an exponential distribution. The time interval represents the inter-arrival time between two lots. After waiting for a time span t , again the generator sends a lot.

In the discrete event model only one buffer is modeled.

```

proc B(a, b: ?lot, c, d: !lot, e: !nat2, ini1, ini2: nat) =
  || x1, x2 : lot, xs1, xs2 : nat
  | xs1 := ini1; xs2 := ini2
  ; e!⟨xs1, xs2⟩
  ; * [ true → [ true; a?x1 → xs1 := xs1 + 1
                  || true; b?x2 → xs2 := xs2 + 1
                  || xs1 > 0; c!x1 → xs1 := xs1 - 1
                  || xs2 > 0; d!x2 → xs2 := xs2 - 1
                ]
        ; e!⟨xs1, xs2⟩; fileout ("bufferize.txt")!τ, "\t", xs1, "\t", xs2, "\n"
  ]
  ]

```

(X-4)

The initialization of the buffer is to fill both buffers at a desired initial level and send the number of lots in the buffers to the controller. After the initialization the buffer receives lots from generator G_1 and G_2 over channels a and b and sends lots to the machine over channel c or d (depending on the type of lot) as long as there are lots available. If buffer levels change, the new buffer levels are passed on to the controller and the new buffer levels and the corresponding time are stored in a text-file. The buffer sends, if possible, products to the machine.

```

proc M(a, b: ?lot, u: ?nat, c: !lot,  $\mu_1, \mu_2, \sigma_{12}, \sigma_{21}$ : real) =
  [ [ x : lot, idle : bool, m : nat, t, trem : real, t1, t2 : →real
    | idle := true; t := 0.0; trem := 0.0
    ; t1 := exponential( $\frac{1}{\mu_1}$ ); t2 := exponential( $\frac{1}{\mu_2}$ )
    ; * [ m = 1 ∧ idle; a?x → t :=  $\sigma t_1 + \tau$ ; idle := false
        [ m = 1 ∧ ¬idle;  $\Delta(t - \tau)$  → c!x; idle := true
          [ m = 2 ∧ idle; b?x → t :=  $\sigma t_2 + \tau$ ; idle := false
            [ m = 2 ∧ ¬idle;  $\Delta(t - \tau)$  → c!x; idle := true
              [ true ; u?m → idle := false
                ; [ m = 1 → ⟨trem, t⟩ := ⟨t -  $\tau$ , trem +  $\tau$  +  $\sigma_{21}$ ⟩
                  [ m = 2 → ⟨trem, t⟩ := ⟨t -  $\tau$ , trem +  $\tau$  +  $\sigma_{12}$ ⟩
                ]
              ]
            ]
          ]
        ]
    ]
  ]
]

```

(X-5)

The initial state of the machine (X-5) is assumed to be ready to process lots ('idle') of type 1 ('process1'). When the controller does not interfere, the machine asks the buffer for a lot, the process time is determined by taking an exponential distributed sample. If the lot is processed it is send to the exit process. The same principle is applied for processing type 2. When the controller gives to the machine its input signal u , processing is cut off, the remaining time is stored and the remaining process time is recalled of the lot cut off earlier. When a setup is performed, the setup time is added to the remaining process time of the lot that has cut off earlier. When the machine finished the lot, it asks for a new lot from the buffer. Note, expression $c!x$ is located behind the arrow. This channel sends products from the machine to the exit-proces. This is permitted only when the receiver is able to receive the lots immediately at all times.

The final process is the exit process (X-6).

```

proc E(a: ?lot) =
  [ [ x : lot
    | * [ true; a?x → skip ]
  ]
]

```

(X-6)

The exit process can received processed lots at all times.
The system-file and xper are presented in (X-7) and (X-8):

$$\begin{aligned}
 \text{sys}t \ S(\hat{\lambda}_1, \lambda_2, \mu_1, \mu_2, \sigma_{12}, \sigma_{21}, \phi_1, P: \text{real}, ini_1, ini_2, x_1^\# : \text{nat}) = \\
 \llbracket a, b, c, f: - \text{lot} \\
 , e: - \text{nat}^2 \\
 , u: - \text{nat} \\
 | G1(a, \hat{\lambda}_1, \phi_1, P) \ || \ G2(b, \lambda_2) | \\
 | B(a, b, c, e, ini_1, ini_2) | \\
 | M(c, u, f, \mu_1, \mu_2, \sigma_{12}, \sigma_{21}) | \\
 | C(e, u, \phi_1, P, x_1^\#) | \\
 | E(f) | \\
 \rrbracket
 \end{aligned} \tag{X-7}$$

$$\begin{aligned}
 \text{xper} \ (\hat{\lambda}_1, \lambda_2, \mu_1, \mu_2, \sigma_{12}, \sigma_{21}, \phi_1, P: \text{real}, ini_1, ini_2, x_1^\# : \text{nat}) = \\
 \llbracket S(\hat{\lambda}_1, \lambda_2, \mu_1, \mu_2, \sigma_{12}, \sigma_{21}, \phi_1, P, ini_1, ini_2, x_1^\#) \\
 \rrbracket
 \end{aligned} \tag{X-8}$$

Before the simulation can be performed, the parameter setting has to be uploaded. Subsequently, these parameters are used in the different processes as mentioned above. The set of parameters is

$$(\hat{\lambda}_1, \lambda_2, \mu_1, \mu_2, \sigma_{21}, \sigma_{12}, \phi_1, P, x_1(t=0), x_2(t=0), x_1^\#).$$

Remark 6.1.1. Variable $x_1^\#$ has to be computed. It is very unlikely the computed value results in a real number, although it does represent a integer in a discrete event setting. The system contains at least one slow mode (Lemma 4.1.4). The presence of a slow mode implies the machine has more capacity than needed. This makes round up of $x_1^\#$ a reasonable solution as long as enough capacity is available. However, when $x_1^\#$ becomes smaller, the larger the error becomes during round up and capacity can become more critical.

At this moment, all processes are explained except for the controllers. The rest of this section focusses on the three controller and their discrete event simulation results.

6.2 Simulation results of the controller for situation I

The controller, designed for situation I, is checked in this section. The results of the simulations have to be buffer levels which converge to the optimal trajectory and keep the system there. The desired trajectory found in the simulations has to be similar to the optimal steady state trajectory presented in Figure 4.5 of Chapter 4. Before the results

are discussed, first the χ -syntax of controller 1 is presented in (χ -9).

```

proc C1(a: ?nat2, u: !nat,  $\phi_1, P, \mu_1, \sigma_{21}$ : real) =
  || x : nat2, b : bool, m : nat, t : real
  | m := 1
  ; u!m
  ; *[ true  $\longrightarrow$  a?x; t := rmod( $\tau, P$ )
      ; [ m = 1  $\longrightarrow$  b := x.0 = 0  $\wedge$  t  $\geq$   $\phi_1 P$   $\wedge$  t < (P -  $\sigma_{21}$ )
          ; [ b  $\longrightarrow$  m := 2; u!m
              ||  $\neg$ b  $\longrightarrow$  skip
              ]
          ]
      || m = 2  $\longrightarrow$  b := (P - t)  $\leq$   $\frac{x.0}{\mu_1}$  +  $\sigma_{21}$ 
          ; [ b  $\longrightarrow$  m := 1; u!m
              ||  $\neg$ b  $\longrightarrow$  skip
              ]
          ]
      ]
  ]
  ]

```

(χ -9)

The controller (χ -9) is implemented in the discrete event model. The initialization of the controller is to send an input signal to the machine. So independent of the state of the system, the controller wants the machine to start with \bullet initially. Next the controller receives the buffer levels and starts to use the information to send specific tasks to the machine. The combination of the moment in time and the buffer levels lead to a situation where the controller gives the machine a new input signal or skips. The controller switches to type 2 if buffer 1 is empty and switched back to start processing type 1 when products of type 1 start to arrive.

