Control of a reentrant manufacturing system with setup times

Mathijs J.M. Dohmen

SE 420526

Master Thesis

Supervisor: prof.dr.ir. J.E. Rooda
Coach: dr.ir. A.A.J. Lefeber

Eindhoven University of Technology
Department of Mechanical Engineering
Systems Engineering Group

Eindhoven, January 2008

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

January 2008

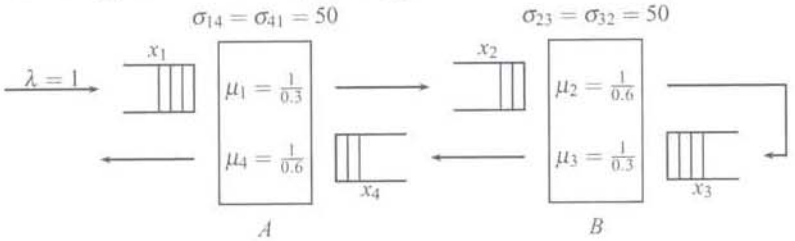
Student	M.J.M. Dohmen
Supervisor	Prof.dr.ir. J.E. Rooda
Advisor	Dr.ir. A.A.J. Lefeber
Start	January 2007
Finish	January 2008
Title	Control of a reentrant manufacturing system with setup times.

Subject

Control of reentrant manufacturing systems with setup times is difficult, since using controllers that are stable for a machine in isolation might render the system unstable. So far, in literature, most people first propose a policy, and then study the resulting behavior of the network under this policy.
Recently, an entirely different way of looking at the problem has been proposed. Instead of starting from a policy and then analyzing the proposed policy, one can also start from a priori specified desired network behavior. Using this desired behavior for the network under consideration as a starting point, a policy can be derived which guarantees convergence of the system towards this desired behavior.

Assignment

The above mentioned approach has recently been applied successfully to the Kumar-Seidman case.



However, in that study costs for wip were assumed to be constant over the system, whereas downstream wip is more expensive due to added value.
Consider the Kumar-Seidman case with increasing costs for downstream wip. First determine good system behavior, preferable optimal. Next, derive a non-distributed feedback controller which makes the system converge towards the desired system behavior. Finally, see if a distributed implementation of the feedback controller can be achieved. Also, analyze the performance of the derived controllers by means of discrete event simulation, both in a deterministic and stochastic setting.
Present all results in a report and provide recommendations for further research.

Prof.dr.ir. J.E. Rooda

Dr.ir. A.A.J. Lefeber

Preface

Since I have a normal voice and not a typical pronunciation common for deaf people only a few know that I have an invisible handicap. At the age of 1.5 years I suffered meningitis and was balancing on the edge of life. I managed to survive but turned to be completely deaf. Soon thereafter it became clear that on one ear only minimal hearing rests (-80dB) remained in a narrow band around 4 kHz, the bandwidth of speech. With a dedicated hearing aid and the emphasis of many people I managed to find my way in the "hearing world". I can only communicate directly by reading lips supported by the "noise" of my hearing aid. I am unable to follow or participate any colleges, speeches, discussions, meetings, telephones and mobiles. Also radio and TV without subtitles are inappropriate for me.

Especially for me a study at the university would be a great challenge which I accepted. In September 2002, I enrolled the Mechanical Engineering program at TU/e. After completing the bachelor phase I decided to specialize myself in the Systems Engineering (SE) group. This group, headed by professor Rooda, aims at developing methods, techniques and tools for the design and control of advanced manufacturing systems. Please be advised that it took me a lot of energy to follow the colleges and meetings. But now I am proud to achieve a master degree the way most of you did without any privileges or extenuating circumstances.

I take the opportunity to end with some personal notes. I worked together with several people, whom I would like to thank for their support, assistance and contributions. First of all, a special thanks to my coach dr.ir. A.A.J. Lefeber for his professional coaching and support. Thanks to prof.dr.ir. J.E. Rooda, supervisor and responsible for creating a pleasant environment in which students feel at home and get the best out of themselves. Thanks to prof.dr.ir. J. van der Wal, who took the time to review this report. Also thanks to all my friends from the university in general and more specific to my fellow students and PhD's from the SE-lab.

And last, but certainly not least, I want to thank my parents, my brother Guy and sister Amy for their motivation, support and being there when needed!

Mathijs Dohmen
Eindhoven, January 2008

Summary

In this report, the control of a reentrant manufacturing system with setup times is studied. In a manufacturing system using reentrant line processing, the incomplete job is conveyed along a line of workstations and it may visit the same workstation several times during its travels along the line. Reentrant line processing is commonly used in the semiconductor industry, where each incomplete job may undergo the same processing steps several times at each workstation before the job is complete. This type of systems can also be found in thin film lines and systems with rework tasks.

The network topology of systems with reentrant lines makes the manufacturing planning and scheduling problems difficult. Several jobs at different stages of processing may be in contention with one another for service at the same workstation. Each workstation is processing multiple job types and switching between these job types take time (setup time). During this setup, the workstation is cleaned and re-adjusted which means that no jobs can be processed at this workstation, so capacity is lost. The question now arises: When should a workstation switch between the job types in an efficient way without instability of the system? A system is unstable when the total number of jobs in the system explodes as time evolves. Whether this happens depends on the policy used to control the flows through the network.

There is a large body of literature on production control of systems with reentrant lines. A lot of research has been done on determining the best scheduling policies to control the flows through the network. The goal of scheduling is to choose such policies to provide good performance with respect to performance measures of interest. Such performance measures may be the cost of the total work in process, the mean manufacturing lead time or cycle time, or even the variance of the manufacturing lead time. Most of these papers have one thing in common: first a policy (or a class of policies) is proposed, and then the resulting closed-loop behavior of the system under this policy (these policies) is considered. Sometimes the system behavior is optimized over the class of considered policies. A strength of these results is that the approach can be applied to general networks. A drawback however is that it is usually unclear if the presented policies result in optimal system behavior, or what to do to obtain prescribed or desired system behavior.

Due to this drawback, an entirely different way of looking at the problem has been proposed recently. Instead of starting from a policy and then analyzing the resulting closed-loop system behavior, one can also start from a priori specified desired closed-loop behavior and then design a policy which guarantees convergence of the system towards this desired behavior. This feedback control strategy has been applied successfully to the Kumar-Seidman case in [LR06a]. However, in that study costs for wip were assumed to be constant over the system,

whereas downstream wip is more expensive due to added value. In this report, this feedback control strategy is investigated further and the way to derive such a feedback policy has been illustrated extensively for the Kumar-Seidman case but now with increasing costs for downstream wip.

Before a feedback policy can be derived, the desired closed-loop system behavior is determined first. For manufacturing systems this is not a problem, the network typically is fixed and given a priori. In this report, the reentrant manufacturing system with setup times as introduced by Kumar and Seidman — a reentrant manufacturing system where the single job-type *revisits* the two workstations with one machine in a fixed predefined four step production process — is considered. For this specific reentrant system, a closed-loop behavior is determined with respect to minimal weighted work in progress level which minimizes the cycle period and the weighted amount of jobs in the system. The analysis is performed using a hybrid fluid model approximation and the behavior is proven analytically.

After this desired closed-loop system behavior is determined, a state feedback controller is designed — based on Lyapunov's direct method — which guarantees convergence of the system towards this desired behavior from any initial condition. Based on the desired closed-loop system behavior, an "energy" of the system can be defined by considering the mean amount of work in the system. By controlling the system in a way that this "energy" is never increasing, the system stabilizes at a fixed energy level and the controller will make the system converge from the initial condition towards the desired behavior. The way to derive this feedback controller from the closed-loop system behavior is illustrated extensively in this report. Also, it has been proven mathematically, that this designed feedback controller guarantees convergence of the system towards the desired behavior from any initial condition.

The designed controller is a non-distributed controller, each workstation in the system needs to have global state information for determining when to switch. For manufacturing systems, this so-called "global policy" is feasible, maybe also for some urban traffic networks. However, for other networks, e.g. communication networks or computer networks, this global information might not be available. For these networks, the designed non-distributed controller is implemented successfully in a distributed way, i.e. such that each workstation only requires local state information for determining when to switch.

An extensive case study demonstrates how both derived feedback controllers perform. Even though both derived feedback policies have been designed for a deterministic system, the derived feedback policies can also be applied in case inter-arrival times, setup times and processing times are stochastic. The resulting responses of the controlled system show stable closed-loop dynamics as well as convergence towards the desired behavior. In comparison with gated policies and clearing policies, the derived feedback policy guarantees convergence of the system from any initial condition towards desired behavior with a small weighted amount of jobs in the system.

Though the way to derive a feedback policy has only been illustrated extensively for a case with two workstations, this feedback control strategy works also for systems consisting of a single workstation [LR06a]. However, for systems consisting of three or more workstations, this feedback control strategy is not applied yet. In further research, this feedback control strategy can be applied to systems with several workstations. Furthermore, it is maybe possible to define a general method for these systems to make this strategy common applicable.

Samenvatting

Dit afstudeerverslag beschrijft de studie naar een optimale regeling van een fabricage systeem met herintreedbare werkstations met omsteltijden. Bij een fabricage lijn met herintreedbare werkstations wordt het onvoltooide produkt langs een lijn van werkstations gevoerd en kan het produkt hetzelfde werkstation meermalen aandoen voordat een volledig fabricage proces bereikt is. Dit principe wordt veelvuldig toegepast in de halfgeleiderindustrie, waar elk onvoltooide wafer specifieke processtappen meermalen kan doorlopen voordat het gereed is. Dergelijke fabricage systemen vindt men ook in de dunne film technologie en aanverwante systemen met herprocestaken.

De netwerktopologie van fabricage systemen met herintreedbare werkstations maakt de produktieplanning en -regeling complex. Verscheidene produkten in verschillende stadia van het fabricage proces kunnen in conflict met elkaar komen voor de verwerking bij hetzelfde werkstation. Elk werkstation verwerkt produkten in verschillende stadia van bewerking en het switchen tussen deze produkten vergt tijd (omsteltijd). Tijdens deze omstelling wordt het werkstation schoongemaakt en aangepast waardoor er geen produkten door dit werkstation geproduceerd kunnen worden wat resulteert in capaciteitsverlies. De vraag is nu: Wanneer zou een werkstation efficiënt moeten omschakelen zonder instabiel te worden. Een systeem is instabiel wanneer het aantal produkten binnen het systeem blijft toenemen. Of dit wel of niet gebeurd hangt af van de taktiek die wordt gebruikt om de goederenstromen in het netwerk te regelen.

Er is veel literatuur over de regeling van fabricage lijnen met herintreedbare werkstations. Veel onderzoek is gedaan naar het bepalen van de beste planningsmethode om de goederenstromen in het netwerk te regelen. Het doel van zo'n planning is het bereiken van een goede balans tussen haalbare eisen en prestaties. Maatstaven kunnen de totale produktiekosten zijn, de gemiddelde cyclustijd of zelfs de variatie in de gemiddelde produktietijd. De meeste artikelen hebben één ding gemeen. Eerst wordt een taktiek (of een combinatie van taktieken) gekozen en vervolgens wordt het resulterende gedrag van het gesloten systeem met deze taktiek geanalyseerd. Soms wordt het systeemgedrag geoptimaliseerd aan de hand van de reeds overwogen taktieken. Een voordeel van deze methode is dat het over het algemeen kan worden toegepast. Een nadeel is echter dat het onduidelijk is of de gekozen taktiek in optimaal systeemgedrag resulteert, of welke taktiek gekozen moet worden om een voorgeschreven of gewenste systeemgedrag te verkrijgen.

Vanwege dit nadeel is onlangs voorgesteld dit probleem anders te bekijken. In plaats van te beginnen met het kiezen van een taktiek om vervolgens het resulterende gesloten systeemgedrag te analyseren, kan men ook uitgaan van de bepaling van een specifiek gewenst

gesloten systeem gedrag en vervolgens de regelaar ontwerpen die ervoor zorgt dat het systeem convergeert naar dit gewenste gedrag. Deze strategie van terugkoppelingscontrole is met succes toegepast op de Kumar-Seidman case in [LR06a]. Echter, in deze studie werden de kosten voor onderhanden werk (WIP) constant verondersteld over het gehele systeem, terwijl in werkelijkheid voortschrijdende wip duurder is vanwege meer toegevoegde waarde. In dit rapport wordt deze strategie van de terugkoppelingscontrole verder onderzocht en de manier om een dergelijk terugkoppelingstaktiek af te leiden, maar nu met toenemende kosten voor voortschrijdende wip, aan de hand van de Kumar-Seidman case.

Voordat de terugkoppellingstaktiek wordt bepaald, wordt eerst het gewenste gesloten system gedrag bepaald. Voor fabricage systemen is dit geen probleem, een vastgelegd netwerk met gegeven prioriteiten is typerend. In dit rapport wordt een systeem overwogen met herintreedbare werkstations met omsteltijd, zoals geintroduceerd door Kumar en Seidman. Meer specifiek, een fabricage systeem met herintreedbare processtappen waarbij één type produkt wordt gemaakt op twee werkstations met elk één machine met een vooraf vastgelegd vierstappenproces. Voor dit systeem is een gesloten systeemgedrag bepaald waarin de gewogen wip is geminimaliseerd wat resulteert in een kleine periodieke procestijd met een laag aantal produkten in het systeem.

Nadat dit gewenste gesloten systeemgedrag is bepaald, wordt een terugkoppelingsregelaar ontworpen die gebaseerd is op de directe methode van Lyapunov en die vanuit elke begintoestand convergentie van het systeem naar het gewenste gedrag verzekerd. Uitgaande van het gewenste gesloten systeemgedrag kan een inspanning van het systeem bepaald worden aan de hand van de gemiddelde wip. Door nu het systeem zodanig te regelen dat deze inspanning nooit toeneemt zal het systeem stabiliseren op een vaste systeemtoestand en zal de regelaar het systeem convergeren van het uitgangspunt naar het gewenste systeemgedrag. De methode om vanuit de begintoestand van het gesloten systeemgedrag deze terugkoppelingsregelaar af te leiden wordt uitgebreid in dit verslag beschreven. Ook wordt wiskundig bewezen dat de ontworpen terugkoppelingsregelaar convergentie naar het gewenste gedrag vanuit elke begintoestand verzekerd.

De ontworpen regelaar is een niet gedistribueerde regelaar. Elk werkstation in het systeem heeft globale toestandsinformatie nodig om het schakelmoment te bepalen. Voor fabricage systemen is deze zogenaamde globale systeemtoestand beschikbaar. Misschien geldt dit ook voor verkeersnetwerken. Echter voor andere netwerken zoals communicatie- en of computernetwerken zal deze globale informatie misschien niet beschikbaar zijn. Voor deze netwerken is de ontworpen niet gedistribueerde regelaar succesvol toegepast op een gedistribueerde manier waarbij elk werkstation alleen lokale toestandsinformatie nodig heeft om het schakelmoment te bepalen.

Een uitgebreide case studie toont het gedrag aan van beide ontworpen terugkoppellingsregelaars. Ondanks dat beide verkregen terugkoppellingstaktieken ontworpen zijn voor een deterministisch systeem, kan de verkregen terugkoppellingstaktiek ook toegepast worden als de aankomstintervallen, omsteltijden en procestijden stochastisch zijn. De resulterende reactie van het geregelde systeem laat een stabiel gesloten dynamisch gedrag zien met steeds convergentie naar het gewenste systeemgedrag. In vergelijking met drempelniveau- en leegmaaktaktieken, verzekerd de verkregen terugkoppellingstaktiek convergentie van het systeem naar het gewenste gedrag vanuit elke begintoestand met gemiddeld een laag aantal produkten in het systeem.

Hoewel de manier om een terugkoppelingstaktiek af te leiden slechts uitgebreid is beschreven voor een case met twee werkstations, werkt deze strategie van de terugkoppelingscontrole ook voor systemen die uit één enkel werkstation bestaat [LR06a]. Echter, voor systemen die uit drie of meer werkstations bestaan, is deze strategie van de terugkoppelingscontrole niet nog toegepast. In verder onderzoek, kan deze strategie van de terugkoppelingscontrole op systemen met meer dan twee werkstations worden toegepast. Daarnaast is het misschien mogelijk om een algemene methode voor deze systemen te bepalen om deze strategie algemeen toepasbaar te maken.

Contents

Table of symbols

Symbol	Description
c_i	Weighting factor for buffer contents of buffer i
$b_{p,i}$	Job-type p stored in buffer i that works with machine m
B_m	Set of buffers that works with machine m
δ	Average number of jobs leaving the system per unit of time
d_p	Desired amount of processed jobs of job-type p
$\wp$	Feasible domain
$\mathcal{F}$	Feasible area
J	Weighted wip level function
k_m	Number of buffers that works with machine m
λ	Arrival rate of jobs into the system
$\mathbf{m}$, m_i	System mode, mode of workstation i
μ_i	Maximal process rate for jobs of buffer i
N	Total number of buffers in the system
σ_{ij}	Setup time to switch from step i to step j
τ_0^i	Idle duration of workstation i
τ_j^i	Process duration of step j at workstation i
T	Cycle period
t_0, t_f	Start, end time of a system mode
u_0^i	Activity of workstation i
$u_{p,i}$	Process rate for job-type p of buffer i at workstation m
$u_{IN_i}(t)$	Input rate of jobs into buffer i
$u_{OUT_i}(t)$	Output rate of jobs that leave buffer i
φ	Average time spent by a job in the system
V	Lyapunov function candidate
$v_{p,i}$	Arrival rate of job-type p into the regulated buffer from the regulator buffer
w	Average number of jobs present in the system
x	System state
x_0^i	Remaining setup time of workstation i
x_i	Buffer contents of buffer i
$\dot{x}_i(t)$	Nett rate of jobs in buffer i
x_i^0	Buffer contents of buffer i at the beginning of a system mode
x_i^f	Buffer contents of buffer i at the end of a system mode

Chapter 1

Introduction

Most people are familiar with systems of mass-production which use flow shop processing. In a production system using flow shop processing, a conveyor system carries the incomplete job along a line of workstations. At each workstation, a different step in the fabrication of the job is performed. For example, in an automobile assembly line using flow shop processing, the chassis of the automobile may be conveyed along to a first workstation where the engine is installed, a second workstation where the axles are installed, a third workstation where the wheels are installed and so on. In theory, from beginning to end, the incomplete job visits each workstation only once during its travel down the assembly line.

Most people are probably not familiar with systems of mass-production using a second type of processing known as reentrant line processing. In a production system using reentrant line processing, the incomplete job is conveyed along a line of workstations, but the incomplete job may visit the same workstation several times during its travels along the line. Reentrant line processing is commonly used in the semiconductor industry, where each incomplete job may undergo the same processing steps several times at each workstation before the job is completed. But these systems can also be found in thin film lines and systems with rework tasks.

Systems with reentrant line processing have great advantages in economical and technological points of view [KK01]. In reentrant lines, jobs may return more than once to the same machine. It is therefore possible to process the jobs with fewer machines, which leads to a reduction in capital investment. In the semiconductor industry for example, each machine costs millions of euro. Also some technological advantages such as maintaining alignment can be obtained by processing each job more than once on the same machine. For example in the semiconductor industry, the same stepper may be used to expose wafers at different layers for improving overlay accuracy.

Besides these advantages, systems with reentrant lines have certain characteristics that make them very difficult to schedule. The most important is the *complexity of process flow* [KK01], several jobs at different stages of processing may be in contention with one another for processing at the same machine. An other characteristic is the *diversity of equipment* [ULMV92]. In a system with several workstations, the arrival rates, setup times and process rates are often different for each workstation. Finally, *stochastic variability*, uncertainties in the form of random arrivals, services or setup times, as well as random machine failures and repairs

makes these systems difficult to schedule. In the context of manufacturing systems, the term 'scheduling' refers to the control of the process flow.

There is a large body of literature on production control of systems with reentrant lines. A lot of research has been done on determining the best scheduling or dispatching policies to control the flow through the network. The goal of scheduling is to choose such policies to provide good performance with respect to performance measures of interest. Such performance measures may be the cost of the total work in process, the mean manufacturing lead time or cycle time, or even the variance of the manufacturing lead time. Several researchers have studied the design and performance analysis of scheduling policies for several problems in different manufacturing reentrant systems. In general, there are two important decisions that have significant effect on the performance of a reentrant manufacturing system [NK96].

- Input release policies — that decide when to release new jobs into the system
- Dispatching or scheduling policies — that decide which job has to be processed next when a machine becomes available

Input release policies are for example the closed-loop release policy and the workload regulation policy [Kum93]. If the total number of jobs, or work in progress (WIP), in the system should be constant, the closed-loop release policy or Fixed WIP policy only releases a new job into the system if a finished job leaves the system [NK96]. Another possibility is to try to release new jobs into the system whenever the number of jobs, or work, destined for a certain machine in the system, drops below a certain threshold. This policy is known as a workload regulation policy. This is thoroughly investigated in [BG96] for single job types as well as for multiple job types.

The scheduling or dispatching policies in reentrant lines become interesting because several jobs at different stages of processing may be in contention with one another for service at the same machine. The First Come First Serve (FCFS) scheduling policy — where a machine provides service to the job which arrived first to that machine — is well known and used by many researches [LK91]. Another class of scheduling policies discussed in the literature [Kumar [Kum93], Uzsoy *et al.*[ULMV92], Kumar an Meyn [KM95], Lu and Kumar [LK91], Dai *et al.*[DYZ97]] is the class of buffer scheduling policies. When a machine finishes processing a job, a buffer policy selects the next job for processing from among the buffers in a fixed priority order. Two prominent buffer priority-based policies are the Last Buffer First Serve (LBFS) — where the jobs in the last buffer have the highest priority to processed first on a machine and the jobs in the first buffer have the lowest priority — and the First Buffer First Serve (FBFS) — where the jobs in the first buffer have the highest priority to processed first on a machine and the jobs in the last buffer have the lowest priority — policies.

Reentrant manufacturing systems with setup times — time needed for switching between the buffers with stored jobs at the same machine — are investigated by Kumar and Seidman [KS90] and Perkins *et al.* [PJK94]. They have developed stable distributed scheduling policies and analyzed the stability and performance of these type of systems.

This report will not provide much coverage of the extensive general historical developments in dispatching policies and input release policies. A lot of these policies for reentrant systems with references to other thorough surveys can be seen in [Bis97]. However, most of the papers

have one thing in common: first a policy (or a class of policies) is proposed, and then the resulting behavior of the network under this policy (these policies) is considered. Sometimes the system behavior is optimized over the class of considered policies. A strength of these results is that they can be applied to general networks. A drawback however is that it is usually unclear if the presented policies result in optimal behavior, or what to do to obtain prescribed or desired system behavior. In particular, if a network is known a priori (and not subject to change), which typically is the case for manufacturing systems, one has the possibility of taking a global viewpoint and design a controller for the network which imposes desired network performance [LR06a].

1.1 Project objective

The approach followed in this report is exactly the other way around as in the above mentioned papers. First the desired system behavior is determined. Next, this desired closed-loop behavior of the system is used as a starting point and then, based on the ideas presented in [ELR06], a global and local policy is presented which establishes convergence to this desired behavior. This approach has been investigated on the Kumar-Seidman case in [LR06a]. In this report, this approach is investigated further. First, the reentrant manufacturing system with setup times in the Kumar-Seidman case is chosen again. But some system parameters are changed — increasing costs for downstream wip instead of constant costs for wip over the system — which results in another interesting system behavior which should be controlled by a global and local policy.

The objective for this project is formulated as:

Project objective:
Control of a reentrant manufacturing system with setup times and increasing costs: the Kumar-Seidman case.

The approach — starting from a certain policy which works for a general network and analyze the resulting network behavior — is not investigated. The specific manufacturing system in the Kumar-Seidman case is investigated for the approach starting from the desired system behavior and then determining a feedback controller which makes the system converge towards this desired behavior. Therefore, the first part of the research can be formulated as follows:

Research objective 1:
Determine the desired periodic system behavior.

When the desired periodic system behavior is determined, a feedback controller is developed which makes the manufacturing system converge towards this desired steady state behavior from any initial condition. This resulting controller is a non-distributed controller: each workstation needs only global state information. For a manufacturing system this is not a problem, since global information is available, the network typically is fixed and given a priori. The second part of this research can be formulated as follows:

Research objective 2:
Determine a non-distributed controller (global policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

Also a distributed controller is developed. With this feedback controller, each workstation needs local state information. In the specific manufacturing system for example, which contains two workstations (workstation A and B), workstation A does not require information about the state at workstation B to determine its next task and workstation B does not require information about the state at workstation A. The third part of this research can be formulated as follows:

Research objective 3:
Determine a distributed controller (local policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

The desired periodic system behavior and both feedback controllers are determined for systems with deterministic system parameters. But this approach can also be used for systems with stochastic system parameters. The developed feedback controller is used for systems with stochastic system parameters by performing simulations. The fourth part of this research can be formulated as follows:

Research objective 4:
Check if the developed feedback controllers work for systems with stochastic system parameters by performing simulations.

1.2 Report outline

This report is structured as follows. The specific reentrant manufacturing system in the Kumar-Seidman case is explained in Chapter 2. Also the state, input, constraints and dynamics of this system are presented. In Chapter 3, the desired periodic behavior of the system is determined step by step and proved. After that, the desired periodic orbit of the system is determined which can be used as a starting point for the controller design. In Chapter 4, a non-distributed controller where each workstation only needs global state information is designed. Simulations are performed to show the convergence of the system from the initial system state towards the desired system behavior, both for deterministic as well as for stochastic system parameters. Also, the improvements of this derived feedback policy are shown by comparing the resulting responses of the controlled system with other commonly used policies. In Chapter 5, the non-distributed controller is implemented in a distributed way. Also, for this distributed controller, simulations are performed to show the convergence of the system and the improvements. Finally, the conclusions and recommendations for further research are described in Chapter 6.

Chapter 2

Reentrant manufacturing system with setup times

To investigate the approach — starting from the desired periodic system behavior and then determining a feedback controller which makes the system converge towards this desired periodic system behavior — a specific manufacturing system is needed. As mentioned in the introduction, the objective for this project was formulated as follows:

Project objective:
Control of a reentrant manufacturing system with setup times and increasing costs: the Kumar-Seidman case.

First, the specific manufacturing system in the Kumar-Seidman case is explained in detail in this chapter. When this specific system is known, all the data (i.e. arrival rate, number of workstations, number of machines, process rates, setup times) are available to determine a desired periodic system behavior. When this desired periodic system behavior is determined a feedback controller can be derived which makes the specific system converge towards this desired periodic system behavior.

2.1 The Kumar-Seidman case

The manufacturing system of interest to illustrate this approach is the case presented by Kumar and Seidman [KS90], as mentioned in Chapter 1.

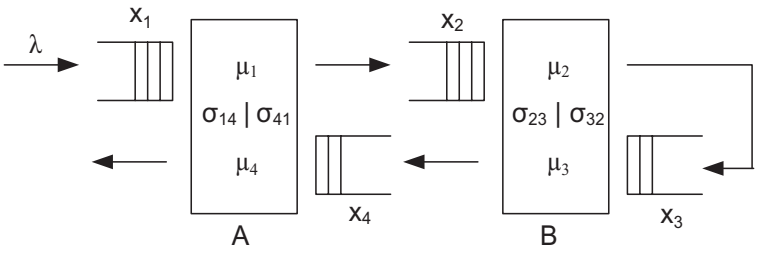


Figure 2.1: Reentrant manufacturing system with setup times.

Table 2.1: System parameter values.

Arrival rate		Workstation A		Workstation B	
λ $[\frac{\text{job}}{\text{time-unit}}]$	1	μ_1 $[\frac{\text{jobs}}{\text{time-unit}}]$	$\frac{1}{0.3}$	μ_2 $[\frac{\text{jobs}}{\text{time-unit}}]$	$\frac{1}{0.6}$
		μ_4 $[\frac{\text{jobs}}{\text{time-unit}}]$	$\frac{1}{0.6}$	μ_3 $[\frac{\text{jobs}}{\text{time-unit}}]$	$\frac{1}{0.3}$
		σ_{14} [time-unit]	50	σ_{23} [time-unit]	50
		σ_{41} [time-unit]	50	σ_{32} [time-unit]	50

The system in Figure 2.1 is a reentrant manufacturing system. The single job-type *revisits* the system with two workstations (workstation A and B) in a fixed predefined 4 step production process. Each job-type is processed first at workstation A for the first step, then at workstation B for the second step, then again at workstation B for the third step and finally again at workstation A for the fourth step, before leaving the system. In this case, each workstation contains one machine. The successive buffers visited will be denoted by 1, 2, 3 and 4 respectively, with its buffer contents of stored jobs x_1, x_2, x_3 and x_4. The single job-type arrives the system with an arrival rate λ $[\frac{\text{job}}{\text{time-unit}}]$ into buffer 1. The maximal process rates required at the stored jobs in the buffers are μ_1, μ_2, μ_3 and μ_4 $[\frac{\text{jobs}}{\text{time-unit}}]$ respectively. Before a workstation can start processing jobs from another buffer a setup is required. During a setup the machine in a workstation is cleaned and re-adjusted to be able to produce jobs from another buffer. The times for setting up the machine to buffers 1 and 4 at workstation A will be denoted by σ_{14} and σ_{41} [time-unit], while the times for setting up the machine to buffers 2 and 3 at workstation B are σ_{23} and σ_{32} [time-unit].
All parameter values of this reentrant system are presented in Table 2.1.

Switching between the buffers with stored jobs at the same workstation takes time (setup time) and during that time no jobs can be processed at that workstation, so capacity is lost. The question now arises: When should a machine switch between the jobs in an efficient way without instability of the system? A system is unstable when the total number of jobs in the system explodes as time evolves. Even though for this specific system each machine in each workstation has enough capacity to process all jobs, $\frac{\lambda}{\mu_1} + \frac{\lambda}{\mu_4} < 1$ and $\frac{\lambda}{\mu_2} + \frac{\lambda}{\mu_3} < 1$, it has been shown in [KS90] that since $\frac{\lambda}{\mu_2} + \frac{\lambda}{\mu_4} > 1$ and setup times are all positive, using a clearing policy — process the jobs in a buffer until it is empty, then switch to another buffer — for both workstations results in an unstable system. This instability is not only under continuous job-type flow but also if the job-types are discrete [PJK94]. A machine is processing jobs from a buffer too long, this results in starvation of the other machine and therefore a waste of its capacity. Due to this waste the effective capacity of the other machine is not sufficient anymore, resulting in an unstable system.