In the simulation a parameter setting is used that meets all requirements needed in situation *I*. In the example the maximum arrival rate of type 1 is larger than the maximum process rate ($\hat{\lambda}_1 > \mu_1$). The parameter settings that have been used are presented in Table 6.1. The results of the hybrid fluid approximation model simulations

$\bar{\lambda}_1$:	0.6	lots/hr.	ϕ_1 :	0.3	-
λ_2 :	0.2	lots/hr.	$x_0(t=0)$:	0	hrs.
μ_1 :	1.5	lots/hr.	$x_1(t=0)$:	400	lots
μ_2 :	1	lots/hr.	$x_2(t=0)$:	400	lots
σ_{12} :	50	hrs.	$\Delta(t=0)$:	$\phi_1 P$	hrs.
σ_{21} :	50	hrs.	$m(t=0)$:	2	-
P :	1000	hrs.			

Table 6.1: Parameter setting for situation I

with the parameter setting of Table 6.1 are presented in Figures 6.2 and 6.4. The results of the discrete event model, with the same parameter setting, are presented in Figures 6.3 and 6.5. The figures on the left-hand side show the buffer levels in time.

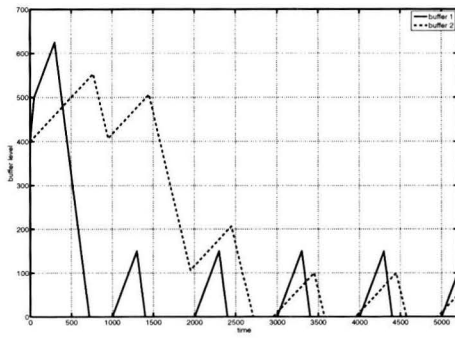


Figure 6.2: Buffer levels over time (Fluid)

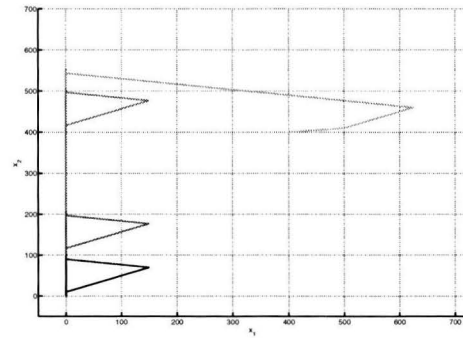


Figure 6.4: Trajectory (Fluid)

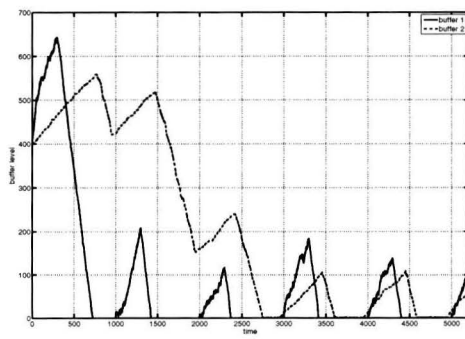


Figure 6.3: Buffer levels over time (Discrete)

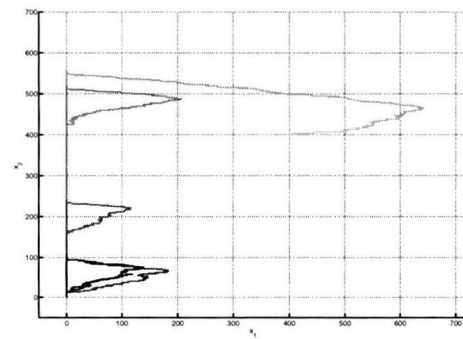


Figure 6.5: Trajectory (Discrete)

Simulation results of the controller for situation I

The figures on the right-hand side show the periodic orbit. Both simulations converge to the desired trajectory. The convergence of the system is explained with the use of the results of the hybrid fluid approximation model.

The system has an initial state of:

$$\vec{x}(0) = [400 \quad 400 \quad 0 \quad 300 \quad 2]^T.$$

The system is ready to process type 2 but with this state the controller makes the system switch to start a setup to type 1 (last condition of Proposition 5.1.1). At the same time, products of type 1 are arriving. During this setup products of both types arrive. When the setup is completed, ① starts. During the processing of type 1 products keep arriving. The buffer level of type 1 keeps increasing, but now at a lower rate. In Figure 6.4, the difference can be recognized as a different angle of increasing buffer levels. In point *A* variable Δ becomes 0 and products of type 1 stop arriving. Still $x_1 > 0$ so the machine stays processing type 1 until the buffer is empty. When the buffer is empty ② is performed. After the setup, type 2 is processed until point *B* is reached. In this point the controller knows that after an interval with a length of σ_{21} products of type 1 will start to arrive. The controller ends ②, starts ① and when products of type 1 start to arrive, the controller starts ① also. Here the arrival rate and processing type 1 are synchronized. After processing type 1 again, the machine has more time to process type 2 because less products of type 1 have to be processed to empty buffer 1 (Lemma 4.2.4). When type 2 is being processed both buffers become empty at a point in time. The system starts to process products of type 2 in a slow mode. The slow mode stops if the controller notices that products of type 1 are going to arrive soon. At this point the machine processes at its desired (optimal) trajectory. So the controller does steer the system to its desired trajectory. The discrete event simulation shows convergence too. Both simulations show great resemblance although it takes the discrete event simulation one loop more to reach the desired trajectory. In the next section a similar analysis is performed for parameter settings which satisfy the conditions of situation II-a.

6.3 Simulation results of the controller for situation II-a

The previous section showed the results of the controller designed for situation I. In this section the experimental results with the controller, designed for situation II-a, are discussed. The results of the simulations have to be similar to the trajectory presented in Figure 4.6 or in Figure 4.7. The parameter setting used in the simulation determines if the optimal trajectory has only a slow mode for type 2 or a slow mode for both types. The accompanying χ -syntax of the controller needed for the discrete event simulation

is presented in (χ -10).

```

proc C2a(a: ?nat2, u: !nat,  $\phi_1$ , P: real,  $x_1^\#$ : nat) =
  || x: nat2, b: bool, m: nat, t: real
  | m := 1
  ; u!m
  ; *[ true  $\rightarrow$  a?x; t := rmod( $\tau$ , P)
      ; [ m = 1  $\rightarrow$  b := x.0 = 0  $\wedge$  t  $\geq$   $\phi_1 P$ 
          ; [ b  $\rightarrow$  m := 2; u!m
              ||  $\neg$ b  $\rightarrow$  skip
              ]
          ]
      ; [ m = 2  $\rightarrow$  b := x.0  $\geq$   $x_1^\#$ 
          ; [ b  $\rightarrow$  m := 1; u!m
              ||  $\neg$ b  $\rightarrow$  skip
              ]
          ]
      ]
  ]
||

```

(χ -10)

The initialization of controller 2 is the same as controller 1. So independent of the state of the system, the controller wants the machine to start with **1** initially. When the controller receives the buffer levels after the initialization, it determines the position in time within one process cycle. The controller switches to type 2 if no products of type 1 arrive anymore and buffer 1 is empty. The controller switches back when the critical value $x_1^\#$ is reached. In all other situations the controller skips and waits for a new update of the buffer levels.

In the simulation an example is used where $\hat{\lambda}_1 < \mu_1$. So in situation II-a the maximum arrival rate of type 1 is less than the maximum process rate of type 1. The parameter setting that is used in the simulation is presented in Table 6.2.

$\bar{\lambda}_1$: 0.2	lots/hr.	ϕ_1 : 0.8	-
λ_2 : 0.45	lots/hr.	$x_0(t=0)$: 0	hrs.
μ_1 : 1	lots/hr.	$x_1(t=0)$: 400	lots
μ_2 : 1	lots/hr.	$x_2(t=0)$: 400	lots
σ_{12} : 50	hrs.	$\Delta(t=0)$: $\phi_1 P$	hrs.
σ_{21} : 50	hrs.	$m(t=0)$: 2	-
P: 1000	hrs.		