In Chapter 4, a global policy is derived according to the approach that will stabilize this reentrant system. Furthermore, the objectives of this approach are to keep the production close to the demand and to keep the WIP inventory level and cycle times as low as possible. This results in a stable reentrant system with a low average number of jobs in the system.
Since arrival rate λ is fixed in this case, Little's Law [Lit61] (2.1) tells us that to reduce the mean time spent in the system, i.e. the mean cycle-time, it suffices to reduce the mean number of jobs in the system, i.e. WIP inventory level, if the system is in steady state ($\lambda = \delta$).

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

With: w = average number of jobs present in the system,
δ = average number of jobs leaving the system per unit of time (mean throughput),
φ = average time spent by a job in the system.

Therefore, the goal of reducing mean cycle-time φ is equivalent to the goal of reducing mean number of jobs w in the system.

2.2 State, input and constraints

Before the approach is investigated, the state, input, constraints and dynamics of the specific system have to be defined first. After that, a desired periodic system behavior of the reentrant system can be determined. Then, a feedback controller is developed which makes the system converge towards this desired periodic system behavior.

State

The state of this system is not only given by the buffer contents x_1, x_2, x_3 and x_4 but also by the remaining setup time at workstation A, x_0^A, the remaining setup time at workstation B, x_0^B, and the current mode of the system.
The current mode of this specific reentrant system (which contains two workstations and each workstation has two buffers) can be defined as $m = (m_A, m_B) \in \{(1,2),(1,3),(4,2),(4,3)\}$.

With: $m = (m_A, m_B) = (1,2)$: workstation A is processing or setting up for step 1 and workstation B is processing or setting up for step 2;
$m = (m_A, m_B) = (1,3)$: workstation A is processing or setting up for step 1 and workstation B is processing or setting up for step 3;
$m = (m_A, m_B) = (4,2)$: workstation A is processing or setting up for step 4 and workstation B is processing or setting up for step 2;
$m = (m_A, m_B) = (4,3)$: workstation A is processing or setting up for step 4 and workstation B is processing or setting up for step 3.

The system state x is given by $\mathrm{x} = (\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4)$.

Input

The input of this system is given by process rates and the current activity of the system. The process rates are denoted by $u_1 \leq \mu_1$, $u_2 \leq \mu_2$, $u_3 \leq \mu_3$ and $u_4 \leq \mu_4$, at which respectively the jobs in buffer 1, buffer 2, buffer 3 and buffer 4 are processed. A machine in a workstation can process the jobs from buffer i (with $i \in \{1,2,3,4\}$) at maximal process rate μ_i, if the machine can continuously process jobs from buffer i without waiting times. If the machine can not continuously process jobs from buffer i — in case when buffer i is empty — the machine can not process jobs at maximal process rate μ_i. It processes the jobs at process rate u_i which is equal to the arrival rate of the jobs into buffer i or equal to zero. When $u_i = 0$, the machine is idling.

Another input of this system is the required activity. Which activity has the system at a certain moment of time? The following activities for workstation A can be distinguished: $u_0^A \in \{❶, ①, ❹, ④\}$.

With: $u_0^A = ❶$: setup workstation A for step 1;
$u_0^A = ①$: process jobs for step 1 at workstation A;
$u_0^A = ❹$: setup workstation A for step 4;
$u_0^A = ④$: process jobs for step 4 at workstation A.

The same holds for workstation B. The following activities can be distinguished: $u_0^B \in \{❷, ②, ❸, ③\}$.

With: $u_0^B = ❷$: setup workstation B for step 2;
$u_0^B = ②$: process jobs for step 2 at workstation B;
$u_0^B = ❸$: setup workstation B for step 3;
$u_0^B = ③$: process jobs for step 3 at workstation B.

The input of this system is given by $(\boldsymbol{u_0}, \boldsymbol{u}) = (u_0^A, u_0^B, u_1, u_2, u_3, u_4)$.

Constraints

In previous section, the inputs of the system are defined but they can not have every possible value. For example, the rates could not be negative and during a setup no jobs can be processed at the workstation. Also, only the jobs can be processed at a workstation for which the workstation has been set up. Therefore, the inputs of this system are bounded by the following constraints.

The process rates can never be negative, they are zero (in case of a setup) or positive. These rates are also bounded by a maximal value, a machine can never process jobs at a rate higher then its maximal rate (μ). These constraints are defined as: $0 \leq u_1 \leq \mu_1$, $0 \leq u_2 \leq \mu_2$, $0 \leq u_3 \leq \mu_3$ and $0 \leq u_4 \leq \mu_4$.

At each time instant, the inputs of workstation A are subject to the constraints:

constraint	activity	process rate for step 1	process rate for step 4	for system state: remaining setup time buffer content, mode
A1	$u_0^A \in \{❶, ❹\}$	$u_1 = 0$	$u_4 = 0$	$x_0^A > 0$
A2	$u_0^A \in \{①, ❹\}$	$u_1 \leq \mu_1$	$u_4 = 0$	$x_0^A = 0$, $x_1 > 0$, $m_A = 1$
A3	$u_0^A \in \{①, ❹\}$	$u_1 \leq \lambda$	$u_4 = 0$	$x_0^A = 0$, $x_1 = 0$, $m_A = 1$
A4	$u_0^A \in \{❶, ④\}$	$u_1 = 0$	$u_4 \leq \mu_4$	$x_0^A = 0$, $x_4 > 0$, $m_A = 4$
A5	$u_0^A \in \{❶, ④\}$	$u_1 = 0$	$u_4 \leq \min(u_3, \mu_4)$	$x_0^A = 0$, $x_4 = 0$, $m_A = 4$

Constraint A1 says that no jobs can be processed at workstation A ($u_1 = 0$ and $u_4 = 0$) when workstation A is setting up ($x_0^A > 0$). In case a setup has been completed, only jobs can be

processed for which the workstation has been set up. At workstation A, this takes place at process rate $u_1 \leq \mu_1$ if jobs are available ($x_1 > 0$) in buffer 1 (constraint A2) and at process rate $u_1 \leq \lambda$ if no jobs are available ($x_1 = 0$) in buffer 1 (constraint A3).
This holds also for production step 4 at this workstation. If jobs are available in buffer 4, the process rate of the machine in workstation A is $u_4 \leq \mu_4$. Furthermore, when buffer 4 is empty the process rate becomes $u_4 \leq u_3$. With these constraints for the process rate of production step 4, buffer 4 will never be negative.

At each time instant, the inputs of workstation B are subject to the following constraints:

constraint	activity	process rate for step 2	process rate for step 3	for system state: remaining setup time, buffer content, mode
B1	$u_0^B \in \{❷, ❸\}$	$u_2 = 0$	$u_3 = 0$	$x_0^B > 0$
B2	$u_0^B \in \{②, ❸\}$	$u_2 \leq \mu_2$	$u_3 = 0$	$x_0^B = 0$, $x_2 > 0$, $m_B = 2$
B3	$u_0^B \in \{②, ❸\}$	$u_2 \leq \min(u_1, \mu_2)$	$u_3 = 0$	$x_0^B = 0$, $x_2 = 0$, $m_B = 2$
B4	$u_0^B \in \{❷, ③\}$	$u_2 = 0$	$u_3 \leq \mu_3$	$x_0^B = 0$, $x_3 > 0$, $m_B = 3$
B5	$u_0^B \in \{❷, ③\}$	$u_2 = 0$	$u_3 = 0$	$x_0^B = 0$, $x_3 = 0$, $m_B = 3$

Constraint B1 says that if workstation B is setting up ($x_0^B > 0$), no jobs can be processed at workstation B ($u_2 = 0$ and $u_3 = 0$). In case a setup has been completed, only jobs can be processed for which the workstation has been set up. At workstation B, this takes place at process rate $u_2 \leq \mu_2$ if jobs are available ($x_2 > 0$) in buffer 2 (constraint B2) and at process rate $u_2 \leq u_1$ if no jobs are available ($x_2 = 0$) in buffer 2 (constraint B3). With these constraints for the process rate of production step 2, buffer 2 will never be negative.
This holds also for production step 3 at this workstation. If jobs are available in buffer 3, the process rate of the machine in workstation B is $u_3 \leq \mu_3$. Furthermore, when buffer 3 is empty the process rate becomes $u_3 \leq u_2$. Workstation B can only process jobs for which the machine has been set up. In this case buffer 3 instead of buffer 2. Therefore, u_2 equals zero.

2.3 Dynamics

The dynamics of this reentrant system are discrete event and continuous, also called hybrid.

The discrete event dynamics of this system can be found in the inputs u_0^A and u_0^B which define the activity of the system. A change in one of these inputs generates an event. For example, if the system is currently in mode $m = (m_A, m_B) = (1, 2)$ and according to the input the current activity becomes "setup to mode $m = (m_A, m_B) = (4, 2)$", then the remaining setup time x_0^A becomes σ_{14} and the system operates in mode $(4, 2)$. This is true for all modes and for every workstation as can be seen in Table 2.3.

	remaining setup time	setting up for mode	current activity and mode
Workstation A	$x_0^A := \sigma_{14}$	$m_A := 4$	$u_0^A =$ ❹ and $m_A = 1$
	$x_0^A := \sigma_{41}$	$m_A := 1$	$u_0^A =$ ❶ and $m_A = 4$
Workstation B	$x_0^B := \sigma_{23}$	$m_B := 3$	$u_0^B =$ ❸ and $m_B = 2$
	$x_0^B := \sigma_{32}$	$m_B := 2$	$u_0^B =$ ❷ and $m_B = 3$

The continuous dynamics can be found in some of the states of the system. For example, the dynamics of the buffer contents and remaining setup times are continuous. During a time interval, the total number of jobs in a buffer propagates over time. Jobs are stored in a buffer or leaving this buffer for processing at a workstation. In general, the nett rate of jobs into a buffer can be denoted by the following continuous time equation:

$$\dot{x}_i(t) = u_{IN_i}(t) - u_{OUT_i}(t). \tag{2.2}$$

With: $\dot{x}_i(t)$ = nett rate of jobs in buffer i,
$u_{IN_i}(t)$ = input rate of jobs into buffer i,
$u_{OUT_i}(t)$ = output rate of jobs that leave buffer i.

The nett rate of jobs into buffer i is equal to the difference between the incoming rate of jobs and the outcoming rate of jobs of this buffer. In case of the reentrant system, the equations for every buffer content can easily be derived while looking at Figure 2.2. These equations can be found in the second column of Table 2.3.

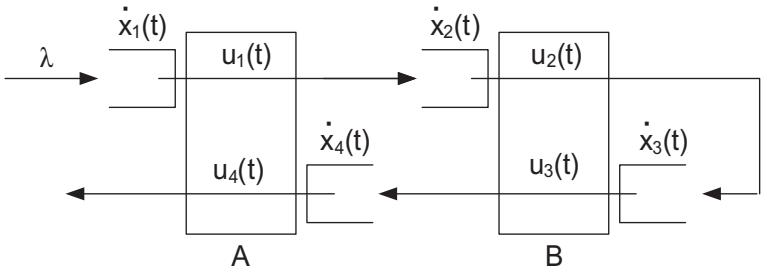


Figure 2.2: Continuous dynamics: Buffer contents.

The dynamics in the calculation of the remaining setup time is also continuous. These equations can be found in the second column of Table 2.3. If a workstation is setting up, i.e. $u_0^A \in \{❶, ❹\}$ and/or $u_0^B \in \{❷, ❸\}$, the remaining setup time is decreased by one unit of time every time instant later, i.e. $\dot{x}_0^i(t) = -1$ for $(i \in \{A, B\})$. If the workstation has been set up, i.e. $u_0^A \in \{①, ④\}$ and/or $u_0^B \in \{②, ③\}$, no calculation is needed for the remaining setup time, i.e. $\dot{x}_0^i(t) = 0$ for $(i \in \{A, B\})$.

	remaining setup time		buffer contents
Workstation A	$\dot{x}_0^A(t) = \begin{cases} -1 & \text{if } u_0^A \in \{❶, ❹\} \\ 0 & \text{if } u_0^A \in \{①, ④\} \end{cases}$		$\dot{x}_1(t) = \lambda - u_1(t)$ $\dot{x}_4(t) = u_3(t) - u_4(t)$
Workstation B	$\dot{x}_0^B(t) = \begin{cases} -1 & \text{if } u_0^B \in \{❷, ❸\} \\ 0 & \text{if } u_0^B \in \{②, ③\} \end{cases}$		$\dot{x}_2(t) = u_1(t) - u_2(t)$ $\dot{x}_3(t) = u_2(t) - u_3(t)$

The structure of the network, the state, input, constraints and hybrid dynamics of the reentrant system are defined. In Chapter 3, this information can be used to derive the desired behavior of the system, which is a starting point of the controller design, i.e. designing an input u which satisfies the constraints and achieves this desired behavior.

Chapter 3

Desired periodic behavior

In previous chapter, the specific reentrant system with setup times has been explained in detail. According to the approach, a desired periodic system behavior should be determined. As already mentioned in the Introduction, the project objective is split into four research objectives. This chapter deals with the first research objective, which was formulated as follows:

Research objective 1:
Determine the desired periodic system behavior.

As mentioned in Section 2.1, a desired periodic system behavior should be determined for which the mean amount of jobs in the system is minimal. From Little's law (2.1) and a constant arrival rate λ for this specific system, it is known that when the mean amount of jobs in the system is minimal, this results in the smallest mean flow time or mean cycle time for this system.
More precisely, a steady state cycle should be derived with respect to minimal weighted work in progress (wip) level to reduce the cycle time and the mean number of jobs in the system.

In general, the weighted wip level function (J) is defined as:

$$J = \frac{1}{T} \int_0^T \sum_{i=1}^{N} c_i x_i(t) dt. \tag{3.1}$$

With: T = cycle period,
N = total number of buffers in the system,
$x_i(t)$ = buffer contents of buffer i at time t,
c_i = weighting factor for buffer contents of buffer i.

For the specific reentrant system with 4 buffers ($N := 4$), this function becomes:

$$J = \frac{1}{T} \int_0^T c_1 x_1(t) + c_2 x_2(t) + c_3 x_3(t) + c_4 x_4(t)\ dt. \tag{3.2}$$

To determine a desired periodic behavior with a small mean number of jobs in the system (wip level), (3.2) with $c_i = 1$ should be minimized over the set of feasible periodic cycles.

3.1 Minimal cycle period

The minimal cycle period is determined first. During a cycle period, the number of jobs that is released into the system has to leave the system too. Otherwise, the total number of jobs in the system is increasing which results in an unstable system.

In the specific reentrant system, each job has to be processed twice at each workstation. At workstation A, this takes $\frac{1}{\mu_1} + \frac{1}{\mu_4} = 0.3 + 0.6 = 0.9$ time-units. During a cycle period, workstation A needs at least two setups to process the jobs for step 1 and step 4. Therefore, in total $\sigma_{14} + \sigma_{41} = 50 + 50 = 100$ time-units are lost due to setups. With a constant arrival rate λ of 1 $[\frac{\text{job}}{\text{time-unit}}]$, during a cycle period of 1000 time-units, the 1000 jobs that arrive in this reentrant system can also be processed by this system.
Since the parameters of workstation B are identical, this analysis holds also for workstation B.

This minimal cycle period can also be found in the literature. For example, Savkin [Sav03] has derived a theorem that calculates the minimal cycle period for deterministic systems with setup times. According to this theorem, the minimal cycle period T is defined by:

$$T = \max_{m=1,\ldots,M} \left[k_m \sigma_m + \sum_{b(p,i)\in\{B_m\}} \frac{d_p}{u_{p,i}} \right]. \tag{3.3}$$

With:
k_m = number of buffers that works with machine m,
σ_m = setup time of machine m,
$b_{p,i}$ = job-type p stored in buffer i that works with machine m,
B_m = set of buffers that works with machine m,
d_p = desired amount of processed jobs of job-type p,
$u_{p,i}$ = process rate for job-type p of buffer i at machine m.

For the specific reentrant manufacturing system, this theorem says that the minimal cycle period of a system equals the largest minimal cycle period of a workstation in that system. This workstation is also called the bottleneck of the system, i.e. the slowest step in the production process. This bottleneck has the largest influence on the minimal cycle period of the system.

In the specific reentrant system, each workstation contains one machine and both workstations are processing two job-types, i.e. each workstation has to process the jobs for two production steps. For the specific reentrant system with two identical workstations, the minimal cycle period becomes:

$$T = 2 \cdot 50 + \frac{1000}{(\frac{1}{0.3})} + \frac{1000}{(\frac{1}{0.6})} = 100 + 300 + 600 = 1000 \text{ time-units}. \tag{3.4}$$

This minimal cycle period is equal to the cycle period determined earlier. But a periodic cycle with the smallest period does not necessarily have the smallest mean amount of jobs in the system [ELR06]. In some cases, a longer period reduces the mean amount of jobs in the

system. But a longer period implies that a workstation does not always process the jobs at its highest possible rate which reduces the production efficiency of the system. On the other hand, a periodic cycle with a smaller period implies that on average more time is wasted on setups. Which cycle period T minimizes the weighted wip level function J? It turns out that a trade-off should be made. A periodic cycle which has to waste capacity by frequent switching or by processing jobs at a rate lower than the maximal process rate.
Therefore, in the analysis for determining the cycle period which minimizes the weighted wip level function J, not only the minimal cycle period ($T = 1000$ time-units) is used, also a longer cycle period ($T \geq 1000$ time-units) is taken into account.

3.2 Increasing weights

After determining the minimal cycle period, several periodic cycles exist which minimize (3.2) (for $c_i = 1$). This number of possible periodic cycles decreases when the weight for each buffer is defined.

In [LR06a], the weight for each buffer is associated with the mean amount of work, i.e. the total remaining process time of a job stored in that buffer. A job which is in buffer 1 needs 0.3 time-units processing for step 1, 0.6 time-units for step 2, 0.3 time-units for step 3 and 0.6 time-units for step 4, see also Figure 2.1. So the weight or amount of work associated with a job in buffer 1 equals $[0.3 + 0.6 + 0.3 + 0.6 =] 1.8$. The same calculation can be done for all buffers in this 4 steps production process. For this case, weighted wip level function J becomes:

$$J = \frac{1}{T} \int_0^T 1.8x_1(t) + 1.5x_2(t) + 0.9x_3(t) + 0.6x_4(t)\ dt. \tag{3.5}$$

In case of [LR06a], during a cycle period the weights for each buffer in the 4 steps production process are decreasing ($c_1 > c_2 > c_3 > c_4 \triangleq 1.8 > 1.5 > 0.9 > 0.6$). In this report, the same case is considered but now with increasing weights. This case can be found in manufacturing systems where more value is added to the products after every step during a production cycle. This case most likely leads to another "weighted wip level function J" which should be minimized to find the new periodic system behavior. The weights are associated with holding costs for storing a job in a buffer. A periodic cycle should be found for which the mean amount of jobs in buffer 4 is as low as possible. Buffer 4 is the most expensive buffer, while buffer 1 is the cheapest one. This case leads to another system behavior than the case investigated in [LR06a]. For simplicity, the following holding costs as weights are defined: $c_1 < c_2 < c_3 < c_4 \triangleq 1 < 2 < 3 < 4$. But other values which satisfy $c_1 < c_2 < c_3 < c_4$ can be used as well. The weighted wip level function J becomes:

$$J = \frac{1}{T} \int_0^T 1x_1(t) + 2x_2(t) + 3x_3(t) + 4x_4(t)\ dt. \tag{3.6}$$

3.3 Optimal sequence of process rates for each system mode

In the four steps production process of the specific reentrant system, only the jobs can be processed at a workstation for which the workstation has been set up. Therefore, the system can only operate in one of the four modes each time, as explained in Section 2.2. Each workstation can process the jobs for step i in a mode with rate $u_i(t)$ which is subject to the constraint $0 \leq u_i(t) \leq \mu_i$. The question now arises: "Which sequence of process rates $u_i(t)$ in each system mode leads to a system behavior for which the weighted amount of jobs is as low as possible?". In this section, each system mode in this production cycle is investigated independently of each other and in random order to determine the optimal sequence of process rates. In next section, the duration of every action is calculated for each workstation. When these durations are known, the system modes can be scheduled into one schedule which results in good desired system behavior.

In this section, each system mode in this production cycle is investigated independently of each other and in random order to determine the optimal sequence of process rates for which the weighted amount of jobs in the system is minimal. For each system mode, an optimization problem is defined with the weighted wip level function J as objective function. Also, some constraints are defined which describes the conditions of the process rates and buffer contents. Each optimization problem has to be solved by finding a solution in the feasible region — where each solution satisfies all the constraints — which has the minimum value of the objective function. In this case, the optimal sequence of process rates for the production steps in the system mode should be determined subject to the constraints to minimize the weighted amount of jobs in the system.

System mode = (4,3)

In Figure 3.1, the specific reentrant system operates in mode $(m_A, m_B) = (4, 3)$. Workstation A processes the jobs for step 4 and workstation B processes the jobs for step 3. In this mode, the arrival rates into buffer 2 and buffer 3 are zero ($\lambda = 0$). While the arrival rate into buffer 1 is always equal to $\lambda = 1$ $[\frac{\text{job}}{\text{time-unit}}]$.

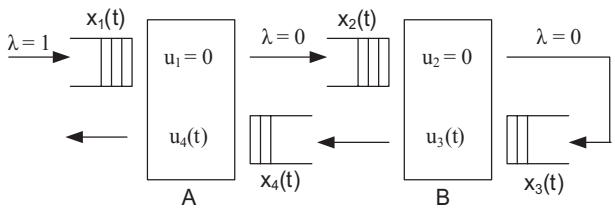


Figure 3.1: System operates in mode (4,3).

To minimize the weighted wip level function J (3.6) for this mode, the following optimization

problem has to be solved, for given t_0, t_f, x_3^0, x_3^f, x_4^0 and x_4^f.

$$\begin{aligned}
&\min_{u_3(t),u_4(t)} \int_{t_0}^{t_f} c_3x_3(\tau) + c_4x_4(\tau)\ d\tau,\\
&\quad \text{with } c_3 < c_4 \text{ and } \mu_3 > \mu_4.\\
&\text{s.t. } x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ 0 \le u_3(t) \le \mu_3,\ x_3(t) \ge 0,\\
&\quad x_4(t_0) = x_4^0,\ x_4(t_f) = x_4^f,\ 0 \le u_4(t) \le \mu_4,\ x_4(t) \ge 0.\\
&\text{where}\quad 0 \le x_3^0 - x_3^f \le \mu_3(t_f - t_0),\\
&\qquad\quad 0 \le (x_3^0 + x_4^0) - (x_3^f + x_4^f) \le \mu_4(t_f - t_0).
\end{aligned}$$

During mode $(m_A, m_B) = (4, 3)$ which has a duration of $(t_f - t_0)$, the process rates $u_3(t)$ and $u_4(t)$ should be determined subject to the constraints to minimize the weighted amount of jobs in buffers 3 and 4, i.e. $\min \int_{t_0}^{t_f} c_3x_3(\tau) + c_4x_4(\tau)\ d\tau$. One of the constraints says that the buffer contents of buffer 3 can never decrease faster than the case when workstation B processes the jobs for step 3 at maximal rate, i.e. $x_3^0 - x_3^f \le \mu_3(t_f - t_0)$. Due to an arrival rate equal to zero into buffer 3 in this case, the buffer contents of buffer 3 will never increase during this mode. The initial amount of jobs at t_0 in this mode is never smaller than the amount of jobs at t_f, i.e. $x_3^0 - x_3^f \ge 0$. The same analysis holds for the total buffer contents of buffers 3 and 4. The nett amount of jobs in buffers 3 and 4 can never decrease faster than the case when workstation A processes the jobs at maximal rate, i.e. $(x_3^0 + x_4^0) - (x_3^f + x_4^f) \le \mu_4(t_f - t_0)$. Also, the nett amount of jobs in buffers 3 and 4 can never be negative, i.e. $(x_3^0 + x_4^0) - (x_3^f + x_4^f) \ge 0$.

Before this optimization problem will be solved, the weighted wip level function J in this optimization problem is rewritten as:

$$\begin{aligned}
&\min_{u_3(t),u_4(t)} \left(c_4 \int_{t_0}^{t_f} x_3(\tau) + x_4(\tau)\ d\tau - (c_4 - c_3) \int_{t_0}^{t_f} x_3(\tau)\ d\tau \right),\\
&\quad \text{with } c_3 < c_4 \text{ and } \mu_3 > \mu_4.\\
&\text{s.t. } x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ 0 \le u_3(t) \le \mu_3,\ x_3(t) \ge 0,\\
&\quad x_4(t_0) = x_4^0,\ x_4(t_f) = x_4^f,\ 0 \le u_4(t) \le \mu_4,\ x_4(t) \ge 0.\\
&\text{where}\quad 0 \le x_3^0 - x_3^f \le \mu_3(t_f - t_0),\\
&\qquad\quad 0 \le (x_3^0 + x_4^0) - (x_3^f + x_4^f) \le \mu_4(t_f - t_0).
\end{aligned}$$

In this form, we can see that the weighted wip level function J is minimized when the first part of this function, i.e. $c_4 \int_{t_0}^{t_f} x_3(\tau) + x_4(\tau)\ d\tau$ is minimal and the second part of this function, i.e. $(c_4 - c_3) \int_{t_0}^{t_f} x_3(\tau)\ d\tau$, is maximal.

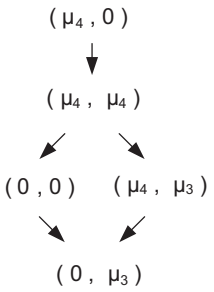
Solving this optimization problem for system mode $(4, 3)$, workstations A and B can process the jobs depending on the amount of jobs in the buffers at the following process rates with the corresponding requirements.

$$(u_4(t), u_3(t)) = \begin{cases}
(0, \mu_3) & \text{if } x_3(t) - x_3^f = \mu_3(t_f - t) \text{ and } x_{34}(t) = x_{34}^f;\\
(\mu_4, \mu_3) & \text{if } x_3(t) - x_3^f = \mu_3(t_f - t) \text{ and } x_{34}(t) > x_{34}^f;\\
(0, 0) & \text{if } x_3(t) - x_3^f < \mu_3(t_f - t) \text{ and } x_{34}(t) = x_{34}^f;\\
(\mu_4, \mu_4) & \text{if } x_3(t) - x_3^f < \mu_3(t_f - t) \text{ and } x_{34}(t) > x_{34}^f \text{ and } x_4(t) = 0;\\
(\mu_4, 0) & \text{if } x_3(t) - x_3^f < \mu_3(t_f - t) \text{ and } x_{34}(t) > x_{34}^f \text{ and } x_4(t) > 0.
\end{cases}$$

where $x_{34}(t) = x_3(t) + x_4(t)$ and $x_{34}^f = x_3^f + x_4^f$.

All these possible process rates with the corresponding requirements for both workstations are solutions of the optimization problem. But which sequence of process rates lead to a system behavior in mode $(m_A, m_B) = (4, 3)$ for which the weighted amount of jobs in this mode is as low as possible? A proof of this statement is given after Lemma 3.1.

Lemma 3.1. *The optimal behavior for the system in Figure 3.1 which operates in mode* $(m_A, m_B) = (4, 3)$ *is achieved when the system processes the jobs during this system mode in the following sequence of process rates.*



During this system mode, workstation A processes the jobs for step 4 at maximal rate μ_4 *and then it idles. Workstation B idles first and processes then the jobs for step 3 eventually at rate* μ_4 *— this action only occurs when buffer 4 is empty and workstation A processes the jobs for step 4 — and at the end of this mode workstation B processes the jobs for step 3 at maximal rate* μ_3*. Given the boundary wip levels, both sequences can result in an optimal behavior for this system mode for which the weighted wip level is as low as possible.*

Proof. For system mode $(4, 3)$, an optimal behavior has to be determined for which the weighted amount of jobs in buffers 3 and 4 is minimal. Looking at the optimization problem, the first part of the weighted wip level function — $c_4 \int_{t_0}^{t_f} x_3(\tau) + x_4(\tau)\, d\tau$ — has to minimized and the second part of this function — $(c_4 - c_3) \int_{t_0}^{t_f} x_3(\tau)\, d\tau$ — has to maximized. For each sequence of process rates in Lemma 3.1, both parts of the weighted wip level function are sketched in Figure 3.2.

First, the left sequence of process rates in Lemma 3.1 is considered. Suppose a behavior is given for the buffer contents of buffers 3 and 4 which is sketched in the left hand side of Figure 3.2. After having completed the setup to process the jobs for steps 3 and 4, buffers 3 and 4 contain an amount of x_3^0 and x_4^0 jobs respectively at t_0 and at the end of this mode, i.e. at t_f, buffers 3 and 4 contain an amount of x_3^f and x_4^f jobs respectively. The amount of jobs at t_0 and t_f are known and fixed. The behavior between t_0 and t_f should be optimized for which the weighted amount of jobs in buffers 3 and 4 is minimal.

In the left hand side of Figure 3.2, the buffer contents of buffer 3 and the sum of the buffer contents of buffers 3 and 4 is sketched. Due to the fact that the arrival rate into buffer 3 equals zero, the total number of jobs in buffers 3 and 4 can never increase during this mode. This total amount of jobs only decreases when the jobs are processed for step 4. Workstation B idles first for a duration until buffer 4 becomes empty. Then, workstation B processes the jobs of buffer 3 at process rate μ_4 until the constraint $x_{34}(t) = x_{34}^f$, i.e. the total number of

jobs at t is equal to total number of jobs at t_f, is satisfied. During this action, workstation A processes also the jobs of buffer 4 at maximal rate μ_4 and the sum between x_3 and x_4 decreases during these actions with rate μ_4. When the total number of jobs at t is equal to total number of jobs at t_f, both workstations idle for a duration until workstation B can process an amount of x_4^f jobs in $(t_f - t)$ time-units. At the end of this mode, workstation B processes the jobs of buffer 3 at maximal rate μ_3 until buffer 4 contains an amount of x_4^f jobs at t_f. During the last two actions, workstation A idles and no jobs are leaving the system. Therefore, the sum between x_3 and x_4 remains constant. Clearly, during this mode the total number of jobs in buffers 3 and 4 can not decrease faster than in this case. Therefore, the first part of the weighted wip level function J — $c_4 \int_{t_0}^{t_f} x_3(\tau) + x_4(\tau)\ d\tau$ — is minimized. Also, the second part of this weighted wip level function — $(c_4 - c_3) \int_{t_0}^{t_f} x_3(\tau)\ d\tau$ — is maximized for this case. With the optimal behavior of buffer 4 in mind and the fact that the holding cost for storing a job in buffer 4 is more expensive than in buffer 3, the jobs stay in buffer 3 in the beginning of this mode and at the end the jobs can not leave buffer 3 faster than in this case. Given the boundary wip levels, this sequence of process rates leads to an optimal behavior for this system mode.

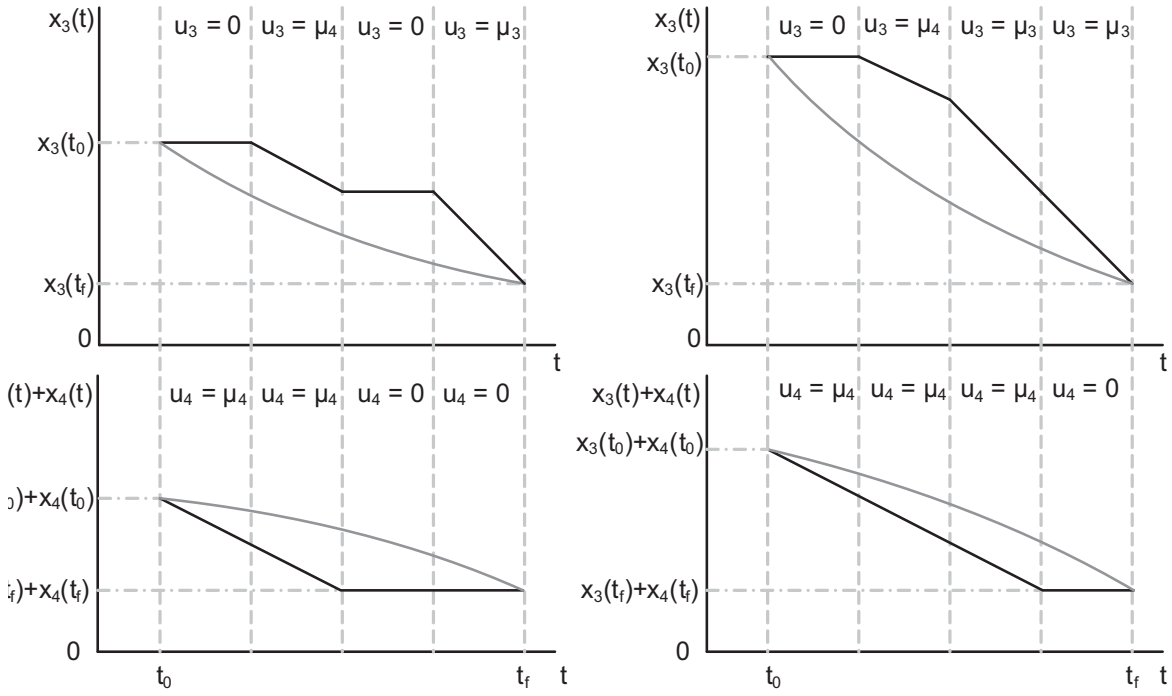


Figure 3.2: Optimal sequence of process rates for system mode (4,3).

According to Lemma 3.1 and depending on the boundary wip levels, the right sequence of process rates can also lead to optimal behavior for this mode. Looking better at the sequences in Lemma 3.1, the only difference between these two sequences can be found in the third action, i.e. when both workstations are idling in the left sequence or when both workstations are processing the jobs at maximal rate in the right sequence. Depending on the fact which of the two constraints — $x_{34}(t) = x_{34}^f$ or $x_3(t) - x_3^f = \mu_3(t_f - t)$ — is satisfied first, the left or right sequence in Lemma 3.1 can result in optimal behavior for this mode. In case when the total amount of jobs in buffers 3 and 4 at time t equals the amount at the end of this

mode, i.e. when the constraint $x_{34}(t) = x_{34}^f$ is satisfied first, the left sequence of process rates in Lemma 3.1 will result in optimal behavior for this mode as explained earlier. In case when workstation B can process an amount of x_4^f jobs at maximal rate μ_3 in $(t_f - t)$ time-units while the constraint $x_{34}(t) = x_{34}^f$ is not satisfied, then the right sequence of process rates of Lemma 3.1 will result in optimal behavior for this system mode. This is behavior is sketched in the right hand side of Figure 3.2. To make sure that buffer 4 contains at the end of this mode an amount of x_4^f jobs, workstation B has to process the jobs for step 3 during the third action instead of idling. During this action, workstation A also processes the jobs for step 4 to reduce the weighted wip level in this mode. When the total amount of jobs in buffers 3 and 4 at time t are equal to the amount at the end of this mode, i.e. when the constraint $x_{34}(t) = x_{34}^f$ is satisfied, workstation A idles and workstation B continues processing the jobs for step 3 at maximal rate μ_3 to make sure that buffer 4 contains an amount of x_4^f jobs at the end of this mode. Also for this case, the total number of jobs in buffers 3 and 4 can not decrease faster during this mode. Therefore, the first part of the weighted wip level function J — $c_4 \int_{t_0}^{t_f} x_3(\tau) + x_4(\tau)\, d\tau$ — is minimized. Also, the second part of this weighted wip level function — $(c_4 - c_3) \int_{t_0}^{t_f} x_3(\tau)\, d\tau$ — is maximized for this case. With the optimal behavior of buffer 4 in mind and the fact that the holding cost for storing a job in buffer 4 is more expensive than in buffer 3, the jobs stay in buffer 3 in the beginning of this mode and at the end the jobs can not leave buffer 3 faster than in this case. Given the boundary wip levels, this sequence of process rates leads to an optimal behavior for this system mode.

System mode (4,2)

In Figure 3.3, the specific reentrant system operates in mode $(m_A, m_B) = (4, 2)$. Workstation A processes the jobs for step 4 and workstation B processes the jobs for step 2. In this case, the arrival rates into buffers 2 and 4 are zero and the arrival rate into buffer 3 depends on the process rate at workstation B. While the arrival rate into buffer 1 is always equal to $\lambda = 1$ $[\frac{\text{job}}{\text{time-unit}}]$.

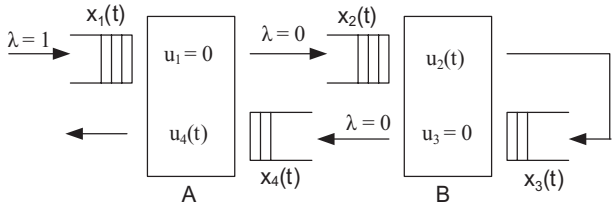


Figure 3.3: System operates in mode (4,2).

To minimize the weighted wip level function J (3.6) for this mode, the following optimization

problem has to be solved, for given t_0, t_f, x_2^0, x_3^0, x_4^0, x_2^f, x_3^f and x_4^f.

$$\begin{aligned}
&\min_{u_2(t),u_4(t)} \int_{t_0}^{t_f} c_2x_2(\tau) + c_3x_3(\tau) + c_4x_4(\tau)\ d\tau,\\
&\quad \text{with } c_2 < c_3 < c_4 \text{ and } \mu_2 = \mu_4.\\
&\text{s.t. } x_2(t_0) = x_2^0,\ x_2(t_f) = x_2^f,\ 0 \le u_2(t) \le \mu_2,\ x_2(t) \ge 0,\\
&\quad x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ u_3(t) = 0,\ x_3(t) \ge 0,\\
&\quad x_4(t_0) = x_4^0,\ x_4(t_f) = x_4^f,\ 0 \le u_4(t) \le \mu_4,\ x_4(t) \ge 0.\\
&\text{where } 0 \le x_2^0 - x_2^f \le \mu_2(t_f - t_0),\\
&\qquad 0 \le x_4^0 - x_4^f \le \mu_4(t_f - t_0),\\
&\qquad x_2^0 + x_3^0 = x_2^f + x_3^f.
\end{aligned}$$

During mode $(m_A, m_B) = (4, 2)$ which has a duration of $(t_f - t_0)$, the process rates $u_2(t)$ and $u_4(t)$ should be determined subject to the constraints to minimize the weighted amount of jobs in buffers 2, 3 and 4, i.e. $\min \int_{t_0}^{t_f} c_2x_2(\tau) + c_3x_3(\tau) + c_4x_4(\tau)\ d\tau$. One of the constraints says that the buffer contents of buffer 2 can never decrease faster than the case when workstation B processes the jobs for step 2 at maximal rate, i.e. $x_2^0 - x_2^f \le \mu_2(t_f - t_0)$. Due to an arrival rate equal to zero in this case, the buffer contents of buffer 2 will never increase during this mode. Thus, the initial amount of jobs at t_0 is never smaller than the amount of jobs at t_f, i.e. $x_2^0 - x_2^f \ge 0$. During this mode, no jobs will leave buffer 3. Therefore the total number of jobs in buffers 2 and 3 at t_0 equals the total number of jobs in buffers 2 and 3 at t_f, i.e. $x_2^0 + x_3^0 = x_2^f + x_3^f$. The same analysis holds for the buffer contents of buffer 4.

In this mode, the arrival rates between the workstations are zero. This means that both workstations can operate independently of each other and the optimization problem can be solved for each workstation separately.

First, consider workstation B for which the following optimization problem should be solved for given t_0, t_f, x_2^0, x_3^0, x_2^f and x_3^f.

$$\begin{aligned}
&\min_{u_2(t)} c_2 \int_{t_0}^{t_f} (x_2(\tau) + x_3(\tau))\ d\tau + (c_3 - c_2) \int_{t_0}^{t_f} x_3(\tau)\ d\tau,\\
&\quad \text{with } c_2 < c_3.\\
&\text{s.t. } x_2(t_0) = x_2^0,\ x_2(t_f) = x_2^f,\ 0 \le u_2(t) \le \mu_2,\ x_2(t) \ge 0,\\
&\quad x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ u_3(t) = 0,\ x_3(t) \ge 0.\\
&\text{where } 0 \le x_2^0 - x_2^f \le \mu_2(t_f - t_0),\\
&\qquad x_2^0 + x_3^0 = x_2^f + x_3^f.
\end{aligned}$$

Lemma 3.2. *The optimal behavior for which the weighted wip level is minimal for workstation B which operates in mode* $(m_A, m_B) = (4, 2)$ *is achieved when workstation B processes the jobs in the following sequence.*

$$\begin{aligned}
u_2(t) &= 0 \quad \textit{if} \quad x_3^f - x_3(t) < \mu_2(t_f - t);\\
u_2(t) &= \mu_2 \quad \textit{if} \quad x_3^f - x_3(t) = \mu_2(t_f - t).
\end{aligned}$$

Given the boundary wip levels, this optimal behavior results in the smallest weighted wip level for this system mode.

Proof. The total amount of jobs in buffers 2 and 3 remains constant during this system mode. The arrival rate into buffer 2 is equal to zero and no jobs are leaving buffer 3 in this case. Looking at the optimization problem for workstation B, the sum between $x_2(t)$ and $x_3(t)$ in the first part of the function is constant and can not be minimized. But the second part, which contains the buffer contents of buffer 3, can be minimized. Therefore, Workstation B should adapt its process rate for step 2 to minimize the weighted amount of jobs in buffer 3. In Figure 3.4, the buffer contents of buffer 3 is sketched during this mode. With the sequence of process rates in Lemma 3.2 in mind, workstation B first idles for a duration until the constraint $x_3^f - x_3(t) = \mu_2(t_f - t)$ — the desired amount of jobs in buffer 3 can be processed at maximal rate μ_2 at the end of this mode — is satisfied. Then, workstation B processes the jobs for step 2 at maximal rate μ_2 until buffer 3 contains an amount of x_3^f jobs at the end of this mode. In this case, the buffer contents of buffer 3 is as low as possible in the beginning of this mode and at the end buffer 3 can never increase faster than in this case. Therefore, the weighted amount of jobs in buffer 3 is minimal and this sequence of process rates leads to optimal behavior for workstation B in this mode.

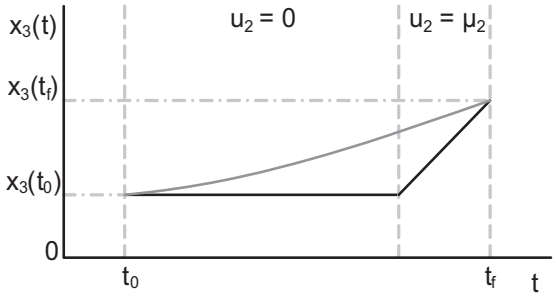


Figure 3.4: Optimal buffer behavior of buffer 3 in mode (4,2).

The same analysis can be done for workstation A for which the following optimization problem should be solved for given t_0, t_f, x_4^0 and x_4^f.

$$\begin{aligned}
&\min_{u_4(t)} \int_{t_0}^{t_f} c_4 x_4(\tau)\, d\tau. \\
&\text{s.t. } x_4(t_0) = x_4^0,\ x_4(t_f) = x_4^f,\ 0 \le u_4(t) \le \mu_4,\ x_4(t) \ge 0, \\
&\qquad 0 \le x_4^0 - x_4^f \le \mu_4(t_f - t_0).
\end{aligned}$$

Lemma 3.3. *The optimal behavior for which the weighted wip level is minimal for workstation A which operates in mode $(m_A, m_B) = (4, 2)$ is achieved when workstation A processes the jobs in the following sequence.*

$$\begin{aligned}
u_4(t) &= \mu_4 \quad \textit{if} \quad x_4(t) > x_4^f; \\
u_4(t) &= 0 \quad \textit{if} \quad x_4(t) = x_4^f.
\end{aligned}$$

Given the boundary wip levels, this optimal behavior results in the smallest weighted wip level for this system mode.

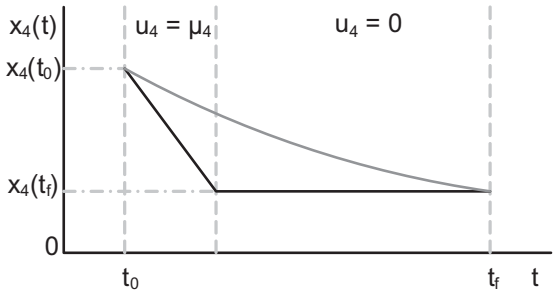


Figure 3.5: Optimal buffer behavior of buffer 4 in mode (4,2).

Proof. Suppose a behavior is given for the buffer contents of buffer 4 which is sketched in Figure 3.5. After having completed the setup to process the jobs for step 4, buffer 4 contains an amount of x_4^0 jobs at t_0 and at the end of this mode, at t_f, buffer 4 contains an amount of x_4^f jobs. Consider now the case when the sequence of process rates in Lemma 3.3 is used. This result should lead to an optimal behavior for this mode for which the weighted amount of jobs in buffer 4 is as low as possible. Workstation A starts processing the jobs for step 4 at maximal rate μ_4 until buffer 4 contains an amount of x_4^f jobs. Then, workstation A idles. This case is also sketched in Figure 3.5. Clearly, during this mode the number of jobs in buffer 4 can not decrease faster than in this case. Therefore, at each time instant the wip level in buffer 4 is minimal for this case which results in optimal behavior for workstation A in this mode.

The optimal sequence of process rates is determined for each workstation separately in this system mode. But with the results in Lemma 3.2 and 3.3 in mind, the optimal sequence of process rates for this system for which the weighted wip level is minimal is achieved when both workstations process the jobs in the following sequence of process rates depending on the wip levels in the beginning and at the end of this system mode.

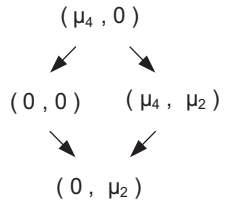


Figure 3.6: Optimal sequence of process rates for system mode (4,2).

System mode (1,2)

In Figure 3.7, the specific reentrant system operates in mode $(m_A, m_B) = (1, 2)$. Workstation A processes the jobs for step 1 and workstation B processes the jobs for step 2. In this case, the arrival rate into buffer 4 is zero and the arrival rates into buffers 2 and 3 depend on the process rates at the workstations. Whereas the arrival rate into buffer 1 is always equal to $\lambda = 1\ [\frac{\text{job}}{\text{time-unit}}]$.

To minimize the weighted wip level function J (3.6) for this mode, the following optimization

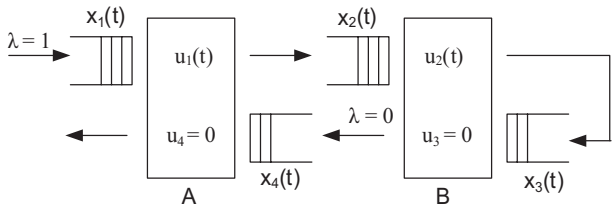


Figure 3.7: System operates in mode (1,2).

problem has to be solved, for given t_0, t_f, x_1^0, x_2^0, x_3^0, x_1^f, x_2^f and x_3^f.

$$\begin{aligned}
&\min_{u_1(t),u_2(t)} \int_{t_0}^{t_f} c_1x_1(\tau) + c_2x_2(\tau) + c_3x_3(\tau)\ d\tau,\\
&\quad \text{with } c_1 < c_2 < c_3 \text{ and } \mu_1 > \mu_2.\\
&\text{s.t. } x_1(t_0) = x_1^0,\ x_1(t_f) = x_1^f,\ 0 \le u_1(t) \le \mu_1,\ x_1(t) \ge 0,\\
&\quad x_2(t_0) = x_2^0,\ x_2(t_f) = x_2^f,\ 0 \le u_2(t) \le \mu_2,\ x_2(t) \ge 0,\\
&\quad x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ u_3(t) = 0,\ x_3(t) \ge 0,\\
&\text{where } (\lambda - \mu_1)(t_f - t_0) \le x_1^f - x_1^0 \le \lambda(t_f - t_0),\\
&\qquad (\lambda - \mu_2)(t_f - t_0) \le (x_1^f + x_2^f) - (x_1^0 + x_2^0) \le \lambda(t_f - t_0),\\
&\qquad x_1^0 + x_2^0 + x_3^0 + \lambda(t_f - t_0) = x_1^f + x_2^f + x_3^f.
\end{aligned}$$

During mode $(m_A, m_B) = (1, 2)$ which has a duration of $(t_f - t_0)$, the process rates $u_1(t)$ and $u_2(t)$ should be determined subject to the constraints to minimize the weighted number of jobs in buffers 1, 2 and 3, i.e. $\min \int_{t_0}^{t_f} c_1x_1(\tau) + c_2x_2(\tau) + c_3x_3(\tau)\ d\tau$. During a cycle period, the buffer contents of buffer 1 always increases with rate λ. Therefore, this buffer contents can never increase faster than with this rate, i.e. $x_1^f - x_1^0 \le \lambda(t_f - t_0)$. Also, this buffer contents can never decrease faster than the case when workstation A processes the jobs at maximal rate, i.e. $x_1^0 - x_1^f \le (\mu_1 - \lambda)(t_f - t_0)$. The total amount of jobs in buffers 1 and 2 increases with rate λ and it only decreases when workstation B is processing the jobs for step 2. Therefore, the total amount of jobs in buffers 1 and 2 can never increase faster than rate λ and it can never decrease faster than the case when workstation B processes the jobs at maximal rate, i.e. $(\lambda - \mu_2)(t_f - t_0) \le (x_1^f + x_2^f) - (x_1^0 + x_2^0) \le \lambda(t_f - t_0)$. It is also known, that the total amount of jobs in buffers 1, 2 and 3 can never decrease during this mode, no jobs are leaving buffer 3. But this total amount of jobs is always increasing with rate λ. This is denoted by $x_1^0 + x_2^0 + x_3^0 + \lambda(t_f - t_0) = x_1^f + x_2^f + x_3^f$.

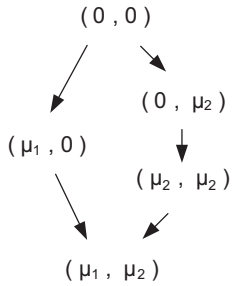
Solving the optimization problem for this system, workstations A and B can process the jobs depending on the amount of jobs in the buffers at the following process rates with the corresponding requirements.

$$(u_1(t), u_2(t)) = \begin{cases}
(0,0) & \text{if } x_3^f - x_3(t) < \mu_2(t_f - t) \text{ and } x_{23}^f - x_{23}(t) < \mu_1(t_f - t);\\
(\mu_2, \mu_2) & \text{if } x_3^f - x_3(t) = \mu_2(t_f - t) \text{ and } x_{23}^f - x_{23}(t) < \mu_1(t_f - t) \text{ and } x_2(t) = 0;\\
(\mu_1, \mu_2) & \text{if } x_3^f - x_3(t) = \mu_2(t_f - t) \text{ and } x_{23}^f - x_{23}(t) = \mu_1(t_f - t);\\
(\mu_1, 0) & \text{if } x_3^f - x_3(t) < \mu_2(t_f - t) \text{ and } x_{23}^f - x_{23}(t) = \mu_1(t_f - t);\\
(0, \mu_2) & \text{if } x_3^f - x_3(t) = \mu_2(t_f - t) \text{ and } x_{23}^f - x_{23}(t) < \mu_1(t_f - t) \text{ and } x_2(t) > 0.
\end{cases}$$

where $x_{23}(t) = x_2(t) + x_3(t)$ and $x_{23}^f = x_2^f + x_3^f$.

All these possible process rates with their corresponding requirements for both workstations are solutions of the optimization problem. But which sequence of process rates lead to a system behavior in mode $(m_A, m_B) = (1, 2)$ for which the weighted amount of jobs in this mode is minimal? A proof of this statement is given after Lemma 3.4.

Lemma 3.4. *The optimal behavior for the system in Figure 3.7 which operates in mode* $(m_A, m_B) = (1, 2)$ *is achieved when the system processes the jobs during this system mode in the following sequence of process rates.*



During this system mode, workstation A first idles for a duration. Then, it processes the jobs for step 1 eventually at rate μ_2 *— this action only occurs when buffer 2 is empty and workstation B processes the jobs for step 2 — and at the end of this mode workstation A processes the jobs for step 1 at maximal rate* μ_1*. Workstation B also idles first for a duration and then it processes the jobs for step 2 at maximal rate* μ_2*. Given the boundary wip levels, both sequences can result in an optimal behavior for which the weighted wip level is minimal for this system mode.*

Before the sequences in Lemma 3.4 is proved, the weighted wip level function is rewritten while the constraints are not changed. This rewritten function is used for determining the optimal behavior for this system mode.

$$\min_{u_1(t),u_2(t)} c_1 \int_{t_0}^{t_f} \Big(x_1(\tau) + x_2(\tau) + x_3(\tau) \Big)\, d\tau + (c_2 - c_1) \int_{t_0}^{t_f} \Big(x_2(\tau) + x_3(\tau) \Big)\, d\tau + (c_3 - c_2) \int_{t_0}^{t_f} x_3(\tau)\, d\tau.$$

Proof. The total amount of jobs in buffers 1, 2 and 3 will never decrease during this system mode, no jobs are leaving buffer 3. With an arrival rate equal to $\lambda = 1\ [\frac{\text{job}}{\text{time-unit}}]$ into buffer 1, the total amount of jobs is always increasing with this rate. The first part of the optimization function, which contains the sum of $x_1(t)$, $x_2(t)$ and $x_3(t)$ is fixed and can not be minimized. The second part of this function, which contains the sum of $x_2(t)$ and $x_3(t)$ can be minimized. The sum of $x_2(t)$ and $x_3(t)$ during this mode is sketched in Figure 3.8. Workstation B should adapt its process rate for step 2 to minimize the weighted amount of jobs in buffer 3. When the weighted amount of jobs in buffer 3 is minimized, then the third part of the optimization function is also minimized which is also sketched in the figure.

Consider now the case when the right sequence of process rates in Lemma 3.4 is used. This case should lead to an optimal behavior for this system mode for which the weighted wip level

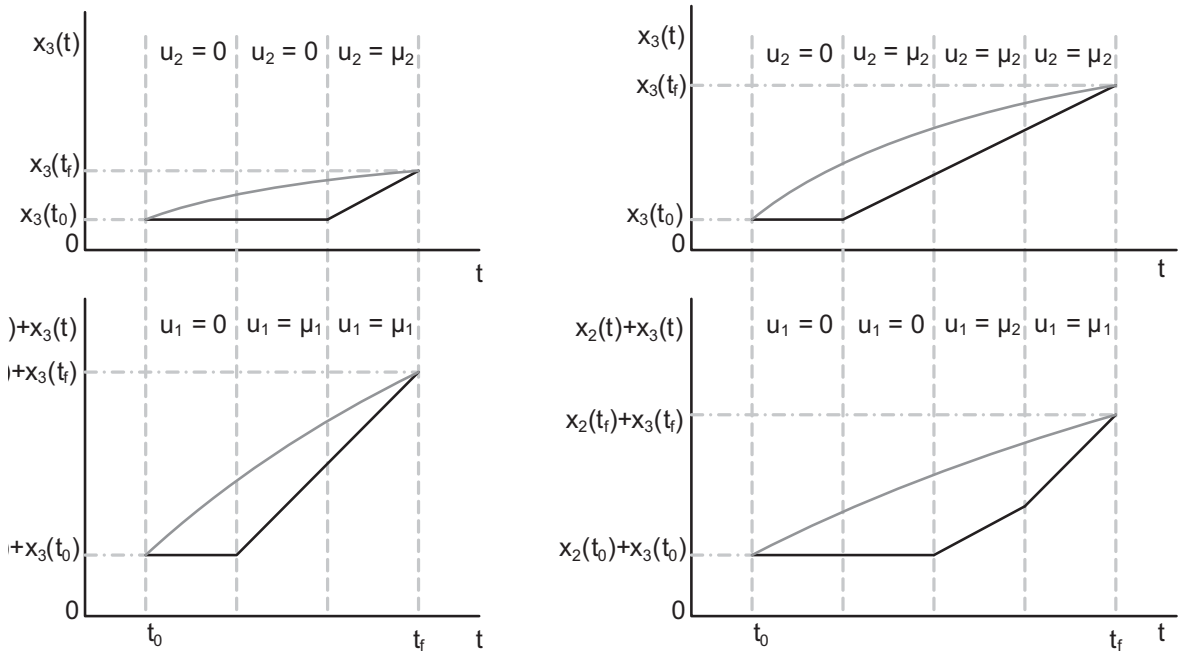


Figure 3.8: Optimal buffer behavior of buffers 2 and 3 in mode (1,2).

is minimal. This behavior is sketched in the left hand side of Figure 3.8. To minimize the weighted amount of jobs in buffers 2 and 3, both workstations idle first for a duration until the constraint $x_3^f - x_3(t) = \mu_2(t_f - t)$, i.e. the amount of $x_3^f - x_3(t)$ jobs can be processed at maximal rate μ_2 during $(t_f - t)$, is satisfied. Then workstation B processes the jobs for step 2 at maximal rate μ_2. Workstation A starts only processing the jobs for step 1 at rate μ_2 if buffer 2 is empty and workstation B processes the jobs for step 2 at rate μ_2 otherwise it idles. Workstation A starts processing the jobs for step 1 at maximal rate μ_1 if the constraint $x_{23}^f - x_{23}(t) = \mu_1(t_f - t)$, i.e. the amount of $x_{23}^f - x_{23}(t)$ jobs can be processed at maximal rate μ_1 during $(t_f - t)$, is satisfied. In this case, the weighted amount of jobs in buffers 2 and 3 are as low as possible in the beginning of this mode and at the end they can never increase faster than in this case. Therefore, the weighted wip level function J for this system mode is minimized. Given the boundary wip levels, the right sequence of process rates in Lemma 3.4 leads to an optimal behavior for this system mode for which the weighted amount of jobs in buffers 2 and 3 is minimal.

According to Lemma 3.4 and depending on the boundary wip levels, the left sequence of process rates can also lead to optimal behavior for this system mode. Depending on the fact which of the two constraints — $x_3^f - x_3(t) = \mu_2(t_f - t)$ or $x_{23}^f - x_{23}(t) = \mu_1(t_f - t)$ —- is satisfied first, the left or right sequence in Lemma 3.4 can result in optimal behavior for this mode. The case when the constraint $x_3^f - x_3(t) = \mu_2(t_f - t)$ is satisfied earlier than the constraint $x_{23}^f - x_{23}(t) = \mu_1(t_f - t)$ is sketched in the right hand side of Figure 3.8. In this case, workstation B starts processing the jobs earlier at maximal rate than workstation A. This behavior is explained above and given the boundary wip levels, the right sequence of process rates in Lemma 3.4 will result in optimal behavior for this mode. The case when the constraint $x_{23}^f - x_{23}(t) = \mu_1(t_f - t)$ is satisfied earlier than the constraint $x_3^f - x_3(t) = \mu_2(t_f - t)$ is sketched in the left hand side of Figure 3.8. In this case, workstation A starts processing the

jobs earlier at maximal rate than workstation B. To minimize the weighted amount of jobs in buffers 2 and 3, both workstations idle for a duration in the beginning of this mode. All buffer contents remain then constant and jobs arrive into buffer 1 with arrival rate λ. After this idle duration, workstation A processes the jobs for step 1 at maximal rate μ_1 to make sure that buffer 2 contains an amount of x_2^f at the end of this mode. Workstation B continues idling until it can process an amount of $x_3^f - x_3(t)$ jobs at maximal rate μ_2 in $(t_f - t)$ time-units. With this sequence of process rates and given the boundary wip levels, the weighted amount of jobs in buffers 2 and 3 are as low as possible which results in an optimal behavior for this system mode.