Table 6.2: Parameter setting for situation II-a

The results of both simulations with the parameter setting of Table 6.2 are presented in Figures 6.6, 6.8, 6.7 and 6.9. The system starts with an initial state for the system:

$$\vec{x}(0) = [400 \quad 400 \quad 0 \quad 800 \quad 2]^T.$$

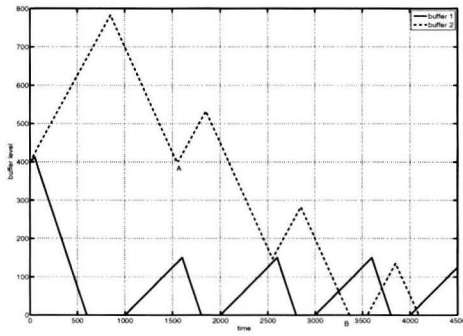


Figure 6.6: Buffer levels over time (Fluid)

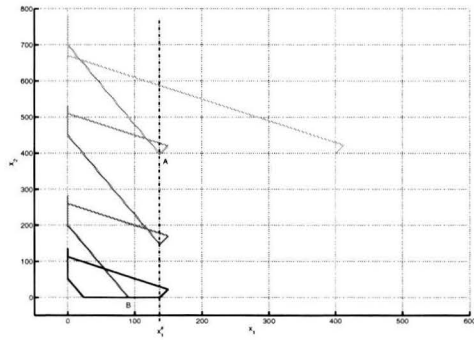


Figure 6.8: Trajectory (Fluid)

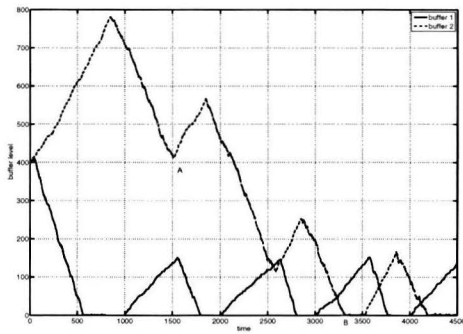


Figure 6.7: Buffer levels over time (Discrete)

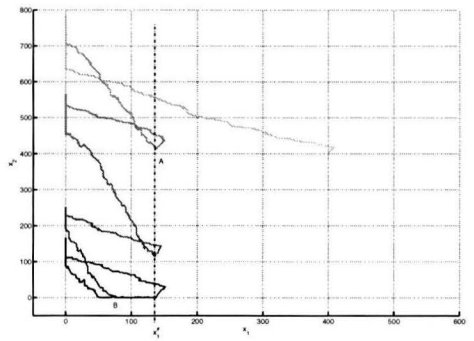


Figure 6.9: Trajectory (Discrete)

Simulation results of the controller for situation II-a

Since the buffer level of type 1 exceeds $x_1^\#$. The controller wants the system to start processing type 1. First ① is performed, during this setup both types arrive, so both buffer levels increase. After the setup to type 1 the machine processes type 1 until the buffer is empty. When the buffer is empty, the machine keeps processing in a slow mode until the arrivals of type 1 stop. At this moment the arrival pattern of type 1 is synchronized with the processing of type 1. But the buffer level of type 2 is still to large. When ② is performed, products of type 2 are processed. Type 2 is processed as long as $x_1 < x_1^\#$. In point *A* this equality becomes false. Setup ① is performed and the machine starts processing type 1. Again ① continues until the buffer is empty and the arrivals of type 1 stop. This behavior repeats itself until the buffer level of type 2 becomes zero (*B*). In point *B* type 2 is processed in a slow mode. As long as $x_1 < x_1^\#$ holds, the machine processes type 2 at the same rate as the arrival rate of type 2. When the machine starts to process type 2 in a slow mode for the first time, the system has reached its optimal steady state process cycle. The results of the discrete event simulation show similar behavior again, although the convergence to reach the desired trajectory takes longer. From the figures one can not see if there is a slow mode for type 1 present. When computing the optimum, with (4.25), $\tau_1^* = 160.52$ hrs. The lower and upper bound are 200 hrs. and 450 hrs. respectively. The result is an optimum of $\tau_1 = 200$ hrs. what implies that only product type 2 has a slow mode. The desired trajectory has a similar trajectory like scenario 1 (Figure 4.6) describes.

The results of the simulations for situation II-a correspond to the optimal trajectory as mentioned in Chapter 4. In the next section the last controller is checked by simulation. The simulation uses a parameter setting which satisfies the conditions of situation II-b.

6.4 Simulation results of the controller for situation II-b

The previous section showed the results of the controller designed for a desired trajectory with at least a slow mode for type 2. This section discusses a system which has at least a slow mode for type 1 in its desired trajectory. The desired trajectory that has to evolve must be similar to the scenarios 2 and 3 as mentioned in Chapter 4. The parameter setting used in this simulation determines with which scenario the results must be compared. Before the results are shown the accompanying χ -syntax of the controller

for situation II-b is presented in (χ -11).

$$\begin{aligned}
&\text{proc } C2b(a: ?\text{nat}^2, u: !\text{nat}, \phi_1, P: \text{real}, x_1^\# : \text{nat}) = \\
&\llbracket x : \text{nat}^2, b : \text{bool}, m : \text{nat}, t : \text{real} \\
&\quad | m := 1 \\
&\quad ; u!m \\
&\quad ; *[\text{true} \longrightarrow a?x; t := \text{rmod}(\tau, P) \\
&\quad \quad ; [m = 1 \longrightarrow b := t \geq \phi_1 P \\
&\quad \quad \quad ; [b \longrightarrow m := 2; u!m \\
&\quad \quad \quad \quad \parallel \neg b \longrightarrow \text{skip} \\
&\quad \quad \quad \quad] \\
&\quad \quad \parallel m = 2 \longrightarrow b := x.0 \geq x_1^\# \wedge x.1 = 0 \\
&\quad \quad \quad ; [b \longrightarrow m := 1; u!m \\
&\quad \quad \quad \quad \parallel \neg b \longrightarrow \text{skip} \\
&\quad \quad \quad \quad] \\
&\quad \quad] \\
&\quad] \\
&\rrbracket
\end{aligned} \tag{\chi-11}$$

The initialization of the controller (11) contains the assignment to start the setup to type 1. So independent of the state of the system, the controller wants to start with **1** initially. When the controller receives the buffer levels after the initialization, it determines the position in time within one process cycle. The controller switches when $\Delta = 0$, which is independent of the buffer level of type 1. Next, **2** is performed and buffer 2 is emptied. After **1** and products of type 1 arrive, the machine processes them until the arrivals stop. In all other situations the controller skips and waits for a new update of the buffer levels. A parameter setting for which the simulations have to be performed correctly is presented in Table 6.3. The results of both simulations with the

$\bar{\lambda}_1$: 0.5	lots/hr.	ϕ_1 : 0.9	-
λ_2 : 0.15	lots/hr.	$x_0(t=0)$: 0	hrs.
μ_1 : 1	lots/hr.	$x_1(t=0)$: 400	lots
μ_2 : 1	lots/hr.	$x_2(t=0)$: 400	lots
σ_{12} : 50	hrs.	$\Delta(t=0)$: $\phi_1 P$	hrs.
σ_{21} : 50	hrs.	$m(t=0)$: 1	-
P : 1000	hrs.		

Table 6.3: Parameter setting for situation II-b

parameter setting of Table 6.3 are presented in Figures 6.10, 6.12, 6.11 and 6.13. The system initially starts processing type 1 or type 2. In the parameter setting $m(t=0)$ is set to $m = 1$. So the machine starts processing type 1. When the arrivals stop, point A , variable Δ is synchronized with the processing of type 1, but the buffer level of type 1 is not. When $\Delta = 0$ the machine switches to **2** and then to processing type 2. When buffer 2 is empty, setup **1** is performed to start with processing type 1. At point B

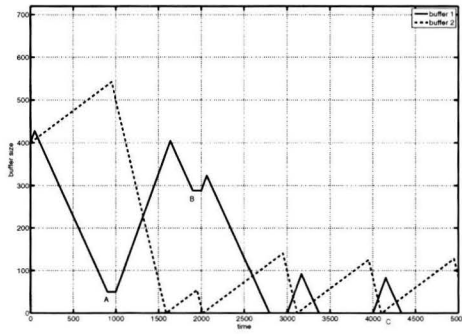


Figure 6.10: Buffer levels over time (Fluid)