System mode (1,3)

In Figure 3.9, the specific reentrant system operates in mode $(m_A, m_B) = (1, 3)$. Workstation A processes the jobs for step 1 and workstation B processes the jobs for step 3. In this case, the arrival rate into buffer 3 is zero and the arrival rates into buffers 2 and 4 depend on the process rate at the workstations. Whereas the arrival rate into buffer 1 is always equal to $\lambda = 1$ $[\frac{\text{job}}{\text{time-unit}}]$.

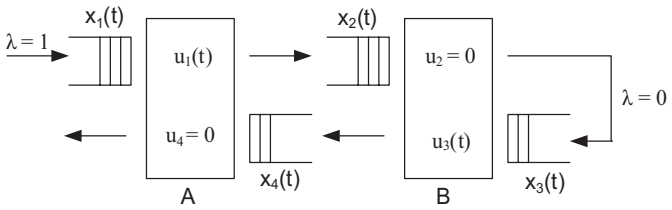


Figure 3.9: System operates in mode (1,3).

In this mode, both workstations can process the jobs independently of each other. Therefore, to minimize the weighted wip level function J (3.6) for this mode, the optimization problem can be split into an optimization problem for each workstation. First consider workstation A, for which the following optimization problem should be solved for given t_0, t_f, x_1^0, x_2^0, x_1^f and x_2^f.

$$\begin{aligned}
&\min_{u_1(t)} \; c_1 \int_{t_0}^{t_f} (x_1(\tau) + x_2(\tau)) \; d\tau + (c_2 - c_1) \int_{t_0}^{t_f} x_2(\tau) \; d\tau, \\
&\qquad \text{with } c_1 < c_2. \\
&\text{s.t. } x_1(t_0) = x_1^0, \; x_1(t_f) = x_1^f, \; 0 \le u_1(t) \le \mu_1, \; x_1(t) \ge 0, \\
&\qquad x_2(t_0) = x_2^0, \; x_2(t_f) = x_2^f, \; u_2(t) = 0, \; x_2(t) \ge 0. \\
&\text{where } (\lambda - \mu_1)(t_f - t_0) \le x_1^f - x_1^0 \le \lambda(t_f - t_0), \\
&\qquad\qquad x_1^0 + x_2^0 + \lambda(t_f - t_0) = x_1^f + x_2^f.
\end{aligned}$$

During mode $(m_A, m_B) = (1, 3)$ which has a duration of $(t_f - t_0)$, the process rate $u_1(t)$ should be determined subject to the constraints to minimize the weighted number of jobs in buffers 1 and 2, i.e. $\min \int_{t_0}^{t_f} c_1 x_1(\tau) + c_2 x_2(\tau) \; d\tau$. As already mentioned in mode $(m_A, m_B) = (1, 2)$, the buffer contents of buffer 1 always increases with rate λ during a cycle period. Therefore, this buffer contents can never increase faster than with this rate, i.e. $x_1^f - x_1^0 \le \lambda(t_f - t_0)$. Also,

this buffer contents can never decrease faster than the case when workstation A processes the jobs at maximal rate, i.e. $x_1^f - x_1^0 \geq (\lambda - \mu_1)(t_f - t_0)$. Besides this, the total number of jobs in buffer 1 and 2 can never decrease during this mode, no jobs are leaving buffer 2. This total amount of jobs only increases with rate λ, i.e. $x_1^0 + x_2^0 + \lambda(t_f - t_0) = x_1^f + x_2^f$.

Lemma 3.5. *The optimal behavior for which the weighted wip level is minimal for workstation A which operates in mode $(m_A, m_B) = (1, 3)$ is achieved when workstation A processes the jobs in the following sequence.*

$$\begin{aligned} u_1(t) &= 0 \quad \text{if} \quad x_2^f - x_2(t) < \mu_1(t_f - t); \\ u_1(t) &= \mu_1 \quad \text{if} \quad x_2^f - x_2(t) = \mu_1(t_f - t). \end{aligned}$$

Given the boundary wip levels, this optimal behavior results in the smallest weighted wip level for this system mode.

Proof. In system mode $(m_A, m_B) = (1, 3)$, no jobs are leaving buffer 2. Therefore, the total amount of jobs in buffers 1 and 2 will never decrease. With an arrival rate of λ into buffer 1, the total amount of jobs in buffers 1 and 2 is increasing with this rate. The sum of the buffer contents of buffers 1 and 2 is known and fixed and the first part of the optimization function which contains the sum of x_1 and x_2 can not be minimized. But the second part which contains the buffer contents of buffer 2 can be minimized. Consider now the case when the sequence of process rates in Lemma 3.5 is used. This result should lead to optimal behavior for this mode for which the weighted amount of jobs in buffer 2 is as low as possible. In the beginning of this mode, workstation A idles for a duration until the constraint $x_2^f - x_2(t) = \mu_1(t_f - t)$ is satisfied. Then, workstation A processes the jobs for step 1 at maximal rate μ_1 until buffer 2 contains an amount of x_2^f jobs. In this case, the buffer contents of buffer 2 is as low as possible in the beginning of this mode and at the end buffer 2 can never increase faster than in this case. Therefore, the weighted amount of jobs in buffer 2 is minimal and lead to optimal behavior in this mode. This optimal behavior is sketched in Figure 3.10.

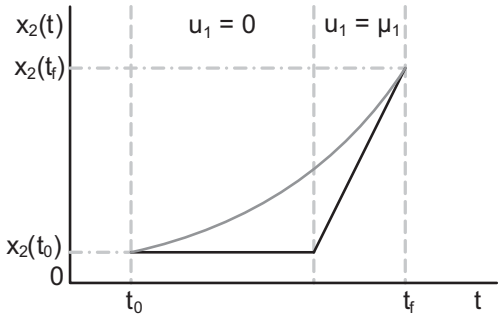


Figure 3.10: Optimal buffer behavior of buffer 2 in mode (1,3).

The same analysis holds for workstation B for which the following optimization problem

should be solved for given t_0, t_f, x_3^0, x_4^0, x_3^f and x_4^f.

$$\begin{aligned}
&\min_{u_3(t)} c_3 \int_{t_0}^{t_f} (x_3(\tau) + x_4(\tau))\ d\tau + (c_4 - c_3) \int_{t_0}^{t_f} x_4(\tau)\ d\tau, \\
&\quad \text{with } c_3 < c_4. \\
&\text{s.t. } x_3(t_0) = x_3^0,\ x_3(t_f) = x_3^f,\ 0 \le u_3(t) \le \mu_3,\ x_3(t) \ge 0, \\
&\quad x_4(t_0) = x_4^0,\ x_4(t_f) = x_4^f,\ u_4(t) = 0,\ x_4(t) \ge 0. \\
&\text{where } 0 \le x_3^0 - x_3^f \le \mu_3(t_f - t_0), \\
&\qquad x_3^0 + x_4^0 = x_3^f + x_4^f.
\end{aligned}$$

Lemma 3.6. *The optimal behavior for which the weighted wip level is minimal for workstation B which operates in mode $(m_A, m_B) = (1, 3)$ is achieved when workstation B processes the jobs in the following sequence.*

$$\begin{aligned}
u_3(t) = 0 \quad &\text{if} \quad x_4^f - x_4(t) < \mu_3(t_f - t); \\
u_3(t) = \mu_3 \quad &\text{if} \quad x_4^f - x_4(t) = \mu_3(t_f - t).
\end{aligned}$$

Given the boundary wip levels, this optimal behavior results in the smallest weighted wip level for this system mode.

Proof. In this system mode, the total amount of jobs in buffers 3 and 4 is constant. No jobs arrive into buffer 3 and no jobs are leaving buffer 4 during this system mode. The sum of the buffer contents of buffers 3 and 4 is constant and the first part of the optimization function can not be minimized. But the second part, which contains the buffer contents of buffer 4 can be minimized. Consider now the case when the sequence of process rates in Lemma 3.6 is used. This result should lead to optimal behavior for this mode for which the weighted amount of jobs in buffer 4 is as low as possible. First, workstation B idles for a duration until the constraint $x_4^f - x_4(t) = \mu_3(t_f - t)$ is satisfied. Then, workstation B processes the jobs for step 3 at maximal rate μ_3. In this case, the buffer contents of buffer 4 is as low as possible in the beginning of this mode and at the end buffer 4 can never increase faster than in this case. Therefore, the weighted amount of jobs in buffer 4 is minimal and lead to optimal behavior in this mode. This optimal behavior is sketched in Figure 3.11.

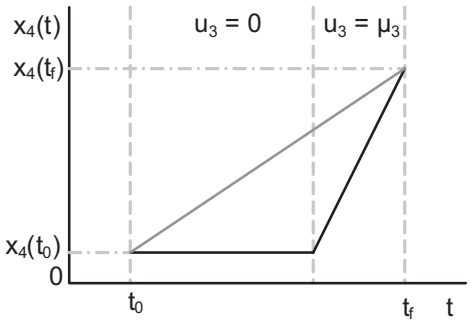


Figure 3.11: Optimal buffer behavior of buffer 4 in mode (1,3).

The optimal sequence of process rates is determined for each workstation separately in this system mode. But with the results in Lemma 3.5 and 3.6 in mind, the optimal sequence of

process rates for this system, see Figure 3.12, for which the weighted wip level is minimal is achieved when both workstations process the jobs in the following sequence of process rates depending on the wip levels in the beginning and at the end of this system mode.

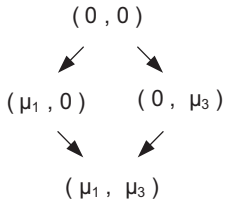


Figure 3.12: Optimal sequence of process rates for system mode (1,3).

3.4 Duration of every action in both workstations

In previous section, the optimal sequence of process rates in each system mode for which the weighted wip level is as low as possible is determined. This optimal sequence of process rates for each system mode depends on the boundary wip levels and it is explained for each system mode independently of each other and in a random order. Before good desired system behavior can be determined, the duration of every action in a system mode is determined first. When these durations are known, the system modes can be scheduled into one schedule. In Table 3.1, the optimal sequence of process rates for each workstation in a system mode is summarized which is explained in the previous section.

Table 3.1: Optimal sequence of process rates for each system mode.

System mode	Workstation A	Workstation B
$(m_A, m_B) = (4, 3)$	μ_4 0	0 μ_4* μ_3
$(m_A, m_B) = (4, 2)$	μ_4 0	0 μ_2
$(m_A, m_B) = (1, 2)$	0 μ_2** μ_1	0 μ_2
$(m_A, m_B) = (1, 3)$	0 μ_1	0 μ_3

* *This only occurs when workstation A processes the jobs for step 4 at* μ_4 *and* $x_4 = 0$.
** *This only occurs when workstation B processes the jobs for step 2 at* μ_2 *and* $x_2 = 0$.

Workstation A

Before the periodic cycle behavior of workstation A is determined, the duration of every activity is calculated. Workstation A can process the jobs for step 1 at 0, μ_2 and μ_1 and processes the jobs for step 4 at 0 and μ_4. Also, only the jobs can be processed at a workstation for which the workstation has been set up. Therefore, if workstation A should process the

jobs for another step a setup is required. In Table 3.2, these activities and the corresponding setup times and process rates are presented. The sum of all these durations is equal to the cycle period T, which is presented in the first column of the table.

Table 3.2: Duration of every activity for workstation A.

duration	setup/process rate
τ_0^A	$u_1 = u_4 = 0$
τ_2^A	$u_1 = \mu_2 = \frac{1}{0.6}$
τ_1^A	$u_1 = \mu_1 = \frac{1}{0.3}$
τ_4^A	$u_4 = \mu_4 = \frac{1}{0.6}$
σ_{41}	50
σ_{14}	50
T	+

Only the durations of the setup times σ_{41} and σ_{14} are known. But with a constant arrival rate λ of 1 $\left[\frac{\text{job}}{\text{time-unit}}\right]$ into buffer 1, during a cycle period of T time-units, both step 1 and step 4 need to process T jobs to create a stable system. For step 4, this means that the released amount of jobs into buffer 1 should also be processed for step 4 during a cycle period. This can be denoted by the following equation:

$$\mu_4 \ \tau_4^A = \lambda T. \tag{3.7}$$

Similarly for step 1 where the jobs can be processed at two different rates, workstation A can process an amount of $\mu_1 \ \tau_1^A$ jobs in duration τ_1^A. In duration τ_2^A workstation A can process an amount of $\mu_2 \ \tau_2^A$ jobs. In a cycle period of T time-units, workstation A has to process T jobs for step 1. This can be denoted by the following equation:

$$\mu_1 \ \tau_1^A + \mu_2 \ \tau_2^A = \lambda T. \tag{3.8}$$

Besides these 2 equations, from the first column in Table 3.2 the following equation can be derived which gives the cycle period T as a function of all theses durations.

$$T = \tau_0^A \ + \ \tau_1^A \ + \ \tau_2^A \ + \ \tau_4^A \ + \ \sigma_{41} \ + \ \sigma_{14}. \tag{3.9}$$

With these equations, the durations of every step for workstation A can be determined as a function of the cycle period T and the idling duration τ_0^A:

$$\tau_1^A = \frac{1}{5}T + \tau_0^A + 100, \ \text{time-units}, \tag{3.10}$$

$$\tau_2^A = \frac{1}{5}T - 2\tau_0^A - 200, \ \text{time-units}, \tag{3.11}$$

$$\tau_4^A = \frac{3}{5}T \ \text{time-units}. \tag{3.12}$$

For any given cycle period T and idling duration τ_0^A, the duration of every activity can easily be determined for workstation A.

Workstation B

The same analysis holds for workstation B. Workstation B can process the jobs for step 2 at 0 and μ_2 and processes the jobs for step 3 at 0, μ_4 and μ_3. Also, only the jobs can be processed at this workstation for which the workstation has been set up. Therefore, if workstation B should process the jobs for another step a setup is required. In Table 3.2, these activities and the corresponding setup times and process rates are presented. The sum of all these durations is equal to the cycle period T, which is presented in the first column of the table.

Table 3.3: Duration of every activity for workstation B.

duration	setup/process rate
τ_0^B	$u_2 = u_3 = 0$
τ_2^B	$u_2 = \mu_2 = \frac{1}{0.6}$
τ_3^B	$u_3 = \mu_3 = \frac{1}{0.3}$
τ_4^B	$u_3 = \mu_4 = \frac{1}{0.6}$
σ_{23}	50
σ_{32}	50
T	+

Only the durations of the setup times σ_{23} and σ_{32} are known. But during a cycle period of T time-units, both step 2 and step 3 need to process T jobs to create a stable system. For step 2, this can be denoted by the following equation:

$$\mu_2 \ \tau_2^B = \lambda T. \tag{3.13}$$

Similarly for step 3, workstation B can process an amount of $\mu_3 \ \tau_3^B$ jobs in a duration of τ_3^B time-units. Also, in a duration of τ_4^B time-units, workstation B can process an amount of $\mu_4 \ \tau_4^B$ jobs. In a cycle period of T time-units, workstation B has to process T jobs for step 3. This can be denoted by the following equation:

$$\mu_3 \ \tau_3^B + \mu_4 \ \tau_4^B = \lambda T. \tag{3.14}$$

Besides these 2 equations, from the first column in Table 3.3 the following equation can be derived which gives the cycle period T as a function of all these durations:

$$T = \tau_0^B \ + \ \tau_2^B \ + \ \tau_3^B \ + \ \tau_4^B \ + \ \sigma_{23} \ + \ \sigma_{32}. \tag{3.15}$$

With these equations, the durations of every step for workstation B can be determined as a function of the cycle period T and the idling duration τ_0^B:

$$\tau_2^B = \frac{3}{5}T \text{ time-units}, \tag{3.16}$$

$$\tau_3^B = \frac{1}{5}T + \tau_0^B + 100 \text{ time-units}, \tag{3.17}$$

$$\tau_4^B = \frac{1}{5}T - 2\tau_0^B - 200 \text{ time-units}. \tag{3.18}$$

For any given cycle period $T \geq 1000$ time-units and idle durations $\tau_0^A \geq 0$ and $\tau_0^B \geq 0$, the duration of every activity in both workstations can be determined. But with these constraints it is possible that the durations τ_2^A and τ_4^B are negative while the constraints are satisfied. For example, if the cycle period T is equal to 1000 time-units, any given positive idle duration results in a negative duration for τ_2^A and τ_4^B. Therefore, the constraints have to be improved to neglect this problem of negative durations. First consider the duration $\tau_2^A \geq 0$ and (3.11). After rewriting this equation, a new constraint can be made which will never result in negative durations of the activities in both workstations if this new constraint is satisfied. This new constraint becomes:

$$\tau_2^A = \frac{1}{5}T - 2\tau_0^A - 200 \geq 0 \quad \rightarrow \quad \tau_0^A \leq \frac{1}{10}T - 100. \tag{3.19}$$

With this result and the requirement that the idle duration can never be negative, the new constraint for the idle duration τ_0^A becomes:

$$0 \leq \tau_0^A \leq \frac{1}{10}T - 100. \tag{3.20}$$

This analysis holds also for the idle duration τ_0^B which becomes after rewriting (3.18) and with the requirement $\tau_4^B \geq 0$:

$$0 \leq \tau_0^B \leq \frac{1}{10}T - 100. \tag{3.21}$$

When these adaptations are made, for any given cycle period T and idle durations τ_0^A and τ_0^B — which satisfy the constraints $T \geq 1000$, $0 \leq \tau_0^A \leq \frac{1}{10}T - 100$ and $0 \leq \tau_0^B \leq \frac{1}{10}T - 100$ — the durations of every activity in both workstations can be determined which will probably lead to an optimal system behavior.

3.5 Minimal buffer contents

With the duration of every activity in both workstations and the optimal sequence of process rates for each system mode, the mean number of jobs in each buffer during a cycle period can be determined. When these buffer contents are known, the weighted wip level function J can be calculated for any given cycle period T and idle durations τ_0^A and τ_0^B.

Lemma 3.7. *During a cycle period of T time-units, each buffer contains at least the following mean amount of jobs:*

$$\frac{1}{T}\int_0^T x_1(t)\ dt \geq 350,$$

$$\frac{1}{T}\int_0^T x_2(t)\ dt \geq 150,$$

$$\frac{1}{T}\int_0^T x_3(t)\ dt \geq 500,$$

$$\frac{1}{T}\int_0^T x_4(t)\ dt \geq 150.$$

In this section, the mean amount of jobs in each buffer is explained.

Workstation A

The buffer contents of buffers 1 and 4 are sketched in Figure 3.13. During the first 50 time-units, workstation A is setting up for processing the jobs for step 4. After this setup has been completed, workstation A starts processing the jobs for step 4 at maximal rate μ_4. From (3.12), it is known that production step 4 takes $\frac{3}{5}T$ time-units. For buffer 4, this duration is sketched in more detail in Figure 3.15. When workstation A has processed all the jobs for step 4, workstation A idles for a duration of τ_0^A time-units. Then, workstation A is setting up for processing the jobs for step 1 which takes again 50 time-units. During this setup and during the idling duration, workstation A can not process jobs, therefore it does not matter if workstation A idles first for a duration and then setting up for step 3 or first setting up and then idling for a duration. After this setup has been completed, workstation A processes the jobs for step 1 at maximal rate μ_1 and eventually at process rate μ_2.

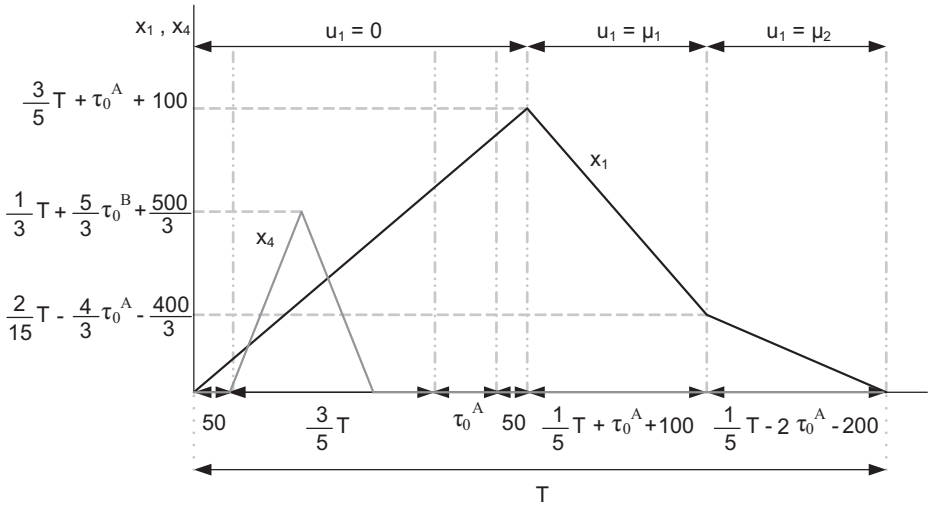


Figure 3.13: Buffer contents of buffers 1 and 4.

First consider the buffer contents of buffer 1 and the case where $T \geq 1000$ time-units and $0 \leq \tau_0^A \leq \frac{1}{10}T - 100$. From previous section, it is known that workstation A can process the jobs for step 1 at process rates $0, \mu_2$ and μ_1. The duration of every action for workstation A is also known. Due to the fixed arrival rate λ of 1 $\left[\frac{\text{job}}{\text{time-unit}}\right]$ into buffer 1 in this case, the buffer contents of buffer 1 always increases with this rate during a cycle period. During a setup, an idle duration or when workstation A processes the jobs for step 4, which take $100 + \tau_0^A + \frac{3}{5}T$ time-units in total, the buffer contents of buffer 1 increases with a rate equal to λ to an amount of $100 + \tau_0^A + \frac{3}{5}T$ jobs. The buffer contents of buffer 1 only decreases when workstation A processes the jobs for step 1 at rate μ_2 or μ_1. To minimize the weighted amount of jobs in buffer 1, workstation A processes first the jobs at maximal rate μ_1 and then eventually at process rate μ_2. Workstation A processes for a duration of $\frac{1}{5}T + \tau_0^A + 100$ the jobs for step 1 at maximal rate μ_1. During this action, jobs are released into this buffer with rate λ and jobs are leaving this buffer with a rate equal to the maximal process rate. After this action, buffer 1 contains an amount of $\left[\left(\frac{3}{5}T + \tau_0^A + 100\right) - \left(\frac{1}{5}T + \tau_0^A + 100\right)(\mu_1 - \lambda) =\right] \frac{2}{15}T - \frac{4}{3}\tau_0^A - \frac{400}{3}$ jobs and workstation A starts processing the jobs for step 1 eventually at process rate μ_2 for a duration of $\frac{1}{5}T - 2\tau_0^A - 200$ time-units. This action only occurs when buffer 2 is empty and

workstation B processes the jobs for step 2 at maximal process rate μ_2. After this action, buffer 1 contains an amount of $\left[\left(\frac{2}{15}T - \frac{4}{3}\tau_0^A - \frac{400}{3}\right) - \left(\frac{1}{5}T - 2\tau_0^A - 200\right)(\mu_2 - \lambda) =\right] 0$ jobs. The buffer contents of buffer 1 is sketched in Figure 3.13 as a function of the cycle period T and the idle duration τ_0^A.

The average number of jobs in buffer 1 during a cycle period of T time-units can be determined. This is equal to the area under the graph of buffer 1 in Figure 3.13 divided by the cycle period T, which becomes:

$$\frac{1}{T}\int_0^T x_1(t)\,dt = \frac{4}{15}T + \frac{200}{3} + \frac{2}{3}\tau_0^A + \frac{50000}{3T} + \frac{1000}{3T}\tau_0^A + \frac{5}{3T}\left(\tau_0^A\right)^2. \tag{3.22}$$

In Figure 3.14, the solutions of (3.22) are plotted for several values of T and τ_0^A. Only the solutions which lies in area $\mathcal{F}$ are feasible solutions. These solutions satisfy the constraints $T \geq 1000$ and $0 \leq \tau_0^A \leq \frac{1}{10}T - 100$ which are the boundaries of the feasible area $\mathcal{F}$.

PSfrag replacements
τ_0^A
$\tau_0^A = \frac{1}{10}T - 100$
$\mathcal{F}$

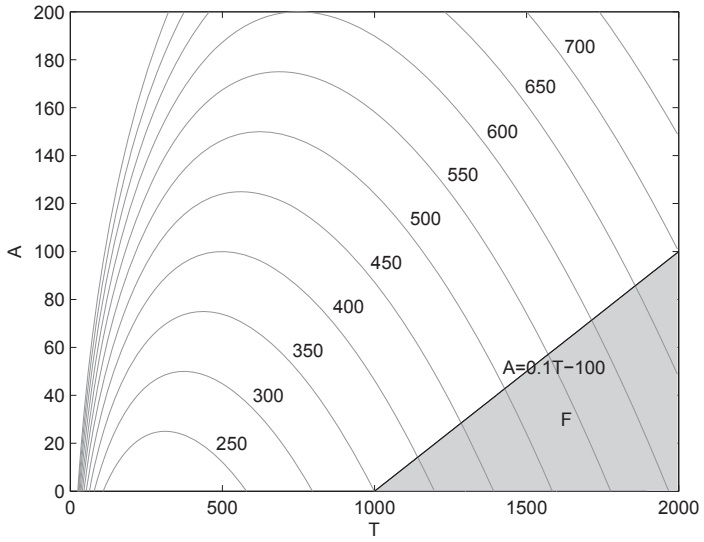


Figure 3.14: Minimal average amount of jobs in buffer 1.

With Figure 3.14, we see that buffer 1 contains at least an average amount of 350 jobs during a cycle period of 1000 time-units without an idle duration ($\tau_0^A = 0$).

The same analysis can be done for the buffer contents of buffer 4. This buffer contents is also sketched in Figure 3.13. During setups, idle durations and when workstation A processes the jobs for step 1, the buffer contents of buffer 4 remains empty.

Jobs arrive into buffer 4 with a rate equal to the process rate for step 3 at workstation B and leave this buffer with a rate equal to the process rate for step 4 at workstation A. From (3.12), it is known that production step 4 takes $\frac{3}{5}T$ time-units. For buffer 4, this activity is sketched in more detail in Figure 3.15. Workstation B processes the jobs during $\frac{1}{5}T + \tau_0^B + 100$ time-units at maximal rate μ_3. Due to the fact that the process rate μ_3 is two times larger than the process rate μ_4, the buffer contents of buffer 4 increases to

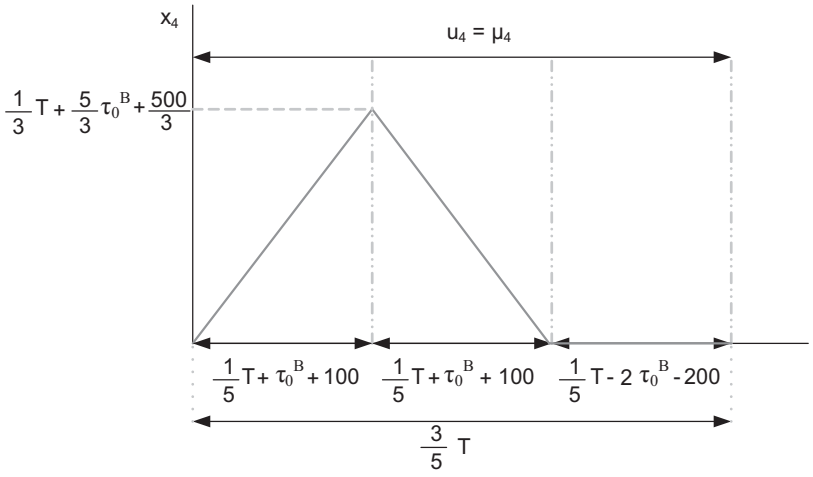


Figure 3.15: Buffer contents of buffer 4.

$\left[\left(\frac{1}{5}T + \tau_0^B + 100\right)(\mu_3 - \mu_4) =\right] \frac{1}{3}T + \frac{5}{3}\tau_0^B + \frac{500}{3}$ jobs. When workstation B has processed the jobs for step 3 at maximal rate μ_3, workstation A needs again $\frac{1}{5}T + \tau_0^B + 100$ time-units to process the $\frac{1}{3}T + \frac{5}{3}\tau_0^B + \frac{500}{3}$ jobs. After this action, buffer 4 becomes empty. As mentioned in previous section, workstation B can process the jobs for step 3 at process rate μ_4 which takes $\frac{1}{5}T - 2\tau_0^B - 200$ time-units. This action only occurs when buffer 4 is empty and workstation A processes the jobs for step 4. The buffer contents of buffer 4 remains empty during this action. All the jobs that are processed at workstation B for step 3 can immediately processed at workstation A for step 4 before leaving the system.

The average number of jobs in buffer 4 during a cycle period of T time-units can be determined. This is equal to the area under the graph in Figure 3.15 divided by the cycle period T, which becomes:

$$\frac{1}{T}\int_0^T x_4(t)\,dt = \frac{1}{15}T + \frac{200}{3} + \frac{2}{3}\tau_0^B + \frac{50000}{3T} + \frac{1000}{3T}\tau_0^B + \frac{5}{3T}\left(\tau_0^B\right)^2. \tag{3.23}$$

In Figure 3.16, the solutions of (3.23) are plotted for several values of T and τ_0^B. Only the solutions which lies in area $\mathcal{F}$ are feasible solutions. Solutions in this area satisfy the constraints $T \geq 1000$ and $0 \leq \tau_0^B \leq \frac{1}{10}T - 100$.

The minimal average number of jobs in buffer 4 is achieved when the cycle period T is equal to 1000 time-units. The idle duration τ_0^B is then equal to 0. In this case, buffer 4 contains at least an average amount of 150 jobs.

Workstation B

The buffer contents of buffers 2 and 3 are sketched in Figure 3.17. During the first 50 time-units, workstation B is setting up for processing the jobs for step 2. After this setup has been completed, workstation B idles for a duration before it starts processing the jobs for step 2 at maximal rate μ_2. From (3.16), it is known that production step 2 takes $\frac{3}{5}T$ time-units. For buffer 2, this duration is sketched in more detail in Figure 3.19. When workstation B has

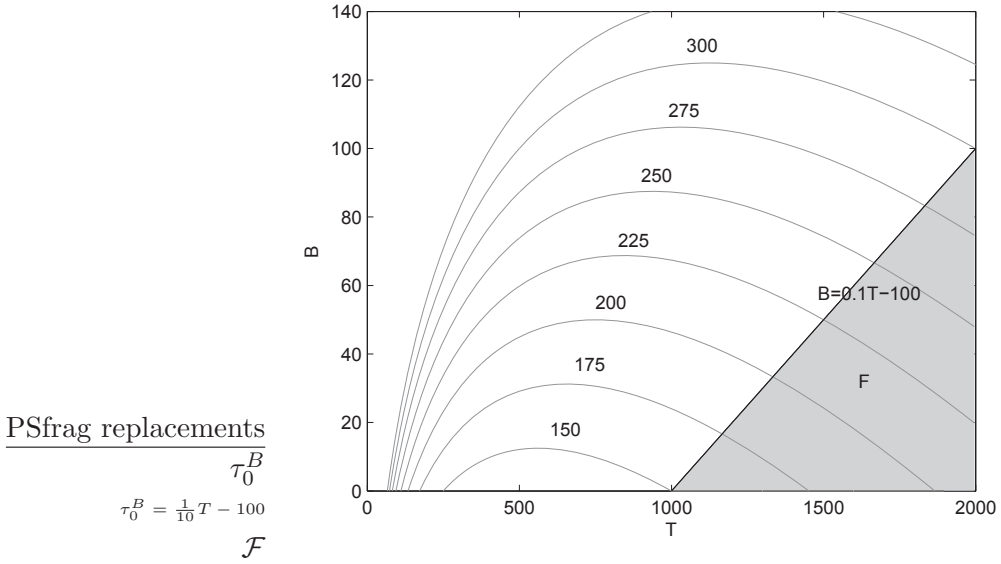


Figure 3.16: Minimal averaged amount of jobs in buffer 4.

processed all the jobs for step 2, a setup for processing the jobs for step 3 is needed which takes again 50 time-units. After this setup has been completed, workstation B processes the jobs for step 3 at maximal rate μ_3 and eventually at process rate μ_4.

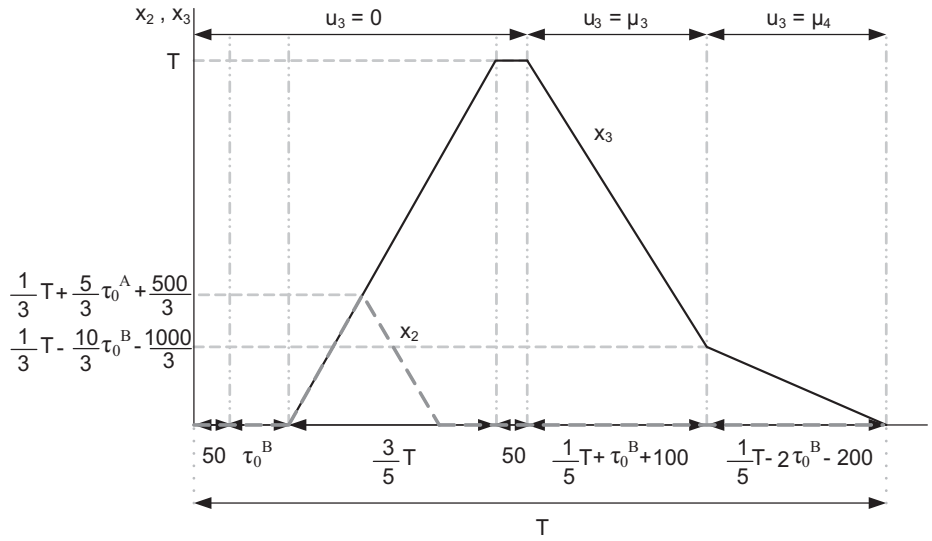


Figure 3.17: Buffer contents of buffers 2 and 3.

Consider the buffer contents of buffer 3. Jobs arrive into buffer 3 with a rate equal to the process rate for step 2 at workstation B and leave this buffer with a rate equal to the process rate for step 3 at workstation B. During the first setup and idle duration, buffer 3 remains empty. During the next $\frac{3}{5}T$ time-units, workstation B processes the jobs for step 2 at maximal rate μ_2. Buffer 3 increases with this rate to an amount of $[\left(\frac{3}{5}T\right)\mu_2 =]T$ jobs. Then, this buffer contents remains at T for a duration of 50 time-units due to the setup from

jobs for step 2 to jobs for step 3. After this setup has been completed, workstation B can process the jobs of buffer 3 at maximal rate μ_3 and eventually at rate μ_4. To decrease the weighted wip level of buffer 3, it is more optimal to start processing at maximal rate μ_3 and then eventually at rate μ_4. Workstation B processes for a duration of $\frac{1}{5}T + \tau_0^B + 100$ the jobs for step 3 at maximal rate μ_3. During this activity, buffer 3 decreases to an amount of $\left[T - \left(\frac{1}{5}T + \tau_0^B + 100\right)\mu_3 =\right] \frac{1}{3}T - \frac{10}{3}\tau_0^B - \frac{1000}{3}$ jobs. When this activity is completed, workstation B processes the jobs for step 3 eventually at process rate μ_4 for a duration of $\frac{1}{5}T - 2\tau_0^B - 200$ time-units. This action only occurs when buffer 4 is empty and workstation A processes the jobs for step 4 at maximal process rate μ_4. After this action buffer 3 contains an amount of $\left[\left(\frac{1}{3}T - \frac{10}{3}\tau_0^B - \frac{1000}{3}\right) - \left(\frac{1}{5}T - 2\tau_0^B - 200\right)\mu_4 =\right] 0$ jobs. The buffer contents of buffer 3 is sketched in Figure 3.17 as a function of the cycle period T and the idle duration τ_0^B.

The average number of jobs in buffer 3 during a cycle period of T time-units can be determined. This is equal to the area under the graph in Figure 3.17 divided by the cycle period T, which becomes:

$$\frac{1}{T}\int_0^T x_3(t)\,dt = \frac{7}{15}T + \frac{50}{3} - \frac{1}{3}\tau_0^B + \frac{50000}{3T} + \frac{1000}{3T}\tau_0^B + \frac{5}{3T}\left(\tau_0^B\right)^2. \tag{3.24}$$

In Figure 3.18, the solutions of (3.24) are plotted for several values of T and τ_0^B. Only the solutions which lies in area $\mathcal{F}$ are feasible solutions. These solutions satisfy the constraints $T \geq 1000$ and $0 \leq \tau_0^B \leq \frac{1}{10}T - 100$ which are the boundaries of the feasible area $\mathcal{F}$.

PSfrag replacements
τ_0^B
$\tau_0^B = \frac{1}{10}T - 100$
$\mathcal{F}$

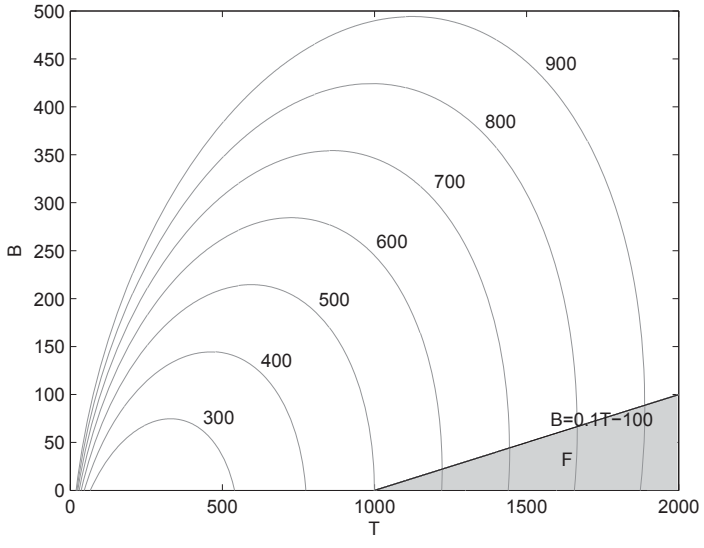


Figure 3.18: Minimal amount of jobs in buffer 3.

With the constraints $T \geq 1000$ and $0 \leq \tau_0^B \leq \frac{1}{10}T - 100$, the minimal average number of jobs in buffer 3 is achieved when the cycle period T is equal to 1000 time-units. The idle duration τ_0^B is then equal to 0. In this case, buffer 3 contains at least an average amount of 500 jobs during a cycle period of 1000 time-units.

The same analysis can be done for the buffer contents of buffer 2. This buffer contents is also sketched in Figure 3.17. During setups, idle durations and when workstation B processes the jobs for step 3, the buffer contents of buffer 2 remains empty.

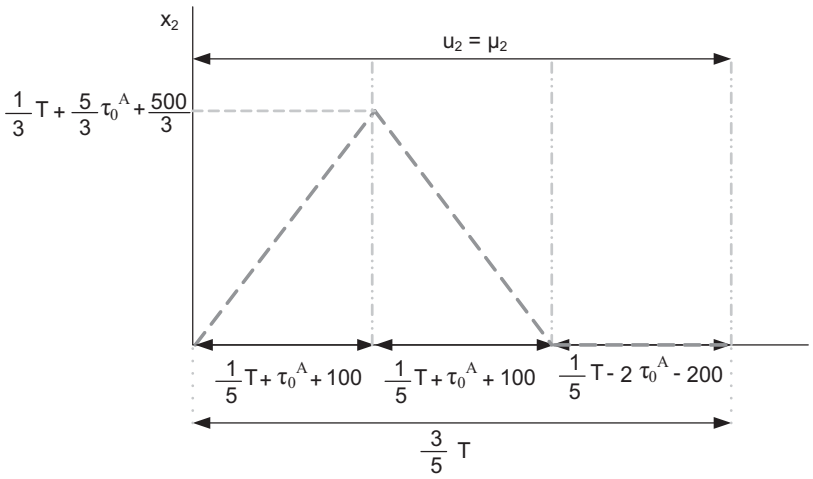


Figure 3.19: Buffer contents of buffer 2.

Jobs arrive into buffer 2 with a rate equal to the process rate for step 1 at workstation A and leave this buffer with a rate equal to the process rate for step 2 at workstation B. From (3.16), it is known that production step 2 takes $\frac{3}{5}T$ time-units. For buffer 2, this activity is sketched in more detail in Figure 3.19. Workstation A processes the jobs during $\frac{1}{5}T + \tau_0^A + 100$ time-units at maximal rate μ_1. Due to the fact that the process rate μ_1 is two times larger than the process rate μ_2, the buffer contents of buffer 2 increases to $\left[\left(\frac{1}{5}T + \tau_0^A + 100\right)(\mu_1 - \mu_2) =\right] \frac{1}{3}T + \frac{5}{3}\tau_0^A + \frac{500}{3}$ jobs. When workstation A has processed the jobs for step 1 at maximal rate μ_1, workstation B needs again $\frac{1}{5}T + \tau_0^A + 100$ time-units to process the $\frac{1}{3}T + \frac{5}{3}\tau_0^A + \frac{500}{3}$ jobs. After this action, buffer 2 is empty. As mentioned in previous section, workstation A can process the jobs for step 1 at process rate μ_2 which takes $\frac{1}{5}T - 2\tau_0^A - 200$ time-units. This action only occurs when buffer 2 is empty and workstation B processes the jobs for step 2. The buffer contents of buffer 2 remains empty during this action. All the jobs that are processed at workstation A for step 1 can immediately processed at workstation B for step 2 before leaving the system.

The average number of jobs in buffer 2 during a cycle period of T time-units can be determined. This is equal to the area under the graph in Figure 3.19 divided by the cycle period T, which becomes:

$$\frac{1}{T}\int_0^T x_2(t)\,dt = \frac{1}{15}T + \frac{200}{3} + \frac{2}{3}\tau_0^A + \frac{50000}{3T} + \frac{1000}{3T}\tau_0^A + \frac{5}{3T}\left(\tau_0^A\right)^2. \tag{3.25}$$

In Figure 3.20, the solutions of (3.25) are plotted for several values of T and τ_0^A. Only the solutions which lies in area $\mathcal{F}$ are feasible solutions. These solutions satisfy the constraints $T \geq 1000$ and $0 \leq \tau_0^A \leq \frac{1}{10}T - 100$ which are the boundaries of the feasible area $\mathcal{F}$.

With the constraints $T \geq 1000$ and $0 \leq \tau_0^A \leq \frac{1}{10}T - 100$, the minimal average number of jobs in buffer 2 is achieved when the cycle period T is equal to 1000 time-units. The idle duration

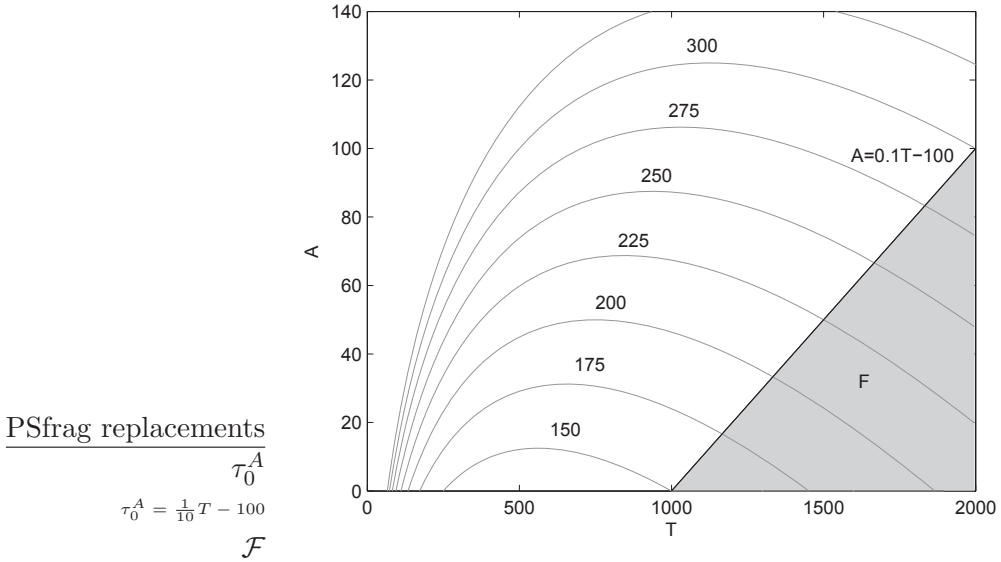


Figure 3.20: Minimal amount of jobs in buffer 2.

τ_0^A is then equal to 0. In this case, buffer 2 contains at least an average amount of 150 jobs during a cycle period of 1000 time-units.

3.6 Desired periodic system behavior

In previous section, the desired periodic behavior of each workstation is determined for which the mean amount of jobs is as low as possible. This behavior is achieved when the cycle period T is 1000 time-units. The idle durations τ_0^A and τ_0^B are then equal to zero. This desired behavior is sketched for each workstation in Figure 3.21.

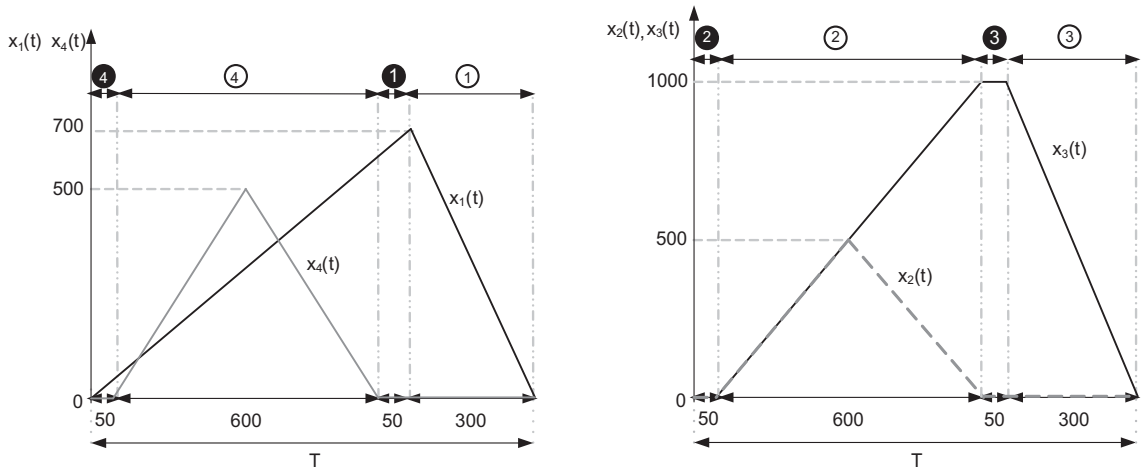


Figure 3.21: Desired behavior of each workstation.

In this section, the desired system behavior is determined by scheduling the behavior of both

workstations in Figure 3.21 into one schedule. As mentioned in previous section, this desired behavior is achieved when workstation A starts processing the jobs for step 4 and workstation B starts processing the jobs for step 3 at the same moment of time. In this case, the initial amount of jobs in the most expensive buffer is as low as possible. Also, when workstations A and B start processing the jobs for step 1 and step 2 at the same moment of time results in a system behavior for which the mean amount of jobs in the system is small.

When both requirements — steps 1 and 2 start at the same time and steps 3 and 4 start at the same time — should be fulfilled into one schedule, the system only operates in modes $(4, 3)$ and $(1, 2)$. But in a cycle period of $T = 1000$ time-units, the system can not only operate in modes $(4, 3)$ and $(1, 2)$. Therefore, both requirements can not be fulfilled in one schedule. Looking better at the durations of every production step in Figure 3.21, the largest durations are production steps 2 and 4 which have both a duration of $[\frac{3}{5}T =]600$ time-units. When these durations are scheduled in a cycle period of $T = 1000$ time-units, both durations always overlap each other for at least $\frac{1}{5}T$ time-units, which is sketched in Figure 3.22.

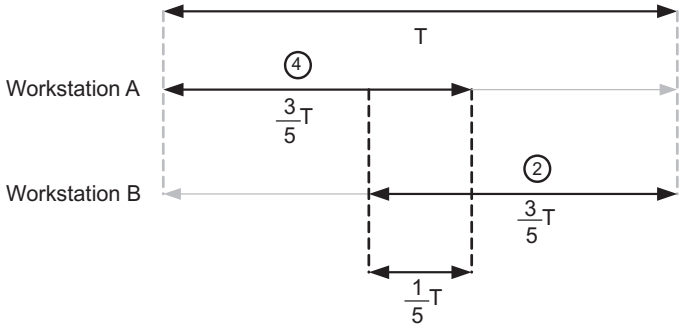


Figure 3.22: At least system mode (4,2) for $\frac{1}{5}T$ time-units.

This means that the system always operates for at least $\frac{1}{5}T$ time-units in system mode $(4, 2)$ and not only in mode $(4, 3)$ and mode $(1, 2)$. Both requirements — steps 1 and 2 start at the same time and steps 3 and 4 start at the same time — can not be fulfilled in one schedule. But a system behavior where both workstations process the jobs according to the desired behavior sketched in Figure 3.21 can be obtained. Depending on which requirement is fulfilled in a schedule, the initial amount of jobs in buffer 2 or buffer 4 should be increased. When workstation A processes first step 4, this case is sketched in Figure 3.22, buffer 2 should contain an initial amount of jobs. If the case is considered when workstation B processes first step 2, then buffer 4 should contain an initial amount of jobs. In both cases, the increased initial amount of jobs is the same. But to minimize the weighted wip level function J for this system, the case where buffer 2 contains an initial amount of jobs is considered due to the fact that buffer 4 is the most expensive buffer in this system. The complete schedule of this case is sketched in Figure 3.23. After the first 50 time-units, workstation A and B start processing the jobs for step 4 and step 3 at the same moment of time and the system fulfills a requirement. Workstation B completes its production step two times faster than workstation A. When workstation B has completed production step 3 and has been set up for processing the jobs for step 2, this workstation can not start processing because buffer 2 is empty. According to the behavior explained in previous section, workstation A should start processing the jobs for step 1 exactly at the time moment that buffer 2 is empty and workstation B is processing the jobs for step 2. But this is not possible because workstation A is still processing the jobs

for step 4. When both workstations have to process the jobs according to the behavior in Figure 3.21, this problem can be solved by increasing the initial amount of jobs in buffer 2. In that case, workstation B can start processing the jobs for step 2 while workstation A still processes the jobs for step 4. To minimize the mean amount of jobs in the system, the initial amount of jobs in buffer 2 should be as low as possible. From the start of processing the jobs for step 2 until the time moment that workstation A can start processing the jobs for step 1, buffer 2 should contain an initial amount of $\left[\left(\frac{3}{5}T - \left(\frac{2}{5}T - 100\right)\right)\mu_2 =\right] \frac{1}{3}T - \frac{500}{3}$ jobs. At the time moment that workstation B has processed this initial amount of jobs, workstation A starts processing the jobs for step 1 and workstation B can continue processing the jobs for step 2. This desired system behavior is sketched in Figure 3.24.

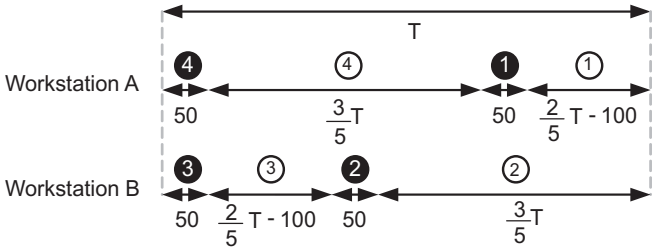


Figure 3.23: Scheduling the system modes.

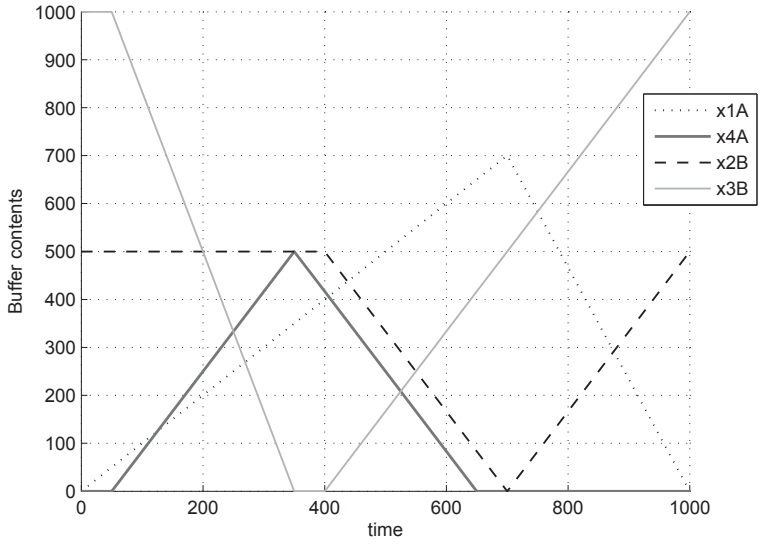


Figure 3.24: Desired periodic system behavior.

The weighted wip level function J becomes:

$$J = \frac{1}{T}\int_0^T 1x_1(t) + 2x_2(t) + 3x_3(t) + 4x_4(t)\ dt = 3150. \tag{3.26}$$

In next chapter, this desired system behavior is used as a starting point for the controller design. A non-distributed controller is designed which has to make the system converge from any initial condition to this desired periodic orbit. The way to derive this feedback controller is explained in next chapter.

Chapter 4

Non-distributed controller

In Chapter 2, the specific reentrant system with setup times has been explained in detail. In Chapter 3, the desired periodic system behavior is derived. In this chapter, a non-distributed controller is designed which makes this reentrant system converge towards this desired periodic behavior from any initial condition. As already mentioned in the Introduction, this chapter deals with the second research objective, which was formulated as follows:

Research objective 2:
Design a non-distributed controller (global policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

Before a non-distributed controller is designed, the desired closed-loop behavior of the system is determined first. The designed controller which is based on Lyapunov's direct method has to make the system converge from any initial condition to this desired periodic orbit. Based on the given desired periodic orbit, an "energy" of the system can be defined by considering the mean amount of work in the system [LR06b]. Depending on the initial condition, either one or more *translated* desired periodic orbits can be obtained which go through this initial condition. The translated desired periodic orbit with the smallest mean amount of work in the system which go through this initial condition is used as a starting point for the controller. The "energy" in the system of the translated desired periodic orbit is always higher than the "energy" in the system of the desired periodic orbit. By controlling the system in a way that this "energy" is never increasing, the system stabilizes at a fixed energy level and the controller has to make the system converge from the initial condition to the desired periodic orbit. In this chapter, the way to derive this feedback controller from the given desired periodic orbit is explained.

4.1 Desired periodic orbit of the system

The desired periodic system behavior, sketched in Figure 3.24, can also be depicted as a desired closed-loop behavior in a four dimensional space. In this space, the buffer contents of every buffer is plotted against each other which results in a desired periodic orbit of the specific system. However, a four dimensional space is difficult to show. Therefore, two views

of this desired periodic orbit, namely the desired periodic orbit of each workstation, are shown in Figure 4.1.

First, consider the desired periodic orbit of workstation A for which the buffer contents of buffer 1 is plotted against the buffer contents of buffer 4. The arrows in this figure give the direction in which the periodic orbit is passed through. Looking at the buffer contents of buffers 1 and 4 in Figure 3.24, this periodic orbit can easily be derived. In the first 50 time-units, the system is setting up for processing the jobs for step 4. During this setup, buffer 4 remains empty and buffer 1 increases with a rate $\lambda = 1 \left[\frac{\text{job}}{\text{time-unit}}\right]$ to 50 jobs. After this setup has been completed, i.e. at point $(x_1, x_4) = (50, 0)$, workstation A is processing the jobs for step 4 at maximal rate $\mu_4 = \frac{1}{0.6} \left[\frac{\text{job}}{\text{time-units}}\right]$. According to the behavior in Figure 3.24, buffer 4 increases first to 500 jobs and decreases then to 0 jobs. During this action, buffer 1 increases with rate $\lambda = 1 \left[\frac{\text{job}}{\text{time-unit}}\right]$ to 650 jobs. When buffer 4 becomes empty, i.e. at point $(x_1, x_4) = (650, 0)$, workstation A is ready with processing the jobs for step 4 and is setting up for processing the jobs for step 1. During this setup, buffer 4 remains empty and buffer 1 increases with rate $\lambda = 1 \left[\frac{\text{job}}{\text{time-unit}}\right]$ to 700 jobs. After this setup has been completed, i.e. at point $(x_1, x_4) = (700, 0)$, workstation A is processing the jobs for step 1 at maximal rate $\mu_1 = \frac{1}{0.3} \left[\frac{\text{job}}{\text{time-units}}\right]$. Buffer 1 decreases then to 0 jobs and buffer 4 remains empty. Workstation A completes then production step 1 and is again setting up for processing the jobs for step 4 which completes the cycle of this periodic orbit.

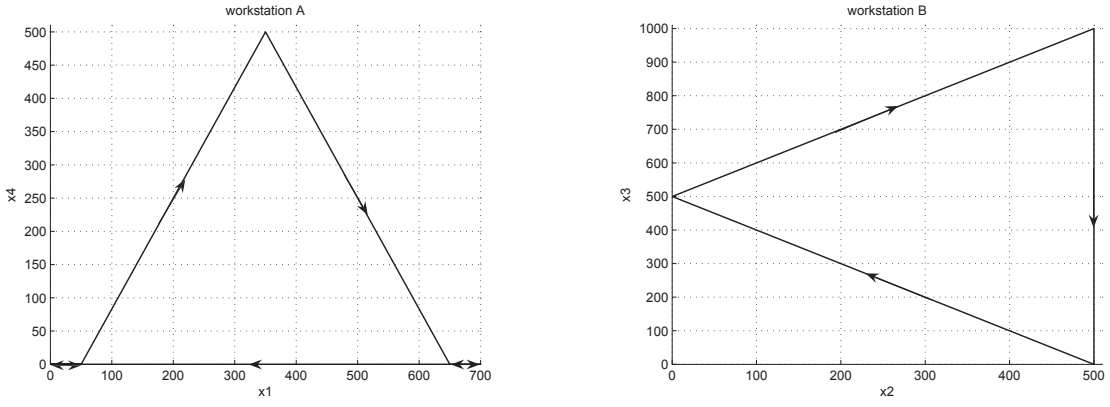


Figure 4.1: Desired periodic orbit of each workstation.

The desired periodic orbit of workstation B is presented in the right hand side of Figure 4.1. The buffer contents of buffer 2 is plotted against the buffer contents of buffer 3. This periodic orbit can also easily be derived from Figure 3.24. In the first 50 time-units, workstation B is setting up for processing the jobs for step 3. During this setup, the buffer contents of buffer 2 and buffer 3 remains constant, i.e. the system stays in point $(x_2, x_3) = (500, 1000)$. After this setup has been completed, workstation B processes the jobs for step 3 at maximal rate $\mu_3 = \frac{1}{0.3} \left[\frac{\text{job}}{\text{time-units}}\right]$ until buffer 3 becomes empty. During this action, buffer 2 remains constant at 500 jobs due to the fact that workstation B is processing the jobs for step 3 and no jobs are entering buffer 2. When buffer 3 becomes empty, i.e. at point $(x_2, x_3) = (500, 0)$, workstation B has completed production step 3 and is setting up for processing the jobs

for step 2. During this setup, buffer 2 remains constant at 500 jobs and buffer 3 remains empty. After this setup has been completed, workstation B is processing the jobs for step 2 at maximal rate $\mu_2 = \frac{1}{0.6}$ $\left[\frac{\text{job}}{\text{time-units}}\right]$. According to the behavior in Figure 3.24, buffer 2 decreases first to 0 jobs and increases then to 500 jobs. During this production step, buffer 3 increases from 0 to 1000 jobs. After this production step, the periodic orbit operates then again in point $(x_2, x_3) = (500, 1000)$ jobs which completes the cycle of this periodic orbit.