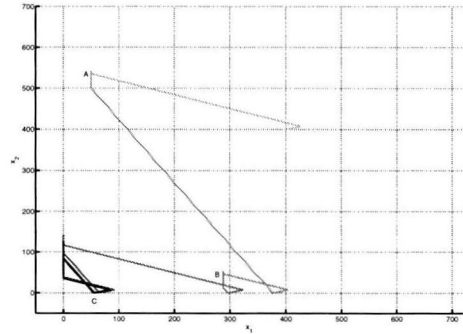


Figure 6.12: Trajectory (Fluid)

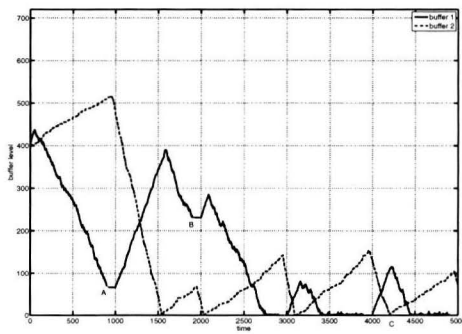


Figure 6.11: Buffer levels over time (Discrete)

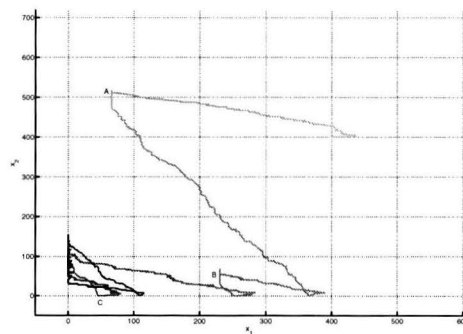


Figure 6.13: Trajectory (Discrete)

Simulation results of the controller for situation II-b

products of type 1 stop to arrive again. This is also the moment to stop processing type 1. A setup to type 2 is performed and the system empties buffer 2 again. Parameter $x_1 > x_1^\#$, this means type 1 needs to be processed. This behavior repeats itself until the buffer level of type 1 becomes zero. Now the system performs a loop close to the optimal trajectory before it reaches the situation where $x_2 = 0$ and $x_1 < x_1^\#$. After a short slow mode for type 2, the system reaches point C . At this point the controller has steered the system to its steady state trajectory. Also the discrete event simulation reaches the same steady state trajectory. Both steady state trajectories are the same. The trajectories have to correspond to either scenario 2 or scenario 3. The results show no slow mode for type 2. Computation of the optimal process length of type 1 results in a value for τ_1 which is equal to the upper bound of the system. So the parameter setting represents an example of the optimal trajectory presented as scenario 3. The controller steers to a correct trajectory.

All three controllers have been tested. All simulations present controllers which steer towards the desired trajectories. The path of convergence is in each simulation the same for the hybrid fluid model as for the discrete event model. The controllers show to be robust enough to deal with the disturbances introduced by exponential distributions on the inter-arrival times and process times.

In different proofs the presence of the slow mode has been discussed. When initial buffer levels are high, time spans for slow modes are replaced by processing at maximum capacity. The larger these time spans are (=the longer slow modes are active in the desired trajectory), the larger the difference is between the number of products that are processed and the number of products that need to be processed. The speed of convergence depends on the length of the time spans of the slow modes. It is only when a slow mode is present the system can keep the synchronization between processing type 1 and the arrival pattern of type 1 and catch up with buffer levels which are too large.

Chapter 7

Conclusions and recommendations

In this chapter the conclusions of this research are presented in Section 7.1 and the recommendations for future research are presented in Section 7.2.

7.1 Conclusions

This report discussed an optimal process cycle and feedback control for a workstation serving two product types with setups, one piecewise-constant arrival rate and a constant arrival rate. The approach is similar to the approach in [Eek06a]. The system is optimized with respect to the weighted time averaged work in process (wip) level. The theory presented in this report contains a hybrid fluid approximation model which gives proper results if the system satisfies two important conditions. The sum of time fractions that each product needs to be processed in the system has to be less than one. The second condition implies that the length of the periodic behavior equals the length of a steady state process cycle. In the steady state process cycle at least one slow mode has to occur. If these conditions hold, different optimal process cycles are obtained. The differences between the optimal process cycles depend on a relationship between the maximum arrival rate and the maximum process rate of type 1 (the type which has a piecewise constant arrival pattern). After determination of the optimal process cycles, one situation described the system if the maximum arrival rate is larger than the process rate ($\hat{\lambda} \geq \mu$). The second situation described the system if the maximum arrival rate is less than the maximum process rate ($\hat{\lambda} \leq \mu$). In the second situation a subdivision is made, based on the slow mode(s) that occur in the optimal process cycle. Together these situations cover all parameter settings which meet the two conditions mentioned earlier. Eventually three controllers have been proposed which together cover all possible optimal process cycles. For these controllers a proof of convergence has been established. The analytical proof shows that the controllers always steer a system with

arbitrary buffer levels, and starting at an arbitrary point in time, to the desired trajectory and keeps it there. These controllers have been tested in a simulation study. In this study the workstation is simulated with a hybrid fluid approximation model and a discrete event model. The simulation results confirmed the convergence to the desired trajectories. In spite of determination of the feedback controller in a continuous setting, the controllers in the discrete event setting, with stochastic distributions for the inter-arrival times and process times, converge in the same way. For the simulation results as presented, the controllers are robust enough to deal with the disturbances introduced by the exponential distributions. The speed of convergence depends on the sum of time fractions that each product needs to be processed in the system. The higher the sum the slower the system converges and vice versa.

7.2 Recommendations

Several items need more attention to obtain a better understanding of the optimal behavior of a system with one piecewise constant arrival rate in this research. These research objectives are presented below:

- The approach of this report is similar to the approach in [Eek06a]. The one step missing, compared with [Eek06a], is to analyze the system with finite buffer capacities. The derived feedback controllers work well for systems with infinite buffers. But it is not clear if the workstation can synchronize its processing of type 1 with the arrival pattern of type 1, due to the buffer restrictions. Especially when buffer capacities are relatively small with respect to the maximum buffer levels that occur during the optimal steady state process cycle of the system. It can become very hard to steer the system to the desired trajectory because the synchronization of the arrival pattern of type 1 with the processing of type 1 often takes time, which may not be available because the other buffer exceeds its maximum capacity.
- If the constraint that one process cycle must equal one period is dropped, it is possible to find steady state process cycles with a lower time averaged weighted wip level. Good candidates are systems which have a sum of time fractions that each product needs to be processed that is small. An example of such a system is presented in the intermezzo at the end of Chapter 4.
- Imagine two workstations in series. The possibility exists that the first workstation processes products in a slow mode. For the departure rate this means products of one type leave the workstation at a rate of μ , λ or 0. This means the piecewise constant arrival pattern of the second workstation contains three possible arrival rates also. After the maximum arrival rate, the products can arrive at a lower speed before the products stop to arrive and the rate becomes zero. In this report

the piecewise constant arrival rate has only two different values ($\hat{\lambda}$ and 0). In general the new arrival pattern will lead to different process cycles and different time averaged weighted wip levels. When these ‘new’ arrival patterns will be analyzed with the possible situations that occur in this report (a system with one constant arrival rate and the ‘modified’ piecewise constant arrival pattern), it might have a positive effect on situation I. Situation I may approve because the system can not keep up with the highest arrival rate, if the rate is tempered after a while the machine might be able to catch up earlier. How situation II will react on such a ‘modified’ piecewise constant arrival pattern is hard to determine. Even a comparison with situation II may not be realistic because the lemmas used in Chapter 4 become useless.

Another topic that forms an interesting research objective is a workstation with two piecewise constant arrival patterns instead of one. This report contains a workstation with one piecewise constant arrival pattern and one constant arrival pattern. The next step is to perform an analysis for the same workstation with two piecewise constant arrival patterns. Workstations with such arrival patterns occur when two workstations are put in series (as mentioned in Example 3.2.1). The output of the first machine is for both product types piecewise constant. This output pattern forms the arrival pattern of the second machine. To optimize the weighted time averaged work in process in the second machine, first the optimal process cycle for each type of product must be determined separately. When the process cycles are combined an overlap in processing the optimal cycles of type 1 and 2 may occur. Removing the overlap from the optimal process cycle results in extra costs. The optimization that has to be performed needs to minimize the extra costs that occurs due to the overlaps. To obtain such a minimum, a function for the possible overlaps has to be determined. This function must determine which type of product has to be processed first in order to keep the weighted time averaged work in process as low as possible.

Bibliography

- [Aln04] K.A. Alnowibet and H. Perros. Nonstationary analysis of the loss queue and of queueing networks of loss queues. <http://www.csc.ncsu.edu/faculty/perros/recentpapers.html>, 2004.
- [Aln06] K.A. Alnowibet and H. Perros. Nonstationary analysis of circuit-switched communication networks. *Performance Evaluation*, 63(9-10):892–909, October 2006.
- [Aza06] A. Azaron, H. Katagiri, K. Kato, and M. Sakawa. Modeling complex assemblies as a queueing network for lead time control. *European journal of operational research*, 174(1):150–168, October 2006.
- [Bee00] D.A. van Beek and J.E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of χ . *Control Engineering Practice*, 8(1):81–91, January 2000.
- [Bek04] R. Bekker, S.C. Borst, O.J. Boxma, and O. Kella. Queues with workload-dependent arrival and service rates. *Queueing Systems*, 46(3-4):537–556, March-April 2004.
- [Cha92] C. Chase and P.J. Ramadge. On real-time scheduling policies for flexible manufacturing systems. *IEEE Transactions on automatic control*, 37(4):491–496, April 1992.
- [Eek06a] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Feedback control of 2-product server with setups and bounded buffers. *Proceedings of the 2006 American Control Conference*, pages 544–549, 2006.
- [Eek06b] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. State feedback control of switching servers with setups. SE Report 2006-03, Eindhoven University of Technology, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2006. <http://se.wtb.tue.nl/sereports>.
- [Gre91] L.V. Green and P.J. Kolesar. The pointwise stationary approximation for queues with nonstationary arrivals. *Management Sciences*, 37:84–97, 1991.

- [Gre97] L.V. Green and P.J. Kolesar. The lagged PSA for estimating peak congestion in multiserver markovian queues with periodic arrival rates. *Management Sciences*, 43:80–87, 1997.
- [Hof02] A.T. Hofkamp and J.E. Rooda. χ reference manual. internal report. http://w3.wtb.tue.nl/nl/organisatie/systems_engineering/documentation/, November 2002.
- [Ign76] J.P. Ignizio. *Goal Programming and Extensions*. Heath, Boston, MA, 1976.
- [Jag75] D.L. Jagerman. Nonstationary blocking in telephone traffic. *The Bell System Technical Journal*, 54:626–661, 1975.
- [Lan06] W.L. Lan and T.L. Olsen. Multi-product systems with both setup times and costs: Fluid bounds and schedules. *Operations Research*, 54(3):505–522, May 2006.
- [Lef06] E. Lefeber and J.E. Rooda. Controller design for switched linear systems with setups. *Physica A: statistical mechanics and its applications*, 363(1):48–61, April 2006.
- [Mas02] W.A. Massey. The analysis of queues with time-varying rates for telecommunication models. *Telecommunication Systems*, 21(2-4):173–204, December 2002.
- [Mat00] A.S. Matveev and A.V. Savkin. *Qualitative Theory of Hybrid Dynamical Systems*. Birkhäuser, Boston, MA, 2000.
- [Mon99] D.C. Montgomery and G.C. Runger. *Applied statistics and probability for engineers*. John Wiley & Sons, Inc., 2nd edition, 1999.
- [Per89] J.R. Perkins and P.R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/diassembly systems. *IEEE transactions on automatic control*, 34(2):139–148, February 1989.
- [Rid03] A.D. Ridley, M.C. Fu, and W.A. Massey. Fluid approximations for a priority call center with time-varying arrivals. In *Proceedings of the 2003 Winter Simulation Conference*, pages 1817–1823. Informs Simulation Society, December 2003.
- [Som06] J. Somlo and A.V. Savkin. Periodic and transient switched server schedules for FMS. *Robotics and Computer-Integrated Manufacturing*, 22(2):93–122, April 2006.

Appendix A

Fluid models

This appendix contains all three fluid models discussed in Chapter 6. The appendix presents situation I, situation II-a and situation II-b respectively.

In the appendix the proposed feedback controllers have been modeled as proposed in propositions 5.1.1, 5.2.1 and 5.3.1. Initially the system has an arbitrary state. The values of this state determine in which case the machine starts. In these simulations the system starts with buffer levels of 400 products for each type, a setup to start processing type 1 and $\Delta = P$. Like the remarks that accompanied each proposition, the models in this appendix contain the different modes also. These modes are indicated as cases. All conditions in each mode are translated into conditions for each case in the model. The result is the following Matlab-code. A detailed explanation of the models is presented after the models.

Controller for situation I

```
% controller situation~I

clc; clear all; close all;

lambda1=0.6;    % mean arrival rate of type 1
lambda2=0.2;    % arrival rate of type 2
mu1=1.5;        % mu_1
mu2=1;          % mu_2
s12=50;         % setup time sigma_12
s21=50;         % setup time sigma_21
c1=1;           % weighting factor for type 1
c2=1;           % weighting factor for type 2
phi=0.3;        % time fraction where products of type~1 arrive.
P=1000;         % length of a period

r1=lambda1/mu1;    % mean rho_1
```

```

r2=lambda2/mu2;          % rho_2
lambda1hat= lambda1/phi;% lambda 1 hat

B1max= phi*P*(lambda1hat-mu1);

B1=400; B2=400; m=6;
B=[0,B1,B2,s21];        %[time, buffer level 1, buffer level 2, remaining setup time]

while length(B) < 40;    % Length of simulation
  switch m
  case 1 %process type 1
    [d,st] = check(P,phi,B(end,1));
    if st == 1
      B=[B;B(end,1)+d B(end,2)+d*(lambda1hat-mu1) B(end,3)+d*lambda2 0];
      m=1;
    else st == 0
      if d < B(end,2)/mu1
        B=[B;B(end,1)+d B(end,2)-d*mu1 B(end,3)+d*lambda2 0];
        m=1;
      else
        B=[B;B(end,1)+B(end,2)/mu1 0 B(end,3)+(B(end,2)/mu1)*lambda2 0];
        m=2;
      end
    end
  case 2 % slow mode type 1
    [d,st] = check(P,phi,B(end,1));
    B=[B;B(end,1) B(end,2) B(end,3) s12];
    m=3;
  case 3 % setup from 1 to 2
    [d,st] = check(P,phi,B(end,1));
    if st == 1
      if d <= B(end,4);
        B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(end,4)-d];
        m=3;
      else
        B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
        m=4;
      end
    else
      if d <= B(end,4)
        B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
        m=3;
      else
        B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
        m=4;
      end
    end
  case 4 % process type 2
    [d,st] = check(P,phi,B(end,1));
    if st == 1
      B=[B;B(end,1) B(end,2) B(end,3) 0];
      m=5;
    else st == 0
      delta=B(end,2)/mu1+s21;
      if d > delta & d-delta > B(end,3)/(mu2-lambda2)
        B=[B;B(end,1)+B(end,3)/(mu2-lambda2) B(end,2) 0 0];
        m=5;
      elseif d > delta & d-delta <= B(end,3)/(mu2-lambda2)
        B=[B;B(end,1)+(d-delta) B(end,2) B(end,3)-(d-delta)*(mu2-lambda2) 0];
        m=5;
      elseif d > delta & d-delta > B(end,3)/(mu2-lambda2)
        B=[B;B(end,1)+delta B(end,2) 0 0];

```



```

        m=5;
    else
        B=[B;B(end,1) B(end,2) B(end,3) 0];
        m=5;
    end
end
case 5 %slow mode type 2
[d,st] = check(P,phi,B(end,1));
if st == 1
    B=[B;B(end,1) B(end,2) B(end,3) s21];
    m=6;
else
    if d > B(end,2)/mu1+s21 & B(end,2) < B1max
        tslow=d-((B(end,2)/mu1)+s21);
        B=[B;B(end,1)+tslow B(end,2) B(end,3) s21];
        m=6;
    else
        B=[B;B(end,1) B(end,2) B(end,3) s21];
        m=6;
    end
end
case 6 %setup from 2 to 1
[d,st] = check(P,phi,B(end,1));
if st == 1
    if d < B(end,4);
        B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else d >= B(end,4);
        B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
else
    if d < B(end,4);
        B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else d >= B(end,4);
        B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
end
end
end
end