The desired periodic orbits in Figure 4.1 minimize the weighted amount of jobs in the specific reentrant system. Every initial condition $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2,$
$x_3, x_4)$ which lies on the desired periodic orbit of the system results in desired periodic system behavior. Any other initial condition which does not lie on the desired periodic orbit, can result in a translated desired periodic orbit of the system. The desired periodic orbit of the system is then translated in the positive x_1, x_2, x_3 and/or x_4 direction. It turns out, that either one or more translated desired periodic orbits of the system are possible. When the initial condition says that one or both workstations in the system are performing a setup, only one translated desired periodic orbit of the system is possible. In case the initial condition says that both workstations are processing the jobs at maximal rate, more than one translated desired periodic orbits can be obtained. Both cases are explained further by means of an example.

First, consider the case that only one translated desired periodic orbit of the system is possible. This case can only occur when the initial condition says that one or both workstations in the system are performing a setup. According to the remaining setup time, only one translated desired periodic orbit of the system is possible. Consider workstation A and suppose that an initial condition is given which says that workstation A is starting a setup, i.e. $x_0^A = 50$, and the system is setting up for processing the jobs for step 1 with initial buffer contents $(x_1, x_4) = (1400, 800)$. The desired periodic orbit of workstation A and the point $(x_1, x_4) = (1400, 800)$ is presented in the left hand side of Figure 4.2. With the initial condition which says that workstation A is starting a setup for step 1, only the translated desired periodic orbit presented in the right hand side of Figure 4.2 can be obtained.

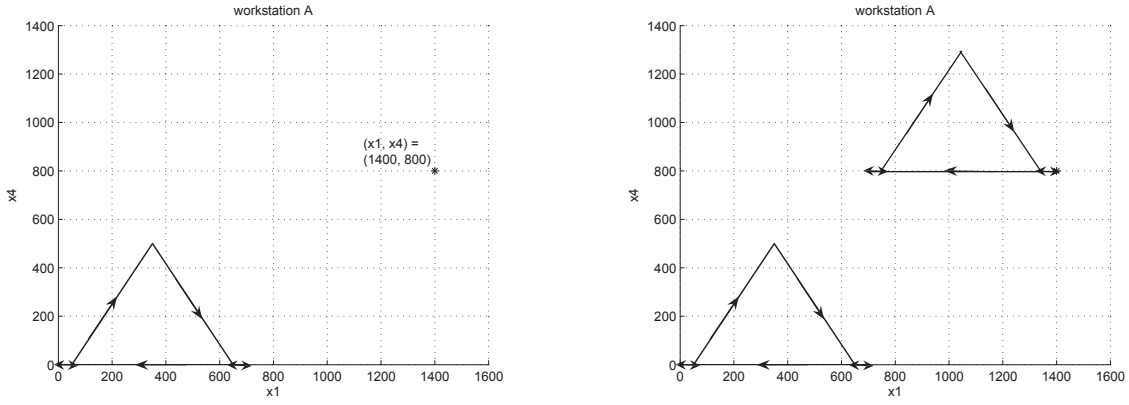


Figure 4.2: If the system is performing a setup, only one translated desired periodic orbit is possible.

In general, every initial condition which says that one or both workstations in the system are performing a setup results in only one possible translated desired periodic orbit of the system.

Consider now the case that more than one translated desired periodic orbit of the system is possible. This case can only occur when the initial condition says that both workstations in the system are processing the jobs. Assume the periodic orbit of workstation B and the initial condition that workstation B is processing the jobs for step 3 with initial buffer contents $(x_2, x_3) = (1500, 2000)$. In the left hand side of Figure 4.3, the desired periodic obit of workstation B and the point $(x_2, x_3) = (1500, 2000)$ are shown. This initial condition says only that workstation B is processing the jobs for step 3. It is not known where the system operates during this production step. The point $(x_2, x_3) = (1500, 2000)$ can be either a start point or an end point for this production step. These two extreme translated desired periodic orbits of workstation B are presented in the right hand side of Figure 4.3. For the translated desired periodic orbit above this point, workstation B just finished processing the jobs for step 3. While for the other translated desired periodic orbit, workstation B just started processing the jobs for step 3. Besides these two extreme translated desired periodic orbits, more translated desired periodic orbits can be obtained which lies between these extreme ones.

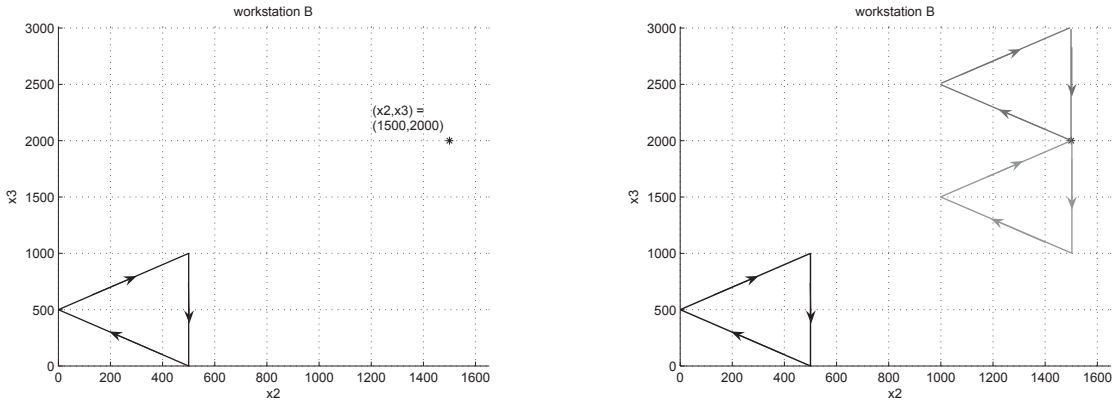


Figure 4.3: Two possible translated desired periodic orbits which through point $(x_2, x_3) = (1500, 2000)$.

As mentioned earlier, a non-distributed controller has to be designed which brings the reentrant system towards the desired periodic orbit from any initial condition. For the case when more than one translated desired periodic is possible, the following question arises: "Which translated desired periodic orbit that goes through the point of the initial condition is taken as a starting point for the controller?". This question can be answered, when every translated desired periodic orbit is associated with the mean amount of work in the system which is explained in next section.

4.2 Mean amount of work in the system

The amount of work in the system is defined as the remaining process time of all jobs present in the system. Jobs stored in different buffers have different weights in amount of work. A job which is stored in buffer 1 needs $\left(\frac{1}{\mu_1} =\right) 0.3$ time-units for processing step 1, $\left(\frac{1}{\mu_2} =\right) 0.6$ time-units for step 2, $\left(\frac{1}{\mu_3} =\right) 0.3$ time-units for step 3 and $\left(\frac{1}{\mu_4} =\right) 0.6$ time-units for step 4. The amount of work associated with a job stored in buffer 1 equals $[0.3 + 0.6 + 0.3 + 0.6 =] 1.8$

time-units. The same calculation can be done for a job stored in buffers 2, 3 and 4. The amount of work in the system at a certain moment of time is given by $1.8x_1(t) + 1.5x_2(t) + 0.9x_3(t) + 0.6x_4(t)$. With this equation and with the desired periodic system behavior, the amount of work in the system can easily be determined which is presented in the right hand side of Figure 4.4.

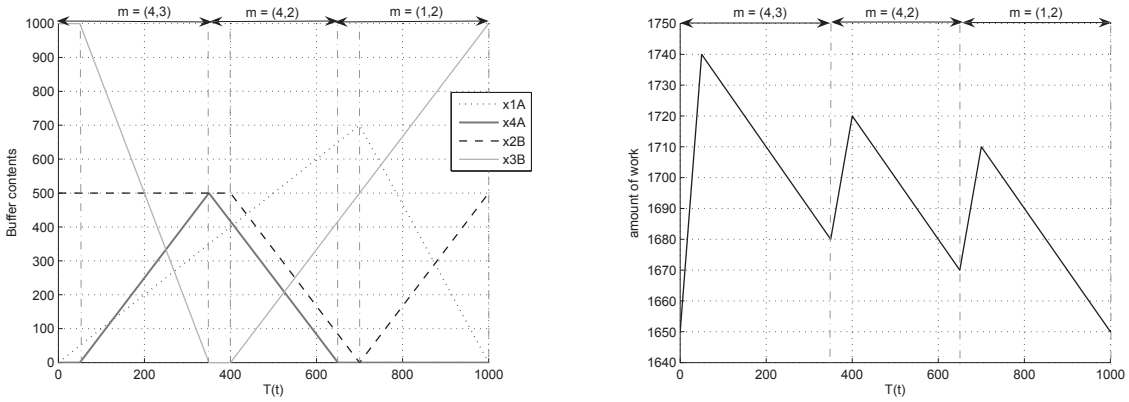


Figure 4.4: Amount of work of the desired periodic system behavior.

Table 4.1: System state and amount of work in the system at important time moments.

t	m_A	m_B	x_0^A	x_0^B	x_1	x_2	x_3	x_4	amount of work
0	4	3	50	50	0	500	1000	0	1650
50	4	3	0	0	50	500	1000	0	1740
350	4	2	0	50	350	500	0	500	1680
400	4	2	0	0	400	500	0	$\frac{1250}{3}$	1720
650	1	2	50	0	650	$\frac{250}{3}$	$\frac{1250}{3}$	0	1670
700	1	2	0	0	700	0	500	0	1710
1000	4	3	50	50	0	500	1000	0	1650

In both figures, the system modes are also presented. Every system mode can be split into two parts, a setup part where the system is setting up to another mode which always takes 50 time-units and a processing part where each workstation is processing the jobs at maximal rate. For simplicity, the system state when a new part starts is presented in Table 4.1. Also, the amount of work in the system is presented in the last column of this table. When a workstation is performing a setup, no jobs can be processed at this workstation. But during this setup, jobs can arrive into the buffers of this workstation which results in an increasing amount of work in the system. In case, when both workstations are processing the jobs at maximal rate, the amount of work in the system decreases. The processed jobs move to a buffer with a lower weight in amount of work.

During a cycle period T, the mean amount of work in the specific reentrant system can be determined with (4.1).

$$\frac{1}{T}\int_0^T 1.8x_1(t) + 1.5x_2(t) + 0.9x_3(t) + 0.6x_4(t)\ dt. \tag{4.1}$$

The mean amount of work in the system which is presented in Figure 4.4 is equal to 1695 time-units. This value — which is associated with the periodic orbit that results in the smallest mean amount of jobs in the system — is used in the controller design. Depending on the initial condition, the desired periodic orbits in Figure 4.1 are translated in the positive x_1, x_2, x_3 and/or x_4 direction. This translation results in another feasible translated desired periodic orbit for the system with a higher mean amount of work. Due to the fact that more jobs are present in the system. As already mentioned, either one or more translated desired periodic orbits can be obtained which go through the point of the initial condition. The mean amount of work in the system of all these possible translated desired periodic orbits can be determined with (4.1). The translated desired periodic orbit with the smallest mean amount of work in the system is taken as a starting point for the controller. The designed controller will bring this translated desired periodic orbit towards the desired periodic orbit of the system by continuously subtracting the value 1695 from the mean amount of work of the translated desired periodic orbit until this difference is equal to zero. In that case, the controller has to make the system converge from the initial condition to the desired periodic orbit.
Before this controller is designed, first a set is defined which contains all possible translated desired periodic orbits which go through the point of the initial condition. This set is used in the controller design as the feasible domain. Every initial condition which lies in this domain results in a feasible translated desired periodic orbit.

4.3 Feasible domain

In this section, the feasible domain is determined. Every arbitrary point in this domain results in a feasible periodic orbit of the system. It is known that depending on the initial condition, one or more translated desired periodic orbits can be obtained. The desired periodic orbits in Figure 4.1 are translated in the positive x_1, x_2, x_3 and/or x_4 direction. As a result, if the system is in a state defined by the initial condition which can be on a translated desired periodic orbit, it is possible for the system to stay on that translated desired periodic orbit. In particular, this means that if the state of the system is in the set of points through which at least one translated periodic orbit goes, it is possible to stay in that set. Before this set is defined, the periodic orbits in Figure 4.1 are characterized first mathematically.

As already mentioned in Section 2.2, the system state x is given by $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4)$. The desired periodic orbits in Figure 4.1 can be expressed in these system state variables. The desired system behavior in the left hand side of Figure 4.4 is used for this characterization. This desired system behavior is divided into six parts and the start of every part is attended with a change in the state of the system. For each part in each system mode the buffer contents of each buffer can be expressed in these state variables. While looking at Table 4.1, the buffer contents in each part can be characterized as:

$$\text{for } \mathbf{m} = (4,3) \begin{cases} \text{if } x_0^A > 0 \ \wedge \ x_0^A = x_0^B : \\ \qquad\qquad x_1 = 50 - x_0^A \ \wedge \ x_2 = 500 \ \wedge \ x_3 = 1000 \ \wedge \ x_4 = 0; \\ \text{if } x_0^A = 0 \ \wedge \ x_0^B = 0 \ \wedge \ x_1 \geq 50 : \\ \qquad\qquad x_1 + \frac{3}{10} x_3 = 350 \ \wedge \ x_2 = 500 \ \wedge \ \frac{1}{2} x_3 + x_4 = 500; \end{cases}$$

$$\text{for } \mathbf{m}=(4,2)\begin{cases}\text{if } x_0^A=0 \ \wedge \ x_0^B>0: \\ \qquad x_1=400-x_0^B \ \wedge \ x_2=500 \ \wedge \ x_3=0 \ \wedge \ x_4=\frac{1250}{3}+\frac{5}{3}x_0^B; \\ \text{if } x_0^A=0 \ \wedge \ x_0^B=0 \ \wedge \ x_1\geq 400 \ \wedge \ x_2\geq\frac{250}{3}: \\ \qquad x_1+\frac{3}{5}x_4=650 \ \wedge \ x_2+x_3=500 \ \wedge \ x_3+x_4=\frac{1250}{3};\end{cases}$$

$$\text{for } \mathbf{m}=(1,2)\begin{cases}\text{if } x_0^A>0 \ \wedge \ x_0^B=0: \\ \qquad x_1=700-x_0^A \ \wedge \ x_2=\frac{5}{3}x_0^A \ \wedge \ x_3=500-\frac{5}{3}x_0^A \ \wedge \ x_4=0; \\ \text{if } x_0^A=0 \ \wedge \ x_0^B=0 \ \wedge \ x_3\geq 500: \\ \qquad \frac{5}{7}x_1+x_2=500 \ \wedge \ \frac{5}{7}x_1+x_3=1000 \ \wedge \ x_4=0.\end{cases}$$

As a translated desired periodic orbit is only possible in positive x_1, x_2, x_3 and/or x_4 direction, the set of states through which at least one translated desired periodic orbit is possible can be determined by changing the equality-signs '=' in above-mentioned equations for the buffer contents by inequality-signs '$\geq$'. Then, the feasible domain or the set of states through which at least one translated desired periodic orbit is possible can be characterized as:

$$\wp=\left\{(\mathbf{m},\boldsymbol{x_0},\mathbf{x})\;\middle|\;\begin{array}{l}\text{if } \mathbf{m}=(4,3),\ x_0^A>0,\ x_0^A=x_0^B: \\ \qquad x_1\geq 50-x_0^A \ \wedge \ x_2\geq 500 \ \wedge \ x_3\geq 1000 \ \wedge \ x_4\geq 0; \\ \text{if } \mathbf{m}=(4,3),\ x_0^A=x_0^B=0,\ x_1\geq 50: \\ \qquad x_1+\frac{3}{10}x_3\geq 350 \ \wedge \ x_2\geq 500 \ \wedge \ \frac{1}{2}x_3+x_4\geq 500; \\ \text{if } \mathbf{m}=(4,2),\ x_0^A=0,\ x_0^B>0: \\ \qquad x_1\geq 400-x_0^B \ \wedge \ x_2\geq 500 \ \wedge \ x_3\geq 0 \ \wedge \ x_4\geq\frac{1250}{3}+\frac{5}{3}x_0^B; \\ \text{if } \mathbf{m}=(4,2),\ x_0^A=x_0^B=0,\ x_1\geq 400,\ x_2\geq\frac{250}{3}: \\ \qquad x_1+\frac{3}{5}x_4\geq 650 \ \wedge \ x_2+x_3\geq 500 \ \wedge \ x_3+x_4\geq\frac{1250}{3}; \\ \text{if } \mathbf{m}=(1,2),\ x_0^A>0,\ x_0^B=0: \\ \qquad x_1\geq 700-x_0^A \ \wedge \ x_2\geq\frac{5}{3}x_0^A \ \wedge \ x_3\geq 500-\frac{5}{3}x_0^A \ \wedge \ x_4\geq 0; \\ \text{if } \mathbf{m}=(1,2),\ x_0^A=x_0^B=0,\ x_3\geq 500: \\ \qquad \frac{5}{7}x_1+x_2\geq 500 \ \wedge \ \frac{5}{7}x_1+x_3\geq 1000 \ \wedge \ x_4\geq 0.\end{array}\right\}$$

This set of translated desired periodic orbits going through an arbitrary point $\mathrm{x}\in\wp$ is used in the controller design which is explained in next section.

4.4 Controller design

In previous sections, the desired periodic orbit of the system, the mean amount of work in the system and the feasible domain are explained. The next step is to design a controller which brings the system from any initial condition towards the desired periodic orbit. This controller design is based on Lyapunov's direct method. In general, the basic idea behind this method is that if the total energy of a mechanical or electrical system is continuously dissipated, then the system has to settle down to an equilibrium point or state in which the energy remains constant. The system operates then in steady state and stability of the system is concluded. If such an "energy-function" can be found for the system and a controller is designed such that this energy is decreasing all the time, the system converges then from the initial condition to steady state [LR06b].

In the remainder of this section, first a candidate for this energy function is proposed, a so called Lyapunov function candidate V. This candidate is defined for a large subset of the

state space. In next section, the time derivative of the Lyapunov function candidate ($\dot{V}$) is considered along solutions of the system. Clearly, this time derivative depends on the input. This input is chosen such that in each point the time derivative of the Lyapunov function candidate is minimized over the set of all allowed inputs. Since the minimizing input depends on the current state, a state feedback is obtained. As a next step, since the controller which has been derived by means of the Lyapunov function candidate is valid for a subset of the state space (for $\mathrm{x} \in \wp$), it needs to be extended to one which is defined for the entire state space.

Lyapunov function candidate

For defining the Lyapunov function candidate V, the feasible domain $\wp$ or the set of possible translated desired periodic orbits is used. More precisely, for an arbitrary initial condition $\mathrm{x} = (\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) \in \wp$, consider the set of all translated desired periodic orbits going through this point. Each translated desired periodic orbit in this set is associated with the mean amount of work in the system. The translated desired periodic orbit in this set with the smallest mean amount of work in the system is used as a starting point. The mean amount of work of the desired periodic orbit is subtracted from this smallest mean amount of work. This difference is associated with an arbitrary point $\mathrm{x} \in \wp$, i.e. this defines $V(\mathrm{x})$.

The Lyapunov function candidate $V(\mathrm{x})$ for the system should have the following properties:

- $V(\mathrm{x}) = 0$, if and only if x lies on the desired periodic orbit;
- $V(\mathrm{x}) > 0$, if x lies on a translated desired periodic orbit;
- $V(\mathrm{x})$ is continuous;
- $\dot{V}(\mathrm{x}) \leq 0$ along solutions of the system.

If an arbitrary point $\mathrm{x} \in \wp$ lies on the desired periodic orbit of the system, the Lyapunov function candidate equals zero. For any other point $\mathrm{x} \in \wp$ which does not lie on the desired periodic orbit, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in comparison with the amount of work of the desired periodic orbit which results in $V(\mathrm{x}) > 0$. If this Lyapunov function candidate for the system is continuous, the controller can make the system continuously converge from the initial condition to the desired periodic orbit. This convergence is done by minimizing $\dot{V}(\mathrm{x})$ over the set of allowed inputs that assure that the system remains in the feasible domain $\wp$.

During a cycle period, the system operates in three different system modes, see Figure 4.4. For each system mode, the Lyapunov function candidate $V(\mathrm{x})$ is derived step by step. Also, the translated desired periodic orbit in the set $\wp$ with the smallest mean amount of work in the system is determined for an arbitrary point $\mathrm{x} \in \wp$.

System mode (4,3)

In the beginning of this system mode, both workstations are performing a setup. According to the remaining setup time, only one translated desired periodic orbit is possible through

an arbitrary point $\mathrm{x} \in \wp$. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit. For this case, the Lyapunov function candidate $V(\mathrm{x})$ is defined as:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 - 50 + x_0^A\right) + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\left(x_4\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{9}{5}x_0^A - 1740, \\ &\text{for } \mathbf{m} = (4,3) \wedge\ x_0^A > 0\ \wedge\ x_0^A = x_0^B\ \wedge\ x_1 \geq 50 - x_0^A\ \wedge\ x_2 \geq 500 \wedge \\ &\qquad x_3 \geq 1000\ \wedge\ x_4 \geq 0. \end{aligned} \tag{4.2}$$

After this setup has been completed, both workstations are processing the jobs at maximal rate. For this system state, the set $\wp$ contains a lot of translated desired periodic orbits which go through an arbitrary point $\mathrm{x} \in \wp$. Looking better at Figure 4.4, only the buffer contents of buffer 3 decreases during this system mode. During this system mode, only the buffer contents of buffer 3 becomes empty and reaches the boundary of the feasible domain. Therefore, buffer 3 is taken into account for finding a valid Lyapunov function candidate for this system state. Depending on the initial buffer contents of buffer 3 in an arbitrary point $\mathrm{x} \in \wp$, two Lyapunov function candidates have to be defined. A Lyapunov function candidate where buffer 3 implies an extra amount of work in comparison with the desired periodic orbit and a Lyapunov function candidate where buffer 3 does not imply an extra amount of work in comparison with the desired periodic orbit.

First, consider the case when buffer 3 implies an extra amount of work in comparison with the desired periodic orbit. This case only occurs when the initial buffer contents of buffer 3 is larger than the maximal buffer contents of buffer 3 in the desired periodic orbit. From Figure 4.4, it is known that the maximal buffer contents of buffer 3 in the desired periodic orbit contains 1000 jobs. If the initial buffer contents of buffer 3 is larger than or equal to 1000 jobs in an arbitrary point $\mathrm{x} \in \wp$, then a lot of translated desired periodic orbits are possible which all have a higher amount of work in comparison with the desired periodic orbit. The translated desired periodic orbit which starts processing the jobs for step 3 in point $\mathrm{x} \in \wp$ is the one with the smallest mean amount of work in the system, see also Figure 4.2. For this case, i.e. for $x_3 \geq 1000$ jobs, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in comparison with the amount of work of the desired periodic orbit:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 - 50\right) + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\left(x_4\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1740, \\ &\text{for } \mathbf{m} = (4,3)\ \wedge\ x_0^A = x_0^B = 0\ \wedge\ x_1 \geq 50\ \wedge\ x_1 + \frac{3}{10}x_3 \geq 350\ \wedge\ x_2 \geq 500 \wedge \\ &\qquad x_3 \geq 1000\ \wedge\ \frac{1}{2}x_3 + x_4 \geq 500. \end{aligned} \tag{4.3}$$

In case when the initial buffer contents of buffer 3 is smaller than or equal to 1000 jobs in an arbitrary point $\mathrm{x} \in \wp$, then buffer 3 does not imply an extra amount of work in comparison with the desired periodic orbit. For this case, the set $\wp$ contains a lot of possible translated desired periodic orbits. The translated desired periodic orbit with the smallest mean amount of work in the system is the periodic orbit which starts processing the jobs for step 3 in point $x_3 = 1000$. To make this more clear, consider the case for point $(x_2, x_3) = (800, 500)$ which

is sketched in Figure 4.5. For this case, the translated desired periodic orbit which finishes processing the jobs for step 3 in this point is moved in the direction of the arrow — which indicates a decrease in amount of work in the system — to the translated desired periodic orbit with the smallest mean amount of work in the system. This translated desired periodic orbit is the one which starts processing the jobs for step 3 in point $x_3 = 1000$. For this system state, the buffer contents of buffer 3 does not imply an extra amount of work in comparison with the desired periodic orbit. Buffer 3 of the translated desired periodic orbit is bounded by the same values as the desired periodic orbit. For this system state, the Lyapunov function candidate is defined as:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 + \frac{3}{10}x_3 - 350\right) + 1.5\left(x_2 - 500\right) + 0.6\left(\frac{1}{2}x_3 + x_4 - 500\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{21}{25}x_3 + \frac{3}{5}x_4 - 1680, \\ &\text{for } \mathbf{m} = (4,3) \ \wedge \ x_0^A = x_0^B = 0 \ \wedge \ x_1 \geq 50 \ \wedge \ x_1 + \frac{3}{10}x_3 \geq 350 \ \wedge \ x_2 \geq 500 \ \wedge \\ &\quad x_3 \leq 1000 \ \wedge \ \frac{1}{2}x_3 + x_4 \geq 500. \end{aligned} \tag{4.4}$$

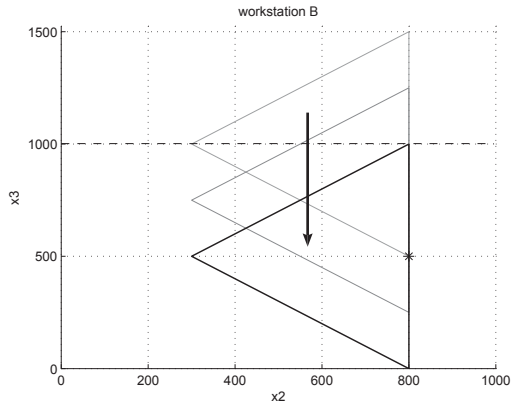


Figure 4.5: Finding the translated desired periodic orbit with the smallest mean amount of work in the system.

The Lyapunov function candidate $V(\mathrm{x})$ has to be continuous during the desired periodic orbit. This continuity between two successive Lyapunov function candidates can be checked by comparing the same system state. Lyapunov function candidate (4.3) should have the same expression as Lyapunov function candidate (4.2) when the system completes the setup. Filling in $x_0^A = 0$ in (4.2) leads to the same expression as (4.3) which proves continuity. The same holds for Lyapunov function candidates (4.3) and (4.4). (4.3) is defined for the case when the initial buffer contents of buffer 3 is larger than or equal to 1000 jobs in system mode $(4,3)$ while (4.4) is defined for the case when $x_3 \leq 1000$ jobs. Filling in $x_3 = 1000$ in both Lyapunov function candidates lead to the same expressions of $V(\mathrm{x})$ which proves continuity of $V(\mathrm{x})$.

System mode (4,2)

After the system has completed production step 3, the system is switching to mode $(4,2)$. In the beginning of this system mode, workstation B is performing a setup. According to this remaining setup time, only one translated desired periodic orbit is possible through an arbitrary point $\mathrm{x} \in \wp$. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit which becomes:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 - 400 + x_0^B\right) + 1.5\left(x_2 - 500\right) + 0.9\left(x_3\right) + 0.6\left(x_4 - \frac{1250}{3} - \frac{5}{3}x_0^B\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{4}{5}x_0^B - 1720, \\ &\text{for } \mathbf{m} = (4,2) \ \wedge \ x_0^A = 0 \ \wedge \ x_0^B > 0 \ \wedge \ x_1 \geq 400 - x_0^B \ \wedge \ x_2 \geq 500 \ \wedge \\ &\qquad x_3 \geq 0 \ \wedge \ x_4 \geq \frac{1250}{3} + \frac{5}{3}x_0^B. \end{aligned} \tag{4.5}$$

After this setup has been completed, both workstations are processing the jobs at maximal rate. In this system state, a lot of translated desired periodic orbits are possible through an arbitrary point $\mathrm{x} \in \wp$. Looking better at Figure 4.4, the buffer contents of buffers 2 and 4 decrease during this system mode. Depending on the initial condition, buffer 2 and/or buffer 4 can reach the boundary of the feasible domain. Therefore, buffers 2 and 4 are taken into account for finding a valid Lyapunov function candidate for this system state. From Figure 4.4, it is known that the buffer contents of buffers 2 and 4 of the desired periodic orbit are bounded by $\frac{250}{3} \leq x_2 \leq 500$ and $0 \leq x_4 \leq \frac{1250}{3}$. But depending on the initial buffer contents of buffers 2 and 4 in an arbitrary point $\mathrm{x} \in \wp$, it turns out that different 5 cases can occur which are presented in Table 4.2.

Table 4.2: System mode (4,2): 5 different cases.

case 1	$x_2 \geq 500$	$x_4 \geq \frac{1250}{3}$	
case 2	$x_2 \geq 500$	$0 \leq x_4 \leq \frac{1250}{3}$	
case 3	$\frac{250}{3} \leq x_2 \leq 500$	$x_4 \geq \frac{1250}{3}$	
case 4	$\frac{250}{3} \leq x_2 \leq 500$	$0 \leq x_4 \leq \frac{1250}{3}$	$x_2 - x_4 \leq \frac{250}{3}$
case 5	$\frac{250}{3} \leq x_2 \leq 500$	$0 \leq x_4 \leq \frac{1250}{3}$	$x_2 - x_4 \geq \frac{250}{3}$

Before a Lyapunov function candidate $V(\mathrm{x})$ is defined for every case, another view of the desired periodic orbit of the system is presented first. In Figure 4.1, the desired periodic orbit of each workstation is shown. But for this system state, the desired periodic orbit presented in Figure 4.6 — where buffer 2 is plotted against buffer 4 — is more usable for determining the translated desired periodic orbit with the smallest mean amount of work in the system.

According to the desired periodic system behavior presented in Figure 4.4, this view of the desired periodic orbit of the system can easily be determined. In the first 50 time-units, the system is setting up for mode $(4,3)$. After this setup has been completed, i.e. at point $(x_2, x_4) = (500, 0)$, both workstations are processing the jobs at maximal rate. During this action, buffer 2 remains constant and buffer 4 increases to 500 jobs. The system operates then

in point $(x_2, x_4) = (500, 500)$ and workstation B has completed production step 3 and the system is setting up for mode $(4, 2)$. During this setup, workstation A continues processing the jobs for step 4. After this setup has been completed, i.e. at point $(x_2, x_4) = (500, \frac{1250}{3})$, both workstations are processing the jobs at maximal rate until buffer 4 becomes empty. The system operates then in point $(x_2, x_4) = (\frac{250}{3}, 0)$ and workstation A has completed production step 4 and the system is setting up for mode $(1, 2)$. During this setup, workstation B continues processing the jobs for step 2. After this setup has been completed, i.e. at point $(x_2, x_4) = (0, 0)$, workstation A processes the jobs for step 1 and workstation B processes the jobs for step 2. During this action, buffer 4 remains empty and buffer 2 increases to 500 jobs. The system operates then in point $(x_2, x_4) = (500, 0)$ and both workstations have completed the production step and the system is setting up for mode $(4, 3)$ which completes the cycle of this periodic orbit.

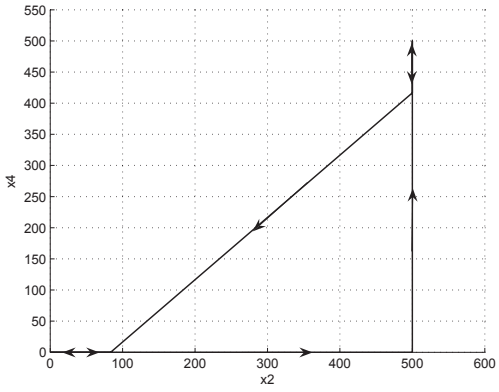


Figure 4.6: Desired periodic orbit of the system.

For every case in Table 4.2, this desired periodic orbit of the system is used for determining a Lyapunov function candidate $V(\mathrm{x})$. Also, the translated desired periodic orbit through an arbitrary point $\mathrm{x} \in \wp$ with the smallest mean amount of work in the system is determined.

First consider *case* 1, where the initial buffer contents of buffer 2 is larger than or equal to 500 jobs and the initial buffer contents of buffer 4 is larger than or equal to $\frac{1250}{3}$ jobs in an arbitrary point $\mathrm{x} \in \wp$. For this case, both buffers imply an extra amount of work in comparison with the desired periodic orbit. The set $\wp$ contains a lot of possible translated desired periodic orbits which go through an arbitrary point $\mathrm{x} \in \wp$. The translated desired periodic orbit where point $\mathrm{x} \in \wp$ corresponds with the maximal buffer contents of buffer 4 is the one with the smallest mean amount of work in the system. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\,(x_1 - 400) + 1.5\,(x_2 - 500) + 0.9\,(x_3) + 0.6\left(x_4 - \frac{1250}{3}\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1720, \\ &\text{for } \mathbf{m} = (4,2) \ \wedge \ x_0^A = x_0^B = 0 \ \wedge \ x_1 \geq 400 \ \wedge \ x_1 + \frac{3}{5}x_4 \geq 650 \ \wedge \\ &\qquad x_2 \geq 500 \ \wedge \ x_3 + x_4 \geq \frac{1250}{3} \ \wedge \ x_4 \geq \frac{1250}{3}. \end{aligned} \tag{4.6}$$

For *case* 2, where the initial buffer contents of buffer 2 is larger than or equal to 500 jobs and the initial buffer contents of buffer 4 is equal to $0 \leq x_4 \leq \frac{1250}{3}$ jobs in an arbitrary point $\mathrm{x} \in \wp$, buffer 2 implies an extra amount of work in comparison with the desired periodic orbit. The set $\wp$ contains a lot of possible translated desired periodic orbits which go through an arbitrary point $\mathrm{x} \in \wp$. The translated desired periodic orbit which starts processing the jobs for step 2 in point $\mathrm{x} \in \wp$ is the one with the smallest mean amount of work in the system. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 + \frac{3}{5}x_4 - 650\right) + 1.5\left(x_2 - x_4 - \frac{250}{3}\right) + 0.9\left(x_3 + x_4 - \frac{1250}{3}\right) \\ &= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{87}{25}x_4 - 1670, \\ &\text{for } \mathbf{m} = (4,2) \ \wedge \ x_0^A = x_0^B = 0 \ \wedge \ x_1 \geq 400 \ \wedge \ x_1 + \frac{3}{5}x_4 \geq 650 \ \wedge \\ &\qquad x_2 \geq 500 \ \wedge \ x_3 + x_4 \geq \frac{1250}{3} \ \wedge \ 0 \leq x_4 \leq \frac{1250}{3} \ \wedge \ x_2 - x_4 \geq \frac{250}{3}. \end{aligned} \tag{4.7}$$

For *case* 3, where the initial buffer contents of buffer 2 is equal to $\frac{250}{3} \leq x_2 \leq 500$ jobs and the initial buffer contents of buffer 4 is larger than or equal to $x_4 \geq \frac{1250}{3}$ jobs in an arbitrary point $\mathrm{x} \in \wp$, buffer 4 implies an extra amount of work in comparison with the desired periodic orbit. The set $\wp$ contains a lot of possible translated desired periodic orbits which go through an arbitrary point $\mathrm{x} \in \wp$. The translated desired periodic orbit where point $\mathrm{x} \in \wp$ corresponds with the maximal buffer contents of buffer 4 is the one with the smallest mean amount of work in the system. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit:

$$\begin{aligned} V(\mathrm{x}) &= 1.8\left(x_1 + \frac{3}{5}x_2 - 700\right) + 0.9\,(x_2 + x_3 - 500) + 0.6\left(x_4 - x_2 + \frac{250}{3}\right) \\ &= \frac{9}{5}x_1 + \frac{69}{50}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1660, \\ &\text{for } \mathbf{m} = (4,2) \ \wedge \ x_0^A = x_0^B = 0 \ \wedge \ x_1 \geq 400 \ \wedge \ x_1 + \frac{3}{5}x_4 \geq 650 \ \wedge \\ &\qquad \frac{250}{3} \leq x_2 \leq 500 \ \wedge \ x_3 + x_4 \geq \frac{1250}{3} \ \wedge \ x_4 \geq \frac{1250}{3} \ \wedge \ x_2 - x_4 \leq \frac{250}{3}. \end{aligned} \tag{4.8}$$

For *case* 4 and *case* 5, where both initial buffer contents are bounded by $\frac{250}{3} \leq x_2 \leq 500$ and $0 \leq x_4 \leq \frac{1250}{3}$, a Lyapunov function candidate is defined for the buffer which reaches

the minimal buffer contents as first. This buffer implies less amount of work in comparison with another buffer. Depending on the initial condition, the buffer which reaches the minimal buffer contents as first can be determined with the equation presented in the last column of Table 4.2. Consider *case* 4 with $x_2 - x_4 \leq \frac{250}{3}$. In this case, the buffer contents of buffer 4 can be larger than the buffer contents of buffer 2. Therefore, buffer 2 reaches the minimal buffer contents as first and implies less amount of work in comparison with buffer 4. For this case, a Lyapunov function candidate has to be defined where buffer 2 does not imply an extra amount of work in comparison with the desired periodic orbit. The Lyapunov function candidate which is defined for *case* 3 is also a suitable Lyapunov function candidate for *case* 4. In both cases, buffer 2 does not imply an extra amount of work in comparison with the desired periodic orbit.

To make *case* 4 more clear, an example is given. Consider the point $(x_2, x_4) = (\frac{1250}{3}, \frac{1250}{3})$ which is sketched in Figure 4.7. In this point the initial buffer contents of buffers 2 and 4 are bounded by $\frac{250}{3} \leq x_2 \leq 500$, $0 \leq x_4 \leq \frac{1250}{3}$ and $x_2 - x_4 \leq \frac{250}{3}$. For this case, the set $\wp$ contains a lot of possible translated desired periodic orbits which goes through this point. When the desired periodic orbit is translated in the positive x_4 direction until the point $(x_2, x_4) = (\frac{1250}{3}, \frac{1250}{3})$ lies on this desired periodic orbit, a translated desired periodic orbit with the smallest mean amount of work in the system is founded.

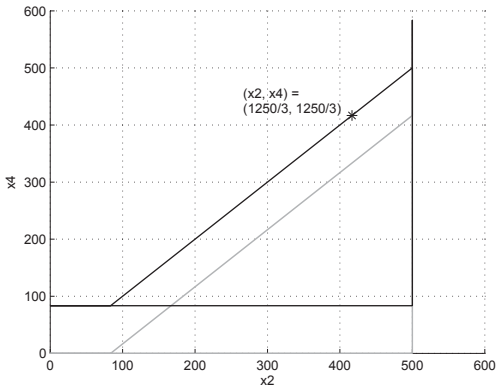


Figure 4.7: Translated desired periodic orbit through $(\frac{1250}{3}, \frac{1250}{3})$.

Consider *case* 5 with $x_2 - x_4 \geq \frac{250}{3}$. In this case, the initial buffer contents of buffer 2 can be larger than the buffer contents of buffer 4. Therefore, buffer 4 reaches the minimal buffer contents as first and implies less amount of work in comparison with buffer 2. The Lyapunov function candidate which is defined for *case* 2 is also a suitable Lyapunov function candidate for *case* 5. In both cases, buffer 4 does not imply an extra amount of work in comparison with the desired periodic orbit.

To make *case* 5 more clear, an example is given. Consider the point $(x_2, x_4) = (450, 200)$ which is sketched in Figure 4.8. In this point the initial buffer contents of buffers 2 and 4 are bounded by $\frac{250}{3} \leq x_2 \leq 500$, $0 \leq x_4 \leq \frac{1250}{3}$ and $x_2 - x_4 \geq \frac{250}{3}$. For this case, the set $\wp$ contains a lot of possible translated desired periodic orbits which goes through this point. When the desired periodic orbit is translated in the positive x_2 direction until the point $(x_2, x_4) = (450, 200)$ lies on this desired periodic orbit, a translated desired periodic orbit with the smallest mean amount of work in the system is founded.

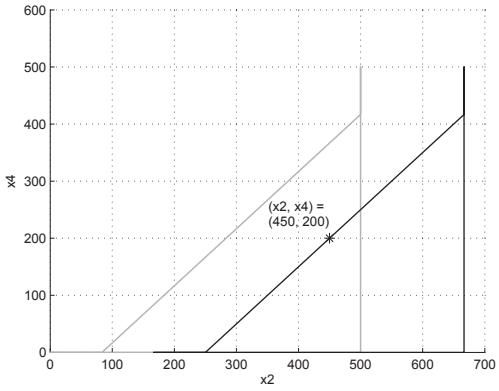


Figure 4.8: Translated desired periodic orbit through $(450, 200)$.

The Lyapunov function candidate $V(\mathrm{x})$ has to be continuous during the desired periodic orbit. This continuity between two successive Lyapunov function candidates can be checked by comparing the same system state. Lyapunov function candidate (4.4) should have the same expression as Lyapunov function candidate (4.5) when the system completes production step 3 and starts setting up for mode $(4, 2)$. Filling in $x_3 = 0$ and $x_0^B = 50$ in both Lyapunov function candidates lead to the same expressions of $V(\mathrm{x})$ which proves continuity. After this setup has been completed, the system can operate in one of the 5 cases presented in Table 4.2. Therefore, depending on the initial buffer contents of buffers 2 and 4, Lyapunov function candidate (4.5) should have the same expression as (4.6), (4.7) and (4.8) when the same system state is considered. Lyapunov function candidate (4.6), where both buffers imply an extra amount of work, should have the same expression of $V(\mathrm{x})$ as (4.5) when the system completes the setup for mode $(4, 2)$. Filling in $x_0^B = 0$ in (4.5) leads to the same expression as (4.6). Lyapunov function candidates (4.5) and (4.7) should have the same expressions of $V(\mathrm{x})$ when $x_0^B = 0$ and $x_2 = 500$. This is the case which proves continuity. Also, Lyapunov function candidates (4.5) and (4.8) should have the same expressions of $V(\mathrm{x})$ when $x_0^B = 0$ and $x_4 = \frac{1250}{3}$. This is also the case which proves continuity of $V(\mathrm{x})$.
When the system operates in *case* 1 after the setup to mode $(4, 2)$ has been completed, the initial buffer contents of buffers 2 and 4 decrease. The system can operate in one of the other 4 cases. Therefore, Lyapunov function candidate (4.6) should have the same expression as (4.7) and (4.8) for $V(\mathrm{x})$ when $x_2 = 500$ and $x_4 = \frac{1250}{3}$. Filling in these values, all Lyapunov function candidates lead to the same expression of $V(\mathrm{x})$ which proves continuity.

System mode (1,2)

After the system has completed production step 4, the system is switching to mode $(1, 2)$. In the beginning of this system mode, workstation A is performing a setup. According to this remaining setup time, only one translated desired periodic orbit is possible through an arbitrary point $\mathrm{x} \in \wp$. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in the system in point $\mathrm{x} \in \wp$ compared to the desired periodic orbit. For this case, the Lyapunov function candidate $V(\mathrm{x})$ is defined as:

$$
\begin{aligned}
V(\mathrm{x}) &= 1.8\left(x_1 - 700 + x_0^A\right) + 1.5\left(x_2 - \frac{5}{3}x_0^A\right) + 0.9\left(x_3 - 500 + \frac{5}{3}x_0^A\right) + 0.6\left(x_4\right) \\
&= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{4}{5}x_0^A - 1710, \\
&\text{for } \mathbf{m} = (1,2) \ \wedge \ x_0^A > 0 \ \wedge \ x_0^B = 0 \ \wedge \ x_1 \geq 700 - x_0^A \ \wedge \ x_2 \geq \frac{5}{3}x_0^A \ \wedge \\
&\quad x_3 \geq 500 - \frac{5}{3}x_0^A \ \wedge \ x_4 \geq 0.
\end{aligned}
\tag{4.9}
$$

After this setup has been completed, both workstations are processing the jobs at maximal rate. In this system state, a lot of translated desired periodic orbits are possible through an arbitrary point $\mathrm{x} \in \wp$. Looking better at Figure 4.4, only the buffer contents of buffer 1 decreases during this system mode. Buffer 1 can only reach the boundary of the feasible domain. Therefore, buffer 1 is taken into account for finding a valid Lyapunov function candidate for this system state. Depending on the initial buffer contents of buffer 1 in an arbitrary point $\mathrm{x} \in \wp$, two Lyapunov function candidates have to be defined. A Lyapunov function candidate where buffer 1 implies an extra amount of work in comparison with the desired periodic orbit and a Lyapunov function candidate where buffer 1 does not imply an extra amount of work in comparison with the desired periodic orbit.

First, consider the case when buffer 1 implies an extra amount of work in comparison with the desired periodic orbit. This case only occurs when the initial buffer contents of buffer 1 is larger than the maximal buffer contents of buffer 1 in the desired periodic orbit. From Figure 4.4, it is known that the maximal buffer contents of buffer 1 in the desired periodic orbit contains 700 jobs. If the initial buffer contents of buffer 1 is larger than or equal to 700 jobs in an arbitrary point $\mathrm{x} \in \wp$, then a lot of translated desired periodic orbits are possible which all have a higher amount of work in comparison with the desired periodic orbit. The translated desired periodic orbit which starts processing the jobs for step 1 in point $\mathrm{x} \in \wp$ is the one with the smallest mean amount of work in the system. For this system state, the Lyapunov function candidate $V(\mathrm{x})$ is defined as the extra amount of work in comparison with the amount of work of the desired periodic orbit:

$$
\begin{aligned}
V(\mathrm{x}) &= 1.8\left(x_1 - 700\right) + 1.5\left(x_2\right) + 0.9\left(x_3 - 500\right) + 0.6\left(x_4\right) \\
&= \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1710, \\
&\text{for } \mathbf{m} = (1,2) \ \wedge x_0^A = x_0^B = 0 \ \wedge \ x_1 \geq 700 \ \wedge \ \frac{5}{7}x_1 + x_2 \geq 500 \ \wedge \\
&\quad \frac{5}{7}x_1 + x_3 \geq 1000 \ \wedge \ x_3 \geq 500 \ \wedge \ x_4 \geq 0.
\end{aligned}
\tag{4.10}
$$

In case when the buffer contents of buffer 1 is smaller than or equal to 700 jobs in an arbitrary point $\mathrm{x} \in \wp$, buffer 1 does not imply an extra amount of work in comparison with the desired periodic orbit. The translated desired periodic orbit with the smallest mean amount of work in the system is the periodic orbit which started processing the jobs for step 1 the latest. See Figure 4.9, where this case is sketched for point $(x_1, x_4) = (500, 250)$. The mean amount of work of the translated desired periodic orbits decreases in the direction of the arrow from

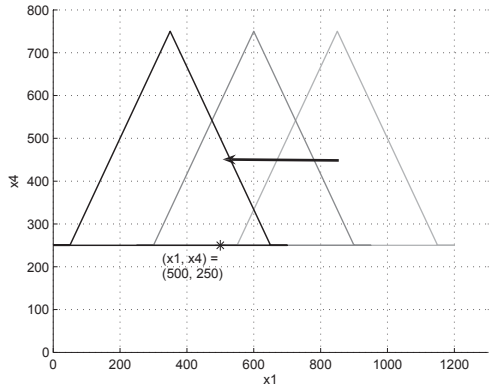


Figure 4.9: Finding the translated desired periodic orbit with the smallest mean amount of work in the system.

the translated desired periodic orbit which ended processing the jobs for step 1 in point $(x_1, x_4) = (500, 250)$ to the translated desired periodic orbit which started processing the jobs for step 1 the latest. For this case, the buffer contents of buffer 1 does not imply an extra amount of work in comparison with the desired periodic orbit. For this system state, the Lyapunov function candidate is defined as:

$$\begin{aligned} V(\mathrm{x}) &= 1.5\left(\frac{5}{7}x_1 + x_2 - 500\right) + 0.9\left(\frac{5}{7}x_1 + x_3 - 1000\right) + 0.6\,(x_4) \\ &= \frac{12}{7}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1650, \\ &\text{for } \mathbf{m} = (1,2) \ \wedge \ x_0^A = x_0^B = 0 \ \wedge \ x_1 \leq 700 \ \wedge \ \frac{5}{7}x_1 + x_2 \geq 500 \ \wedge \\ &\quad \frac{5}{7}x_1 + x_3 \geq 1000 \ \wedge \ x_3 \geq 500 \ \wedge \ x_4 \geq 0. \end{aligned} \tag{4.11}$$

The Lyapunov function candidate $V(\mathrm{x})$ has to be continuous during the desired periodic orbit. This continuity between two successive Lyapunov function candidates can be checked by comparing the same system state. Lyapunov function candidate (4.9) should have the same expression as Lyapunov function candidates (4.7) and (4.8) when the system completes production step 4 and starts setting up for mode $(1, 2)$. For (4.7), filling in $x_4 = 0$ and $x_0^A = 50$ in both Lyapunov function candidates lead to the same expressions of $V(\mathrm{x})$ which proves continuity. For (4.8), filling in $x_2 = \frac{250}{3}$ and $x_0^A = 50$ in both Lyapunov function candidates lead to the same expressions of $V(\mathrm{x})$ which proves continuity. Lyapunov function candidate (4.10) should have the same expression as (4.9) when the system completes the setup for mode $(1, 2)$. Filling in $x_0^A = 0$ in (4.9) leads to the same expression as (4.10). When $x_1 = 700$, Lyapunov function candidates (4.10) and (4.11) should have the same expressions of $V(\mathrm{x})$. This is the case which proves continuity. Also, Lyapunov function candidate (4.11) should have the same expression as Lyapunov function candidate (4.2), when the system completes production step 1 and is setting up to system mode $(4, 3)$. Filling in $x_1 = 0$ and $x_0^A = 50$ in both Lyapunov function candidates lead to the same expressions of $V(\mathrm{x})$ which proves continuity of the Lyapunov function candidate $V(\mathrm{x})$ for the complete system.

As a result, the Lyapunov function candidate is defined for this system as:

$$V(\mathrm{x}) = \begin{cases} \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{9}{5}x_0^A - 1740 & \mathbf{m} = (4,3) \wedge x_0^A > 0 \wedge x_0^A = x_0^B \wedge \\ & x_1 \geq 50 - x_0^A \wedge x_2 \geq 500 \wedge \\ & x_3 \geq 1000 \wedge x_4 \geq 0; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1740 & \mathbf{m} = (4,3) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 50 \wedge \\ & x_1 + \frac{3}{10}x_3 \geq 350 \wedge x_2 \geq 500 \wedge \\ & x_3 \geq 1000 \wedge \frac{1}{2}x_3 + x_4 \geq 500; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{21}{25}x_3 + \frac{3}{5}x_4 - 1680 & \mathbf{m} = (4,3) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 50 \wedge \\ & x_1 + \frac{3}{10}x_3 \geq 350 \wedge x_2 \geq 500 \wedge \\ & x_3 \leq 1000 \wedge \frac{1}{2}x_3 + x_4 \geq 500; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{4}{5}x_0^B - 1720 & \mathbf{m} = (4,2) \wedge x_0^A = 0 \wedge x_0^B > 0 \wedge \\ & x_1 \geq 400 - x_0^B \wedge x_2 \geq 500 \wedge x_3 \geq 0 \wedge \\ & x_4 \geq \frac{1250}{3} + \frac{5}{3}x_0^B; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1720 & \mathbf{m} = (4,2) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 400 \wedge \\ & x_1 + \frac{3}{5}x_4 \geq 650 \wedge x_2 \geq 500 \wedge \\ & x_3 + x_4 \geq \frac{1250}{3} \wedge x_4 \geq \frac{1250}{3}; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{87}{25}x_4 - 1670 & \mathbf{m} = (4,2) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 400 \wedge \\ & x_1 + \frac{3}{5}x_4 \geq 650 \wedge x_2 \geq 500 \wedge \\ & x_3 + x_4 \geq \frac{1250}{3} \wedge 0 \leq x_4 \leq \frac{1250}{3} \wedge \\ & x_2 - x_4 \geq \frac{250}{3}; \\ \frac{9}{5}x_1 + \frac{69}{50}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1660 & \mathbf{m} = (4,2) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 400 \wedge \\ & x_1 + \frac{3}{5}x_4 \geq 650 \wedge \frac{250}{3} \leq x_2 \leq 500 \wedge \\ & x_3 + x_4 \geq \frac{1250}{3} \wedge x_4 \geq \frac{1250}{3} \wedge \\ & x_2 - x_4 \leq \frac{250}{3}; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 + \frac{4}{5}x_0^A - 1710 & \mathbf{m} = (1,2) \wedge x_0^A > 0 \wedge x_0^B = 0 \wedge \\ & x_1 \geq 700 - x_0^A \wedge x_2 \geq \frac{5}{3}x_0^A \wedge \\ & x_3 \geq 500 - \frac{5}{3}x_0^A \wedge x_4 \geq 0; \\ \frac{9}{5}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1710 & \mathbf{m} = (1,2) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \geq 700 \wedge \\ & \frac{5}{7}x_1 + x_2 \geq 500 \wedge \frac{5}{7}x_1 + x_3 \geq 1000 \wedge \\ & x_3 \geq 500 \wedge x_4 \geq 0; \\ \frac{12}{7}x_1 + \frac{3}{2}x_2 + \frac{9}{10}x_3 + \frac{3}{5}x_4 - 1650 & \mathbf{m} = (1,2) \wedge \boldsymbol{x_0} = (0,0) \wedge x_1 \leq 700 \wedge \\ & \frac{5}{7}x_1 + x_2 \geq 500 \wedge \frac{5}{7}x_1 + x_3 \geq 1000 \wedge \\ & x_3 \geq 500 \wedge x_4 \geq 0. \end{cases}$$

Having defined the Lyapunov function candidate for the specific system for all $\mathrm{x} \in \wp$, in next section it is used to derive a feedback controller which brings the system from an arbitrary x to the desired periodic orbit.

4.5 Derivation of the controller

The Lyapunov function candidate defined in previous section can be used to derive a controller which settles down the system from the initial condition to the desired periodic orbit. For each point $\mathrm{x} \in \wp$, first the set of allowed inputs is defined, which assures that the system remains in the feasible domain $\wp$. Depending on the initial system state in point $\mathrm{x} \in \wp$, the system can finish the setups, idling for a duration, processing the jobs or switch to another system mode. The best action is determined by minimizing $\dot{V}(\mathrm{x})$ over the set of allowed inputs that assure that the system remains in the feasible domain $\wp$. Since the minimizing input depends on the current state, a state feedback is obtained.

In this section, the conditions at which the system is allowed to switch to another mode are determined first. Then, a feedback controller is derived for $\mathrm{x} \in \wp$. Finally, this feedback controller needs to be extended to one which is defined for the entire state space.

Allowed switching actions

Before presenting the controller, the conditions at which the system is allowed to switch to another mode are determined. From the desired periodic system behavior, depicted in Figure 4.4, it is known that the system cyclically visits the modes $(4,3)$, $(4,2)$ and $(1,2)$. The system stays in a mode where both workstations are processing the jobs at maximal rate until a buffer becomes empty. Then, the system switches to another mode by performing a setup. These three modes and their actions are sketched in Figure 4.10.

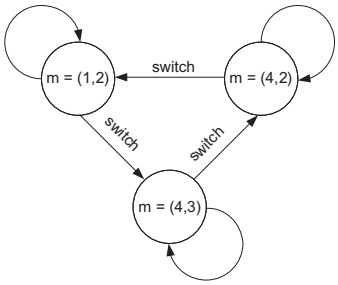


Figure 4.10: Overview of the system modes and their actions according to desired system behavior.

This overview is related to desired system behavior. For any arbitrary point $\mathrm{x} \in \wp$, there are maybe another switching actions possible which minimize $\dot{V}(\mathrm{x})$. These actions are only allowed if the system remains in the feasible domain $\wp$. In Table 4.3, another possible switching actions are presented. Also, the setup times needed for these switching actions and the setup times by definition of the feasible domain are presented.

Table 4.3: Another switching actions.

switch action	setups needed	setups by definition of $\wp$
mode (1,2) $\rightarrow$ mode (4,2)	$x_0^A = 50,\ x_0^B = 0$	$x_0^A = 0,\ x_0^B > 0$
mode (4,2) $\rightarrow$ mode (4,3)	$x_0^A = 0,\ x_0^B = 50$	$x_0^A > 0,\ x_0^B > 0$
mode (4,3) $\rightarrow$ mode (1,2)	$x_0^A = 50,\ x_0^B = 50$	$x_0^A > 0,\ x_0^B = 0$

All these possible switching actions are investigated further on their allowance. First consider the case that the system switches from mode $(1, 2)$ to mode $(4, 2)$. In this case, workstation A needs a setup to process the jobs for step 4 whereas workstation B can continu processing the jobs for step 2, i.e. $x_0^A = 50,\ x_0^B = 0$. According to the description of the feasible domain, the setup to mode $(4, 2)$ is characterized as $x_0^A = 0,\ x_0^B > 0$. If the system want to switch from mode $(1, 2)$ to mode $(4, 2)$, workstation A needs a setup but that is not allowed according to the description of the feasible domain. Therefore, it can be concluded that it is not allowed to switch from mode $(1, 2)$ to mode $(4, 2)$.
In case, when the system switches from mode $(4, 2)$ to mode $(4, 3)$, workstation B needs a setup to process the jobs for step 3 whereas workstation A can continu processing the jobs for step 4, i.e. $x_0^A = 0,\ x_0^B = 50$. According to the description of the feasible domain, the setup to mode $(4, 3)$ is characterized as $x_0^A > 0,\ x_0^B > 0$. Looking at both cases, it can be concluded that it is allowed to switch from mode $(4, 2)$ to mode $(4, 3)$. Maybe this switching action is not optimal, but workstation A can perform again a setup for step 4 which is allowed according to the description of the feasible domain.
In case, when the system switches from mode $(4, 3)$ to mode $(1, 2)$, workstations A and B need both a setup to process the jobs for another step, i.e. $x_0^A = 50,\ x_0^B = 50$. According to the description of the feasible domain, the setup to mode $(1, 2)$ is characterized as $x_0^A > 0,\ x_0^B = 0$. Looking at the setup for workstation B in both cases, it can be concluded that this switching action is not allowed. Workstation B needs a setup when the system want to switch from mode $(4, 3)$ to mode $(1, 2)$, but that is not allowed according to the description of the feasible domain.

Only the second action — switching from mode $(4, 2)$ to mode $(4, 3)$ — of Table 4.3 is allowed. But should this switching action be implemented in the controller design? This question can be answered by comparing the energy of the system for every case when the system operates in mode $(4, 2)$ with the energy of the system after switching to mode $(4, 3)$. The controller will only switch to another mode when the energy in the system decreases more than in current mode. In that way, the system converges to the desired periodic orbit. Therefore, if the energy of the system after switching to mode $(4, 3)$ is always higher than the energy of the system for every case when the system operates in mode $(4, 2)$, then this switching action can be neglected.

The energy of the system after switching from mode $(4, 2)$ to mode $(4, 3)$ can be determined with Lyapunov function candidate (4.2). After this setup, i.e. $x_0^A = 0$, the energy of the system becomes:

$$V_{43}(\mathrm{x}) = 1.8\ x_1 + 1.5\,(x_2 - 500) + 0.9\,(x_3 - 1000) + 0.6\ x_4. \tag{4.12}$$

The energy of the system for every case when the system operates in mode $(4, 2)$ is compared with (4.12). For each case, the Lyapunov function candidate is rewritten as:

$$V_{42}(\mathrm{x}) = V_{43}(\mathrm{x}) + \triangle V. \tag{4.13}$$

When the Lyapunov function candidate defined for each case — when the system operates in mode $(4, 2)$ — is rewritten in this form, the state with the smallest energy can be determined

by means of $\triangle V$. The system should operate in the state with the smallest energy in the system.