```

Controller for situation II-a

```

% controller situation~II-a

clc; clear all; close all;

lambda1=0.2; % mean arrival rate of type 1
lambda2=0.45; % arrival rate of type 2
mu1=1; % mu_1
mu2=1; % mu_2
s12=50; % setup time sigma_12
s21=50; % setup time sigma_21
c1=1; % weighting factor for type 1
c2=1; % weighting factor for type 2
phi=0.8; % time fraction where products of type~1 arrive.
P=1000; % length of a period

```

```

r1=lambda1/mu1;          % mean rho_1
r2=lambda2/mu2;          % rho_2
lambda1hat= lambda1/phi;% lambda 1 hat
%opt tau
tauistar=((c1*mu1*r1/(phi-r1))*(phi*P)-(c2*mu2*r2/(1-r2))*(s12+s21))/...
...(((c1*mu1*r1)/(phi-r1))+((c2*mu2*r2)/(1-r2)));
g1=r1*P;                 % lower bound
g2=(1-r2)*P-(s12+s21);   % upper bound
g3=phi*P;                 % time span products arrive

if tauistar <= g1;
    tauiopt=g1;
elseif g2 >= g3 & tauistar > g3;
    tauiopt=g3;
elseif g2 < g3 & tauistar > g2;
    tauiopt=g2;
else
    tauiopt=tauistar;
end
B2max= (tauiopt+s21)*lambda2;
B1max= (((phi*P-tauiopt)*r1)/(phi-r1))*(mu1-lambda1hat)-(lambda1hat*s21);

B1=400; B2=400; m=6;
B=[0,B1,B2,s21];          %[time, buffer level 1, buffer level 2, remaining setup time]

while length(B) < 40;     % Length of simulation
    switch m
        case 1 %process type 1
            [d,st] = check(P,phi,B(end,1));
            if st == 1
                if d < B(size(B,1),2)/(mu1-lambda1hat)
                    B=[B;B(end,1)+d B(end,2)-d*(mu1-lambda1hat) B(end,3)+d*lambda2 0];
                    m=3;
                else
                    B=[B;B(end,1)+B(size(B,1),2)/(mu1-lambda1hat) 0...
                    ... B(end,3)+B(end,2)/(mu1-lambda1hat)*lambda2 0];
                    m=2;
                end
            end
        else
            B=[B;B(end,1) B(end,2) B(end,3) 0];
            m=3;
        end
        case 2 % slow mode type 1
            [d,st] = check(P,phi,B(end,1));
            if st == 1
                B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 s12];
                m=3;
            else
                B=[B;B(end,1) B(end,2) B(end,3) s12];
                m=3;
            end
        case 3 % setup from 1 to 2
            [d,st] = check(P,phi,B(end,1));
            if st == 1
                if d <= B(size(B,1),4)
                    B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(v,4)-d];
                    m=3;
                else
                    B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
                    m=4;
                end
            end
    end
end

```

```

else
  if d <= B(end,4)
    B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
    m=3;
  else
    B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
    m=4;
  end
end
case 4 % process type 2
[d,st] = check(P,phi,B(end,1));
if st == 1
  if d <= B(end,3)/(mu2-lambda2) & B(end,2)+d*lambda1hat <= B1max
    B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)-d*(mu2-lambda2) 0];
    m=4;
  elseif d > B(end,3)/(mu2-lambda2) & B(end,2)+d*lambda1hat <= B1max
    B=[B;B(end,1)+B(end,3)/(mu2-lambda2) B(end,2)+B(end,3)/(mu2-lambda2)*lambda1hat 0 0];
    m=5;
  elseif d <= B(end,3)/(mu2-lambda2) & B(end,2)+d*lambda1hat > B1max
    tarrival=max((B1max-B(end,2))/lambda1hat,0);
    B=[B;B(end,1)+tarrival B(end,2)+tarrival*lambda1hat B(end,3)-tarrival*(mu2-lambda2) 0];
    m=5;
  else
    tarrival=max((B1max-B(end,2))/lambda1hat,0);
    tempty=B(end,3)/(mu2-lambda2);
    if tarrival <= tempty
      B=[B;B(end,1)+tarrival B(end,2)+tarrival*lambda1hat...
        ... B(end,3)-tarrival*(mu2-lambda2) 0];
    else
      B=[B;B(end,1)+tempty B(end,2)+tempty*lambda1hat 0 0];
    end
    m=5;
  end
end
else
  if d <= B(end,3)/(mu2-lambda2)
    B=[B;B(end,1)+d B(end,2) B(end,3)-d*(mu2-lambda2) 0];
    m=4;
  else
    B=[B;B(end,1)+B(end,3)/(mu2-lambda2) B(end,2) 0 0];
    m=5;
  end
end
case 5 %slow mode type 2
[d,st] = check(P,phi,B(end,1));
if st == 1
  if B(end,2) >= B1max
    B=[B;B(end,1) B(end,2) B(end,3) s12];
    m=6;
  elseif B(end,2)+d*lambda1hat >= B1max
    tslow=(B1max-B(end,2))/lambda1hat;
    B=[B;B(end,1)+tslow B(end,2)+tslow*lambda1hat B(end,3) s12];
    m=6;
  else
    B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3) 1];
    m=5;
  end
end
else
  if B(end,2) >= B1max
    B=[B;B(end,1) B(end,2) B(end,3) s12];
    m=6;
  else
    B=[B;B(end,1)+d B(end,2) B(end,3) s12];

```

```

        m=5;
    end
end
case 6 %setup from 2 to 1
[d,st] = check(P,phi,B(end,1));
if st == 1
    if d <= B(end,4)
        B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else
        B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
end
else
    if d <= B(end,4)
        B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else
        B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
end
end
end
end
end

```

Controller for situation II-b

```

%%% controller for situation II-b

clc; clear all; close all;

%%%Parameter Setting

lambda1=0.5;           % mean arrival rate of type 1
lambda2=0.15;          % arrival rate of type 2
mu1=1;                 % mu_1
mu2=1;                 % mu_2
s12=50;                % setup time sigma_12
s21=50;                % setup time sigma_21
c1=1;                  % weighting factor for type 1
c2=1;                  % weighting factor for type 2
phi=0.9;               % time fraction where products of type 1 arrive.
P=1000;                % length of a period

r1=lambda1/mu1;        % mean rho_1
r2=lambda2/mu2;        % rho_2
lambda1hat= lambda1/phi;% lambda 1 hat

%%%Optimization of tau1

taulstar=((c1*mu1*r1/(phi-r1))*(phi*P)-(c2*mu2*r2/(1-r2))*(s12+s21))/...
...(((c1*mu1*r1)/(phi-r1))+((c2*mu2*r2)/(1-r2)));
g1=r1*P;               % lower bound
g2=(1-r2)*P-(s12+s21); % upper bound
g3=phi*P;              % time span products arrive

if taulstar <= g1;
    tauiopt=g1;
elseif g2 >= g3 & taulstar > g3;

```

```

    tau1opt=g3;
elseif g2 < g3 & tau1star > g2;
    tau1opt=g2;
else
    tau1opt=tau1star;
end
B2max= (tau1opt+s21)*lambda2;
B1max= (((phi*P-tau1opt)*r1)/(phi-r1))*(mu1-lambda1hat)-(lambda1hat*s21);

B1=400; B2=400; m=6;      % Initial buffer levels and start mode.
B=[0,B1,B2,s12];        % [time, buffer level 1, buffer level 2, remaining setup time]

while length(B) < 40;    % Length of simulation
    switch m
    case 1 %process type 1
        [d,st] = check(P,phi,B(end,1));
        if st == 1
            if d < B(end,2)/(mu1-lambda1hat);
                B=[B;B(end,1)+d B(end,2)-d*(mu1-lambda1hat) B(end,3)+d*lambda2 0];
                m=1;
            else
                B=[B;B(end,1)+B(end,2)/(mu1-lambda1hat) 0...
                    ... B(end,3)+B(end,2)/(mu1-lambda1hat)*lambda2 0];
                m=2;
            end
        end
    case 2 % slow mode type 1
        [d,st] = check(P,phi,B(end,1));
        if st == 1
            B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 s12];
        else
            B=[B;B(end,1) B(end,2) B(end,3) s12];
        end
        m=3;
    case 3 % setup from 1 to 2
        [d,st] = check(P,phi,B(end,1));
        if st == 1
            if d <= B(end,4)
                B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(end,4)-d];
                m=3;
            else
                B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
                m=4;
            end
        else
            if d <= B(end,4)
                B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
                m=3;
            else
                B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
                m=4;
            end
        end
    case 4 % process type 2
        [d,st] = check(P,phi,B(end,1));
        if st == 1
            if d < B(end,3)/(mu2-lambda2)
                B=[B;B(end,1)+d B(end,2)+d*(lambda1hat) B(end,3)-d*(mu2-lambda2) 0];
                m=4;
            else

```

```

        B=[B;B(end,1)+B(end,3)/(mu2-lambda2) B(end,2)+(B(end,3)/(mu2-lambda2))*lambda1hat 0 0];
        m=5;
    end
else
    if d < B(end,3)/(mu2-lambda2);
        B=[B;B(end,1)+d B(end,2) B(end,3)-d*(mu2-lambda2) 0];
        m=4;
    else
        B=[B;B(end,1)+B(end,3)/(mu2-lambda2) B(end,2) 0 0];
        m=5;
    end
end
case 5 %slow mode type 2
[d,st] = check(P,phi,B(end,1));
if st == 1
    if B(end,2)+d*lambda1hat >= B1max
        if B(end,2)>= B1max
            tslow=0;
        else
            tslow=(B1max-B(end,2))/lambda1hat;
        end
        B=[B;B(end,1)+tslow B(end,2)+tslow*lambda1hat B(end,3) s21];
        m=6;
    else
        B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3) 0];
        m=5;
    end
end
else
    if B(end,2) >= B1max
        B=[B;B(end,1) B(end,2) B(end,3) s21];
        m=6;
    else
        B=[B;B(end,1)+d B(end,2) B(end,3) 0];
        m=5;
    end
end
case 6; %setup from 2 to 1
[d,st] = check(P,phi,B(end,1));
if st == 1
    if d <= B(end,4)
        B=[B;B(end,1)+d B(end,2)+d*lambda1hat B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else
        B=[B;B(end,1)+B(end,4) B(end,2)+B(end,4)*lambda1hat B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
end
else
    if d <= B(end,4)
        B=[B;B(end,1)+d B(end,2) B(end,3)+d*lambda2 B(end,4)-d];
        m=6;
    else
        B=[B;B(end,1)+B(end,4) B(end,2) B(end,3)+B(end,4)*lambda2 0];
        m=1;
    end
end
end
end
end

```

The function-file *check.m* is defined as:

```
function [d,st] = check(P,phi,t)
```

```

while t >= P;
    t=t-P;
end;

if t >= phi*P;
    d=P-t;
    st=0;
else
    d=phi*P-t;
    st=1;
end

```

In these Matlab models several parts can be distinguished. First the parameter setting is introduced. Next, (only for situations II) the optimization of τ_1 is performed which results in the computation of maximum buffer levels for type 1 and 2 (respectively $B1_{max}$ and $B2_{max}$). These maximum buffer levels present the maximum level at which the system should perform a setup to start processing the other type of product. The initial conditions are described and the length of the computation is chosen. In these models the initial buffer levels are 400 lots each, and the machine start in mode 6 (Setup to type 1). In the Matlab-code each mode number is presented as a ‘Case’ number. When a case becomes active, the current time is sent to function *check.m*. This file receives the current time, the time fraction where products of type 1 arrive during one period P and the length of a period P . With these variables it determines if lots of type 1 arrive and the time span the current arrival rate maintains. If no lots arrive *check.m* returns $st = 0$ and the time span where this arrival rate stays zero (d). When lots do arrive *check.m* sends $st = 1$ and the time span the products keep arrive at this rate.

When the variables st and d are known, one of the arguments within the active case is met. Performing the argument that is valid, the buffer levels are updated, stored and the case number is updated. When the buffer levels are updated, matrix B stores the current time, buffer level 1, buffer level 2 and the remaining setup time. With the new value for m the whole process of checking the new situation has to be performed again. The simulation stops when the size of matrix B reaches 40. The number of 40 is determined empirically and represents about 4 process cycles.

Appendix B

Discrete event models

This appendix contains the ASCII-code of all three complete discrete event models discussed in Chapter 6. The appendix presents situation I, situation II-a and situation II-b respectively. The iconic representation of the χ -model is presented in Figure B.1. The model contains two generators (G_1 and G_2). Generator 1 sends products with a

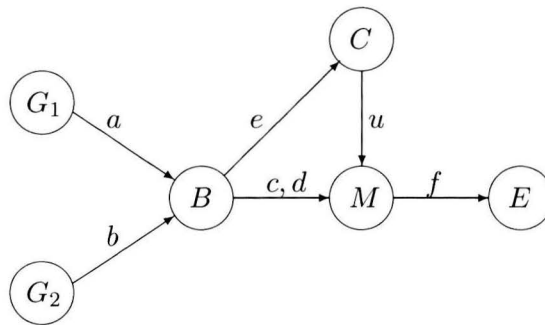


Figure B.1: Iconic model of a two product workstation

piecewise constant arrival pattern with an exponential distribution. Generator 2 sends products with a constant arrival pattern with an exponential distribution. Both generators send the lots to the buffer (B). The buffer counts the number of products in each buffer and sends the information to the controller (C). The controller uses the information to determine if the machine (M) has to perform setups or process products. If the controller allows production, products are available and a product moves from the buffer to the machine for processing. The process times of the machine are exponential distributed also. After processing a product it is send to the exit process (E) where it leaves the system.

Controller of situation I

```

from std import*
from fileio import*
from random import*

type lot = nat,

// Generator of type 1.
proc G1(a: !lot, b: nat, c,d,e: real) =
|[r,s:real, t:->real
 |r:=0.0; t:=exponential(1/c)
 ;*[true -> r:=rmod(time, e); s:=sample t;
      [r+s < d*e -> delta s; a!b
      |r+s >= d*e -> delta e-r
      ]
 ]
]|

// Generator of type 2.
proc G2(a: !lot, b: nat, c: real) =
|[t:->real
 |t:=exponential(1/c)
 ;*[true -> a!b; delta sample t]
]|

// Buffer
proc B(a,b: ?lot, c,d: !lot, e: !nat^2, buf1ini,buf2ini: nat) =
|[x1,x2:lot, xs1,xs2:nat
 |xs1:=buf1ini; xs2:=buf2ini; e!<|xs1,xs2|>
 ;*[true -> [true; a?x1 -> xs1:=xs1 + 1
            |true; b?x2 -> xs2:=xs2 + 1
            |xs1>0; c!x1 -> xs1:=xs1 - 1
            |xs2>0; d!x2 -> xs2:=xs2 - 1
            ]
 ; e!<|xs1,xs2|> ;fileout("buffersize.txt")!time, "\t", xs1, "\t", xs2, "\n"
 ]
]|

// Machine
proc M(a,b: ?lot, u: ?nat, c: !lot, p1,p2,s12,s21: real) =
|[x:lot, m:nat, t,trem:real, t1,t2:->real, idle:bool
 |t:=0.0; trem:=0.0; t1:=exponential(1/p1); t2:=exponential(1/p2); idle:=true
 ;*[m=1 and idle; a?x -> t:=sample t1+time; idle:=false
 |m=1 and not idle; delta t-time -> c!x; idle:=true
 |m=2 and idle; b?x -> t:=sample t2+time; idle:=false
 |m=2 and not idle; delta t-time -> c!x; idle:=true
 |true; u?m -> idle:=false;
 [m=1 -> <trem,t>:=<t-time,trem+time+s21>
 |m=2 -> <trem,t>:=<t-time,trem+time+s12>
 ]
 ]
]|

// Controller
proc C1 (a: ?nat^2, u: !nat, phi,p,mu1,s21: real) =
|[x: nat^2, b:bool, m: nat, t: real
 |m:=1; u!m
 ;*[true -> a?x; t:=rmod(time,p);
 [m=1 -> b:= x.0 = 0 and t >= phi*p and t < p-s21;
 [b -> m:=2; u!m
 |not b -> skip
 ]
 ]
]|

```

```

]
|m=2 -> b:= p-t <= x.0/mu1+s21 ;
|b -> m:=1; u!m
|not b -> skip
]
]
]

]
]

// Exit process
proc E(a: ?lot) =
|[x:lot
|*[true; a?x ->skip]
]|

clus S() =
|[ a,b,c,d,f:-lot, e:-nat^2, u:-nat
| G1(a,1,lambda1hat,phi1,P) || G2(b,2,lambda2)
|| B(a,b,c,d,e,buf1ini,buf2ini)
|| M(c,d,u,f,mu1,mu2,sigma12,sigma21)
|| C1(e,u,phi1,P,mu1,sigma21)
|| E(f)
]|

xper(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P: real, buf1ini,buf2ini:nat)=
|[ S(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P,buf1ini,buf2ini) ]|

```

Controller of situation II-a

```

from std import*
from fileio import*
from random import*

type lot = nat,

// Generator of type 1.
proc G1(a: !lot, b: nat, c,d,e: real) =
|[r,s:real, t:->real
|r:=0.0; t:=exponential(1/c)
;*[true -> r:=rmod(time, e); s:=sample t;
|r+s < d*e -> delta s; a!b
|r+s >= d*e -> delta e-r
]
]

]

// Generator of type 2.
proc G2(a: !lot, b: nat, c: real) =
|[t:->real
|t:=exponential(1/c)
;*[true -> a!b; delta sample t]
]|

// Buffer
proc B(a,b: ?lot, c,d: !lot, e: !nat^2, buf1ini,buf2ini: nat) =
|[x1,x2:lot, xs1,xs2:nat
|xs1:=buf1ini; xs2:=buf2ini; e!<|xs1,xs2|>
;*[true -> [true; a?x1 -> xs1:=xs1 + 1
|true; b?x2 -> xs2:=xs2 + 1
|xs1>0; c!x1 -> xs1:=xs1 - 1

```

```

        |xs2>0; d!x2    -> xs2:=xs2 - 1
        ]
        ; e!<|xs1,xs2|> ;fileout("buffersize.txt")!time, "\t", xs1, "\t", xs2, "\n"
    ]
]
]

// Machine
proc M(a,b: ?nat, u: ?nat, c: !lot, p1,p2,s12,s21: real) =
|[x:lot, m:nat, t,trem:real, t1,t2:->real, idle:bool
 |t:=0.0; trem:=0.0; t1:=exponential(1/p1); t2:=exponential(1/p2); idle:=true
 ;*[m=1 and idle; a?x          -> t:=sample t1+time; idle:=false
   |m=1 and not idle; delta t-time -> c!x; idle:=true
   |m=2 and idle; b?x          -> t:=sample t2+time; idle:=false
   |m=2 and not idle; delta t-time -> c!x; idle:=true
   |true; u?m                  -> idle:=false;
                               [m=1 -> <trem,t>:=<t-time,trem+time+s21>
                               |m=2 -> <trem,t>:=<t-time,trem+time+s12>
                               ]
]
]
]

// Controller
proc C2a (a: ?nat^2, u: !nat, phi,p: real, x1m: nat) =
|[x: nat^2, b: bool, m: nat, t: real
 |m:=1; u!m
 ;*[true -> t:=rmod(time,p); a?x;
   [m=1 -> b:= x.0 = 0 and t >= phi*p;
     [b      -> m:=2; u!m
       |not b -> skip
     ]
   |m=2 -> b:= x.0 >= x1m ;
     [b      -> m:=1; u!m
       |not b -> skip
     ]
   ]
]
]
]

// Exit process
proc E(a: ?lot) =
|[x:lot
 |*[true; a?x ->skip]
]
]

clus S() =
|[ a,b,c,d,f:-lot, e:-nat^2, u:-nat
 | G1(a,1,lambda1hat,phi1,P) || G2(b,2,lambda2)
 || B(a,b,c,d,e,buf1ini,buf2ini)
 || M(c,d,u,f,mu1,mu2,sigma12,sigma21)
 || C2a(e,u,phi1,P,x1max)
 || E(f)
]
]

xper(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P: real, buf1ini,buf2ini,x1max:nat)=
|[ S(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P,buf1ini,buf2ini,x1max) ]|

```

Controller of situation II-b

```

from std import*
from fileio import*

```

```

from random import*

type lot = nat,

// Generator of type 1.
proc G1(a: !lot, b: nat, c,d,e: real) =
|[r,s:real, t:->real
 |r:=0.0; t:=exponential(1/c)
 ;*[true -> r:=rmod(time, e); s:=sample t;
      [r+s < d*e -> delta s; a!b
      |r+s >= d*e -> delta e-r
      ]
 ]
]|

// Generator of type 2.
proc G2(a: !lot, b: nat, c: real) =
|[t:->real
 |t:=exponential(1/c)
 ;*[true -> a!b; delta sample t]
]|

// Buffer
proc B(a,b: ?lot, c,d: !lot, e: !nat^2, buf1ini,buf2ini: nat) =
|[x1,x2:lot, xs1,xs2:nat
 |xs1:=buf1ini; xs2:=buf2ini; e!<|xs1,xs2|>
 ;*[true -> [true; a?x1 -> xs1:=xs1 + 1
            |true; b?x2 -> xs2:=xs2 + 1
            |xs1>0; c!x1 -> xs1:=xs1 - 1
            |xs2>0; d!x2 -> xs2:=xs2 - 1
            ]
 ; e!<|xs1,xs2|> ;fileout("buffersize.txt")!time, "\t", xs1, "\t", xs2, "\n"
 ]
]|

// Machine
proc M(a,b: ?lot, u: ?nat, c: !lot, p1,p2,s12,s21: real) =
|[x:lot, m:nat, t,trem:real, t1,t2:->real, idle:bool
 |t:=0.0; trem:=0.0; t1:=exponential(1/p1); t2:=exponential(1/p2); idle:=true
 ;*[m=1 and idle; a?x -> t:=sample t1+time; idle:=false
   |m=1 and not idle; delta t-time -> c!x; idle:=true
   |m=2 and idle; b?x -> t:=sample t2+time; idle:=false
   |m=2 and not idle; delta t-time -> c!x; idle:=true
   |true; u?m -> idle:=false;
   [m=1 -> <trem,t>:=<t-time,trem+time+s21>
   |m=2 -> <trem,t>:=<t-time,trem+time+s12>
   ]
 ]
]|

// Controller
proc C2b (a: ?nat^2, u: !nat, phi,p: real, x1m: nat) =
|[x:nat^2, b: bool, m: nat, t: real
 |m:=1; u!m
 ;*[true -> t:=rmod(time,p); a?x;
      [m=1 -> b:= t >= phi*p;
        [b -> m:=2; u!m
        |not b -> skip
        ]
      |m=2 -> b:= x.1=0 and x.0 >= x1m;
        [b -> m:=1; u!m
        |not b -> skip
        ]
      ]
]|

```

```

        ]
    ]
]

// Exit process
proc E(a: ?lot) =
| [x:lot
  |[true; a?x ->skip]
]|

clus S() =
| [ a,b,c,d,f:-lot, e:-nat^2, u:-nat
  | G1(a,1,lambda1hat,phi1,P) || G2(b,2,lambda2)
  || B(a,b,c,d,e,buf1ini,buf2ini)
  || M(c,d,u,f,mu1,mu2,sigma12,sigma21)
  || C2b(e,u,phi1,P,x1max)
  || E(f)
]|

xper(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P: real, buf1ini,buf2ini,x1max:nat)=
|[ S(lambda1hat,lambda2,mu1,mu2,sigma12,sigma21,phi1,P,buf1ini,buf2ini,x1max) ]|

```

In the three discrete event models the buffer levels are stored after each product that leaves or arrives at the buffer. Besides the buffer levels the time at which the buffer levels change is stored also. The data is stored in the text-file *'buffersize.txt'*. The data is used to visualize the behavior of the system.