- $\triangle V > 0$: the system can better switch to mode $(4,3)$ instead of staying in mode $(4,2)$
- $\triangle V \leq 0$: the system can better stay in mode $(4,2)$ instead of switching to mode $(4,3)$

First consider the case when the system is performing a setup to mode $(4,2)$. Lyapunov function candidate (4.5) is defined for this case and rewritten with (4.13) as:

$$\begin{aligned} V_{42}(\mathrm{x}) &= 1.8\left(x_1 - 400 + x_0^B\right) + 1.5\left(x_2 - 500\right) + 0.9\left(x_3\right) + 0.6\left(x_4 - \frac{1250}{3} - \frac{5}{3}x_0^B\right) \\ &= 1.8\ x_1 + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\ x_4 \\ &+ 1.8\ \left(x_0^B - 400\right) + 0.9\left(1000\right) - 0.6\left(\frac{5}{3}\ x_0^B + \frac{1250}{3}\right) \\ &= V_{43}(\mathrm{x}) - 70 + 0.8\ x_0^B. \end{aligned} \tag{4.14}$$

With $0 \leq x_0^B \leq 50$, $\triangle V$ is always negative. This means that the system can better continue setting up to mode $(4,2)$ instead of switching to mode $(4,3)$. The energy of the system when it is setting up to mode $(4,2)$ is always smaller than the energy of the system after switching to mode $(4,3)$.

For the case when the system is processing the jobs while the initial buffer contents are $x_2 \geq 500$ and $x_4 \geq \frac{1250}{3}$, i.e. *case* 1 in Table 4.2, Lyapunov function candidate (4.6) is defined for this case and rewritten with (4.13) as:

$$\begin{aligned} V_{42}(\mathrm{x}) &= 1.8\left(x_1 - 400\right) + 1.5\left(x_2 - 500\right) + 0.9\left(x_3\right) + 0.6\left(x_4 - \frac{1250}{3}\right) \\ &= 1.8\ x_1 + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\ x_4 \\ &- 1.8\left(400\right) + 0.9\left(1000\right) - 0.6\left(\frac{1250}{3}\right) \\ &= V_{43}(\mathrm{x}) - 70. \end{aligned} \tag{4.15}$$

In this case, $\triangle V$ is always negative. This means that the system can better continue processing the jobs in mode $(4,2)$ instead of switching to mode $(4,3)$.

For *case* 2 and *case* 5 in Table 4.2, Lyapunov function candidate (4.7) is rewritten with (4.13) as:

$$\begin{aligned} V_{42}(\mathrm{x}) &= 1.8\left(x_1 + \frac{3}{5}x_4 - 650\right) + 1.5\left(x_2 - x_4 - \frac{250}{3}\right) + 0.9\left(x_3 + x_4 - \frac{1250}{3}\right) \\ &= 1.8\ x_1 + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\ x_4 \\ &+ 1.8\left(\frac{3}{5}x_4 - 650\right) + 1.5\left(\frac{1250}{3} - x_4\right) + 0.9\left(x_4 + 500 + \frac{250}{3}\right) - 0.6\ x_4 \\ &= V_{43}(\mathrm{x}) - \frac{3}{25}\ x_4 - 20. \end{aligned} \tag{4.16}$$

With $x_4 \geq 0$, $\triangle V$ is always negative. This means that the system can better continue processing the jobs in mode $(4,2)$ instead of switching to mode $(4,3)$.

For *case* 3 and *case* 4 in Table 4.2, Lyapunov function candidate (4.8) is rewritten with (4.13) as:

$$\begin{aligned} V_{42}(\mathrm{x}) &= 1.8\left(x_1 + \frac{3}{5}x_2 - 700\right) + 0.9\left(x_2 + x_3 - 500\right) + 0.6\left(x_4 - x_2 + \frac{250}{3}\right) \\ &= 1.8\ x_1 + 1.5\left(x_2 - 500\right) + 0.9\left(x_3 - 1000\right) + 0.6\ x_4 \\ &+ 1.8\left(\frac{3}{5}x_2 - 700\right) + 1.5\left(500 - x_2\right) + 0.9\left(500 + x_2\right) + 0.6\left(\frac{250}{3} - x_2\right) \\ &= V_{43}(\mathrm{x}) - \frac{3}{25}\ x_2 - 10. \end{aligned} \tag{4.17}$$

With $x_2 \geq \frac{250}{3}$ in this mode, $\triangle V$ is always negative. This means that the system can better continue processing the jobs in mode $(4,2)$ instead of switching to mode $(4,3)$.

After these comparisons, it can be concluded that the switching action from mode $(4,2)$ to mode $(4,3)$ can be neglected. The energy of the system for each action in mode $(4,2)$ is always smaller than the energy of the system after switching to mode $(4,3)$. Therefore, it is better to stay in mode $(4,2)$ instead of switching to mode $(4,3)$. So, only the actions depicted in Figure 4.10 are allowed.

Feedback controller for all $\mathbf{x} \in \wp$

The allowed switching actions are explained in previous section. In this section, the conditions at which the system is allowed to switch to an other mode have to be determined. After that, the controller which makes use of these conditions can be derived.

All the conditions for the actions presented in Figure 4.10 can be determined with the Lyapunov function candidate for the system defined in previous section which assures that the system remains in the feasible domain.

- If the system operates in mode $(4,3)$, stay in this mode until buffer 3 is empty. Then switch to mode $(4,2)$.
- If the system operates in mode $(4,2)$, stay in this mode until either $x_2 \leq \frac{250}{3}$ or $x_4 = 0$. Then switch to mode $(1,2)$.
- If the system operates in mode $(1,2)$, stay in this mode until buffer 1 is empty. Then switch to mode $(4,3)$.

With this feedback description and with the Lyapunov function candidate, a controller can be designed which makes the system converge to a periodic orbit and which assures that the system remains in the feasible domain.

The designed feedback controller for all $\mathrm{x} \in \wp$ which makes use of the above-mentioned conditions can be described as:

- If the system operates in mode $(4, 3)$, process all the jobs at maximal rate at both workstations until $x_3 = 0$. Then switch to mode $(4, 2)$. According to the Lyapunov function candidate, the other buffer contents are then equal to $x_1 \geq 350$, $x_2 \geq 500$ and $x_4 \geq 500$.
- If the system operates in mode $(4, 2)$, process all the jobs at maximal rate at both workstations until either $x_2 \leq \frac{250}{3}$ or $x_4 = 0$. Then switch to mode $(1, 2)$. According to the Lyapunov function candidate, the other buffer contents are then equal to $x_1 \geq 650$, $x_2 \geq \frac{250}{3}$, $x_3 \geq \frac{1250}{3}$ and $x_4 \geq 0$.
- If the system operates in mode $(1, 2)$, process all the jobs for step 1 and 2 at maximal rate until $x_1 = 0$. Then switch to mode $(4, 3)$. According to the Lyapunov function candidate, the other buffer contents are then equal to $x_2 \geq 500$, $x_3 \geq 1000$ and $x_4 \geq 0$

The feedback controller which makes use of the above-mentioned conditions will make the system converge from any arbitrary point $\mathrm{x} \in \wp$ to a periodic orbit. Unfortunately, this is not always the desired periodic orbit. To make this more clear, consider the case where the above-mentioned conditions are used as a feedback for an initial condition where only the buffer contents of buffer 4 does not lie on the desired periodic orbit. Suppose that buffer 4 contains an extra amount of 1000 jobs. According to the desired periodic system behavior in Figure 4.4, lets start in mode $(4, 3)$. The initial condition is now $(x_1, x_2, x_3, x_4) = (0, 500, 1000, 1000)$ instead of $(0, 500, 1000, 0)$. Since the initial value of x_3 which determines the departure from mode $(4, 3)$ is the same as for the desired periodic orbit, the duration of mode $(4, 3)$ is the same. Next, since the initial value of x_2 is the same but the initial value of x_4 is higher, the departure from mode $(4, 2)$ is now determined by the event that $x_2 \leq \frac{250}{3}$. Therefore, the duration of mode $(4, 2)$ is also the same. Finally, since the initial value of x_1 is the same as for the desired periodic orbit, the duration of mode $(1, 2)$ is also the same. This shows that using the above-mentioned feedback the system will converge to a *translated* desired periodic orbit instead of to the desired periodic orbit. In this case, buffer 4 contains always at least the extra amount of 1000 jobs, whereas in the desired periodic orbit, buffer 4 contains maximal 500 jobs.

The same analysis holds for the case when $x_2 > \frac{250}{3}$ and $x_4 = 0$. In this case, the departure from mode $(4, 2)$ is determined by the event that buffer 4 becomes empty ($x_4 = 0$). The controller which uses the above-mentioned conditions will also converge the system for this case to a translated desired periodic orbit.

From these two cases, it can be concluded that the controller which makes use of the above-mentioned conditions will not always make the system converge to the desired periodic orbit. This means that the above-mentioned conditions have to be changed if the feedback controller has to make the system converge for every $\mathrm{x} \in \wp$ to the desired periodic orbit. The conditions for switching from mode $(4, 2)$ to mode $(1, 2)$ has to be changed: "The controller has to switch from mode $(4, 2)$ to mode $(1, 2)$ until *both* conditions — $x_2 \leq \frac{250}{3}$ *and* $x_4 = 0$ — are met". If

one of the two conditions is met, which was the case as considered above, the other workstation has to idle until both conditions are met. As a result, during that period $V(\mathrm{x})$ is actually increasing. While a controller has to be designed where the energy in the system is never increasing. Nevertheless, this temporary increase of $V(\mathrm{x})$ makes that later on, $V(\mathrm{x})$ can decrease more. With this adaptation, the controller will always make the system converge from the initial condition which lies in the feasible domain to the desired periodic orbit.

The feedback controller which makes use of the following conditions will always make the system converge from the initial condition which lies in the feasible domain to the desired periodic orbit.

- If the system operates in mode $(4,3)$, stay in this mode until buffer 3 is empty. Then switch to mode $(4,2)$.
- If the system operates in mode $(4,2)$, stay in this mode until both $x_2 \leq \frac{250}{3}$ and $x_4 = 0$ are met. Then switch to mode $(1,2)$.
- If the system operates in mode $(1,2)$, stay in this mode until buffer 1 is empty. Then switch to mode $(4,3)$.

With these conditions, the following feedback controller can be derived for $\mathrm{x} \in \wp$ where the input of this system is given by $(\boldsymbol{u_0}, \boldsymbol{u}) = (u_0^A, u_0^B, u_1, u_2, u_3, u_4)$.

$$(\boldsymbol{u_0}, \boldsymbol{u}) = \begin{cases} (❹, ❸, 0, 0, 0, 0) & \text{if } \mathbf{m} = (4,3),\ x_0^A > 0,\ x_0^A = x_0^B; \\ (④, ③, 0, 0, \mu_3, \mu_4) & \text{if } \mathbf{m} = (4,3),\ \boldsymbol{x_0} = (0,0),\ x_3 > 0; \\ (④, ❷, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,3),\ \boldsymbol{x_0} = (0,0),\ x_3 = 0; \\ (④, ❷, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ x_0^A = 0,\ x_0^B > 0,\ x_4 > 0; \\ (④, ②, 0, \mu_2, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 > \frac{250}{3},\ x_4 > 0; \\ (④, ②, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 \leq \frac{250}{3},\ x_4 > 0; \\ (④, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 > \frac{250}{3},\ x_4 = 0; \\ (❶, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 \leq \frac{250}{3},\ x_4 = 0; \\ (❶, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (1,2),\ x_0^A > 0,\ x_0^B = 0,\ x_2 \geq 0; \\ (①, ②, \mu_1, \mu_2, 0, 0) & \text{if } \mathbf{m} = (1,2),\ \boldsymbol{x_0} = (0,0),\ x_1 > 0; \\ (❹, ❸, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1,2),\ \boldsymbol{x_0} = (0,0),\ x_1 = 0. \end{cases}$$

The controller is derived for all $\mathrm{x} \in \wp$ which means that the controller can make the system converge from any initial condition which lies in this feasible domain to the desired periodic orbit. But the controller has to make the system converge from any arbitrary initial condition towards the desired periodic orbit. Therefore, the controller needs to be extended to one which is defined for the entire state space.

Feedback controller for all x

After having defined the feedback controller for all $\mathrm{x} \in \wp$, this controller has to be extended to one which is defined for the entire state space. If this extension has been made then the

controller will make the system converge from any arbitrary initial condition towards the desired periodic orbit.

It is possible that the initial condition says that there are no jobs in the system, i.e. $x_1 = x_2 = x_3 = x_4 = 0$. This initial condition lies outside the feasible domain for which the feedback controller is designed. For this case and for every another case with an initial condition which lies outside the feasible domain, the extended feedback controller has to make the system converge into the feasible domain. From that point, the designed feedback controller in previous section will make the system converge towards the desired periodic orbit. The convergence of the designed feedback controller for an initial condition which lies in the feasible domain is proven in next section. This proof is also valid for initial conditions which do not lie in the feasible domain. Only the moment — the k^{th} start of mode $(4, 3)$ — in which the system state reaches the feasible domain should be determined.

Lets consider the case with initially empty buffers. According to the feedback controller description in previous section, the controller will switch the system first immediately to another mode instead of finishing the setups. If the system starts initially in mode $(1, 2)$, the controller will switch the system immediately to mode $(4, 3)$ because buffer 1 is empty. But then, the controller will switch the system immediately to mode $(4, 2)$ due to the fact that buffer 3 is empty. The controller will switch the system then again immediately to mode $(1, 2)$ cause both conditions — $x_4 = 0$ and $x_2 \leq \frac{250}{3}$ — are met. But, the system operates now some time-units later in mode $(1, 2)$ for the second time $(k = 2)$, which means that buffer 1 is not empty anymore. It has received some jobs from the inter arrival rate λ. The controller stays now in mode $(1, 2)$ and the system will finish the setups to mode $(1, 2)$. After this setup has been completed, the buffer contents are $(x_1, 0, 0, 0)$ with $x_1 \geq 50$ and the system can process the jobs until buffer 1 is empty. Clearing buffer 1 takes $\frac{3}{7}x_1$ time-units and at the start of mode $(4, 3)$, the buffer contents become $(0, \frac{5}{7}x_1, \frac{5}{7}x_1, 0)$.
In case when the system is not started initially in mode $(1, 2)$, the system will also operate some time-units later in mode $(1, 2)$ where buffer 1 is not empty anymore. But the controller stays in this case in mode $(1, 2)$ which is visit for the first time instead of the second time when the system starts initially in mode $(1, 2)$. Although the controller will stay earlier in mode $(1, 2)$, the case when the system is started initially in mode $(1, 2)$ is considered for determining the k^{th} start of mode $(4, 3)$ in which the system state reaches the feasible domain. For that case, the proof of the convergence of the designed feedback controller is always valid for any arbitrary initial condition.
To achieve desired system behavior, the system has to visit the modes $(4, 3)$, $(4, 2)$ and $(1, 2)$ cyclically. The condition for leaving mode $(4, 3)$ and mode $(1, 2)$ are depending on only one buffer contents which can always be satisfied. However, the condition for leaving mode $(4, 2)$ — when both $x_4 = 0$ and $x_2 \leq \frac{250}{3}$ are met — depends on two buffer contents which can not always be satisfied at the right moment of time. For example, it is possible that $x_2 \leq \frac{250}{3}$ and buffer 4 becomes empty during the setup from mode $(4, 3)$ to mode $(4, 2)$. According to the designed feedback controller, the system switches then to mode $(1, 2)$ which is not allowed. As already mentioned in section 4.5, the system has to finish the setup to mode $(4, 2)$ instead of switching immediately to mode $(1, 2)$ during this setup. To exclude this case during the convergence proof, the buffer contents of buffers 2 and 3 should satisfy the condition $\max(x_2, \frac{1}{2}x_3) > \frac{250}{3}$ at the start of mode $(4, 3)$. Workstation B only processes the jobs for step 3 in mode $(4, 3)$ and at the end of mode $(4, 3)$, when buffer 3 becomes empty, buffer 4 contains $\frac{1}{2}x_3$ jobs. When this condition is satisfied at the start of mode $(4, 3)$, buffer

2 and/or buffer 4 contains at the end of mode $(4,3)$ which is the start of mode $(4,2)$ more than $\frac{250}{3}$ jobs. In that case, the condition for leaving mode $(4,2)$ will never satisfied during the setup from mode $(4,3)$ to mode $(4,2)$. If the condition $\max(x_2, \frac{1}{2}x_3) > \frac{250}{3}$ on the buffer contents of buffers 2 and 3 has to fulfilled at the start of mode $(4,3)$, the condition on x_2 at the start of mode $(4,3)$ should be at least $\frac{5}{7}x_1 > \frac{250}{3}$. This means that workstation A has to process at least 117 jobs for step 1 which is satisfied when mode $(4,3)$ is started for the third time ($k \geq 3$). If mode $(4,3)$ is started for the second time, two setups which takes both 50 time-units are needed to start mode $(4,3)$ for the third time. With the process times in each mode and an inter arrival rate λ into buffer 1, workstation A has processed at the end of mode $(1,2)$ at least 117 jobs. Therefore, the convergence of the designed feedback controller for any arbitrary initial condition is proven in next section by considering $k \geq 3$.

It is also possible that, at the startup of the controller, the system is in mode $(1,3)$, which does not occur in the desired periodic orbit. In this case, the system has to switch to one of the three allowed modes. To determine the best choice, the energy of each mode after 50 time-units is compared. The one with the largest decrease in energy is taken as the switching action when the system is initial in mode $(1,3)$. with this choice, the controller will make the system converge faster towards the desired periodic orbit. In Table 4.4, all the possible switching actions and their buffer contents after 50 time-units are presented.

Table 4.4: Possible switching actions when the system is initially in mode (1,3).

switch action	buffer contents after setup and/or processing	buffer contents after setup and/or idling
mode (1,3) $\rightarrow$ mode (1,2)	$x_1 + 50 - \frac{500}{3},\ x_2 + \frac{500}{3},\ x_3,\ x_4$	$x_1 + 50,\ x_2,\ x_3,\ x_4$
mode (1,3) $\rightarrow$ mode (4,2)	$x_1 + 50,\ x_2,\ x_3,\ x_4$	
mode (1,3) $\rightarrow$ mode (4,3)	$x_1 + 50,\ x_2,\ x_3 - \frac{500}{3},\ x_4 + \frac{500}{3}$	$x_1 + 50,\ x_2,\ x_3,\ x_4$

First consider the case that the system switches from mode $(1,3)$ to mode $(1,2)$. In this case, workstation A can continu processing the jobs and workstation B is setting up. During this setup, jobs arrive into buffer 1 with an arrival rate λ of 1 $\left[\frac{\text{job}}{\text{time-unit}}\right]$. After 50 time-units, workstation A has processed $[50\mu_1 =]\frac{500}{3}$ jobs which are stored in buffer 2. The buffer contents of the buffers are denoted as $x_1 + 50 - \frac{500}{3},\ x_2 + \frac{500}{3},\ x_3,\ x_4$. The dissipated energy after 50 time-units for this mode is determined by $V(1,2,0,0,50 - \frac{500}{3}, \frac{500}{3}, 0, 0) = \frac{9}{5}(50 - \frac{500}{3}) + \frac{3}{2}(\frac{500}{3}) - 1710 = -1670$. But workstation A can also idling during this setup instead of processing the jobs for step 1. In that case, the buffer contents of the buffers after 50 time-units are denoted as $x_1 + 50,\ x_2,\ x_3,\ x_4$. The dissipated energy after 50 time-units for this case is determined by $V(1,2,0,0,50,0,0,0) = \frac{9}{5}(50) - 1710 = -1620$.
In case, when the system switches from $(1,3)$ to mode $(4,2)$, both workstations are performing a setup and no jobs can be processed. After 50 time-units, the buffer contents of the buffers are denoted as $x_1 + 50,\ x_2,\ x_3,\ x_4$ and the dissipated energy for this mode is determined by $V(4,2,0,0,50,0,0,0) = \frac{9}{5}(50) - 1720 = -1630$.
In case, when the system switches from $(1,3)$ to mode $(4,3)$, workstation A is setting up and workstation B can process the jobs for step 3. After 50 time-units, the buffer contents of the buffers are denoted as $x_1 + 50,\ x_2,\ x_3 - \frac{500}{3},\ x_4 + \frac{500}{3}$ and the dissipated energy for this mode is determined by $V(4,3,0,0,50,0,-\frac{500}{3}, \frac{500}{3}) = \frac{9}{5}(50) + \frac{9}{10}(-\frac{500}{3}) + \frac{3}{5}(\frac{500}{3}) - 1740 = -1700$. In this case, workstation B can idle during this setup instead of processing the jobs for step 3. In that case, the buffer contents of the buffers after 50 time-units are denoted as

$x_1 + 50,\ x_2,\ x_3,\ x_4$. The dissipated energy after 50 time-units for this case is determined by $V(4, 3, 0, 0, 50, 0, 0, 0) = \frac{9}{5}(50) - 1740 = -1650$.

The dissipated energy of each action after 50 time-units is presented in Table 4.5. The largest decrease in dissipated energy is obtained when the system switches from mode $(1, 3)$ to mode $(4, 3)$ and workstation B is processing the jobs for step 3 instead of idling during this setup. Therefore, if the system is initially in mode $(1, 3)$ in an arbitrary point $\mathrm{x} \notin \wp$, the system has to switch to the allowed mode $(4, 3)$ and process the jobs for step 3.

Table 4.5: Dissipated energy after 50 time-units.

switch action	after processing	after idling
mode (1,3) → mode (1,2)	−1670	−1620
mode (1,3) → mode (4,2)	−1630	
mode (1,3) → mode (4,3)	−1700	−1650

When this switching action will be added to the feedback controller in previous section, the extended feedback controller will always make the system converge from any initial system state to the desired periodic orbit. This proof is given in the next section.

For completeness, the feedback controller which will always make the system converge from any arbitrary x to the desired periodic orbit can be described as:

- If the system operates initially in mode $(1, 3)$, switch immediately to mode $(4, 3)$.
- If the system operates in mode $(4, 3)$, stay in this mode until buffer 3 is empty. Then switch to mode $(4, 2)$.
- If the system operates in mode $(4, 2)$, stay in this mode until both $x_2 \leq \frac{250}{3}$ and $x_4 = 0$ are met. Then switch to mode $(1, 2)$.
- If the system operates in mode $(1, 2)$, stay in this mode until buffer 1 is empty. Then switch to mode $(4, 3)$.

After determining the conditions under which the system is allowed to switch for an arbitrary point $\mathrm{x} \notin \wp$, the following feedback controller is derived for the entire state space. With an input of this system which is given by $(\boldsymbol{u_0}, \boldsymbol{u}) = (u_0^A, u_0^B, u_1, u_2, u_3, u_4)$, the feedback controller for all x is defined as:

$$(\boldsymbol{u_0}, \boldsymbol{u}) = \begin{cases} (❹, ❸, 0, 0, 0, 0) & \text{if } \mathbf{m} = (4,3),\ x_0^A > 0,\ x_0^A = x_0^B; \\ (④, ③, 0, 0, \mu_3, \mu_4) & \text{if } \mathbf{m} = (4,3),\ \boldsymbol{x_0} = (0,0),\ x_3 > 0; \\ (④, ❷, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,3),\ \boldsymbol{x_0} = (0,0),\ x_3 = 0; \\ (④, ❷, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ x_0^A = 0,\ x_0^B > 0,\ x_4 \geq 0; \\ (④, ②, 0, \mu_2, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 > \frac{250}{3},\ x_4 > 0; \\ (④, ②, 0, 0, 0, \mu_4) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 \leq \frac{250}{3},\ x_4 > 0; \\ (④, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 > \frac{250}{3},\ x_4 = 0; \\ (❶, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (4,2),\ \boldsymbol{x_0} = (0,0),\ x_2 \leq \frac{250}{3},\ x_4 = 0; \\ (❶, ②, 0, \mu_2, 0, 0) & \text{if } \mathbf{m} = (1,2),\ x_0^A > 0,\ x_0^B = 0,\ x_2 \geq 0; \\ (①, ②, \mu_1, \mu_2, 0, 0) & \text{if } \mathbf{m} = (1,2),\ \boldsymbol{x_0} = (0,0),\ x_1 > 0; \\ (❹, ❸, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1,2),\ \boldsymbol{x_0} = (0,0),\ x_1 = 0; \\ (❹, ❸, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1,3),\ x_0^A \geq 0,\ x_0^B > 0; \\ (❹, ③, 0, 0, \mu_3, 0) & \text{if } \mathbf{m} = (1,3),\ x_0^A \geq 0,\ x_0^B = 0,\ x_3 \geq 0. \end{cases}$$

4.6 Proof of convergence

Before simulations are executed with this designed feedback controller, the convergence of the controller is proven in this section.

It is known that the system cyclically visits the modes $(4,3)$, $(4,2)$ and $(1,2)$. Let $t_{43}^{(k)}$ denote the time at which the system started mode $(4,3)$ for the k^{th} time ($k \geq 1$) with buffer contents $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right)$ for buffers 1, 2, 3 and 4 respectively. Let $t_{42}^{(k)}$ and $t_{12}^{(k)}$ denote the time moment at which the system started mode $(4,2)$ respectively mode $(1,2)$ for the k^{th} time. The durations of these modes are then defined as $\tau_{43}^{(k)} = t_{42}^{(k)} - t_{43}^{(k)}$, $\tau_{42}^{(k)} = t_{12}^{(k)} - t_{42}^{(k)}$ and $\tau_{12}^{(k)} = t_{43}^{(k+1)} - t_{12}^{(k)}$ respectively.

If the convergence of the controller has to be proven, the buffer contents at which the system started mode $(4,3)$ for the k^{th} time with $k \to \infty$ should be equal to the buffer contents at the start of mode $(4,3)$ of the desired periodic orbit. Looking at Table 4.1, it needs to be shown that

$$\lim_{k\to\infty} \left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = (0, 500, 1000, 0). \tag{4.18}$$

From Table 4.1, it is also known that at the beginning of mode $(1,2)$, i.e. at $t = t_{12}^{(k)}$, buffer 4 is emptied ($x_4 = 0$) and $x_2 \leq \frac{250}{3}$. In mode $(1,2)$, workstation A needs to process at least 1000 jobs before the system can switch again to mode $(4,3)$ for the next time. This means that $x_1^{(k)} = 0$, $x_2^{(k)} \geq 500$ and $x_3^{(k)} \geq 1000$ at $t = t_{43}^{(k+1)}$ when the system operates in the feasible domain (i.e. $k \geq 3$). From these observations it follows that without loss of generality we can assume $x_1^{(k)} = 0$, $x_2^{(k)} \geq 500$, $x_3^{(k)} \geq 1000$, $x_4^{(k)} = 0$ by considering $k \geq 3$. Under these assumptions we would like to determine $x_2^{(k+1)}$ and $x_3^{(k+1)}$. When these buffer contents are determined, we can give the expressions for $k \to \infty$ which proves the convergence of the derived feedback controller towards the desired periodic orbit.

At $t_{43}^{(k)}$ — the start of entering mode $(4,3)$ for the k^{th} time (with $k \geq 3$) — the buffer contents are $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \left(0, x_2^{(k)}, x_3^{(k)}, 0\right)$. During mode $(4,3)$, first both workstations need a set up which takes 50 time-units. After this setup, i.e. at $t_{43}^{(k)} + 50$, only the buffer contents of buffer 1 is increased with rate λ. This results in $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \left(50, x_2^{(k)}, x_3^{(k)}, 0\right)$. If workstation B processes the jobs for step 3 at maximal rate, buffer 3 reduces at a rate of $\left(\mu_3 = \frac{1}{0.3} =\right) \frac{10}{3}$ jobs per time-unit. Therefore, clearing buffer 3 takes $\frac{3}{10}x_3^{(k)}$ time-units and mode $(4,3)$ has a duration of $\tau_{43}^{(k)} = 50 + \frac{3}{10}x_3^{(k)}$ time-units. During this mode, workstation A processes the jobs for step 4 at maximal rate μ_4, with $\mu_3 = 2\mu_4$. Therefore, at the beginning of mode $(4,2)$, i.e. at $t_{42}^{(k)} = t_{43}^{(k)} + \tau_{43}^{(k)}$, it is known that the buffer contents are equal to $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \left(50 + \frac{3}{10}x_3^{(k)}, x_2^{(k)}, 0, \frac{1}{2}x_3^{(k)}\right)$.
During mode $(4,2)$, workstation B needs first a setup which takes 50 time-units before it can process the jobs for step 2 at maximal rate. Workstation A continues processing the jobs for step 4 at maximal rate. At the end of this setup, workstation A has processed $50\mu_4 = \frac{250}{3}$ jobs and $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \left(100 + \frac{3}{10}x_3^{(k)}, x_2^{(k)}, 0, \max\left(\frac{1}{2}x_3^{(k)} - \frac{250}{3}, 0\right)\right)$. It is known that mode $(4,2)$ is ready when both conditions — $x_4 = 0$ and $x_2 \leq \frac{250}{3}$ — are met. Therefore, the duration of mode $(4,2)$ is defined by the duration of the condition which is reached as last. From the condition on x_4 in mode $(4,3)$, clearing buffer 4 takes $\tau_{42}^{(k)} \geq \frac{3}{5}\max\left(\frac{1}{2}x_3^{(k)} - \frac{250}{3}, 0\right)$ time-units. While clearing buffer 2 takes $\tau_{42}^{(k)} \geq \frac{3}{5}\left(x_2^{(k)} - \frac{250}{3}\right)$ time-units. So, the duration of mode $(4,2)$ takes $\tau_{42}^{(k)} = \max\left(\frac{3}{5}\left(x_2^{(k)} - \frac{250}{3}\right), \frac{3}{5}\max\left(\frac{1}{2}x_3^{(k)} - \frac{250}{3}, 0\right)\right)$ time-units. For $k \geq 3$, it is known that $x_1^{(k)} = 0, x_2^{(k)} \geq \frac{250}{3}, x_3^{(k)} \geq 0$ and $x_4^{(k)} = 0$. This means that the duration of mode $(4,2)$ takes $\tau_{42}^{(k)} = \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right) - 50$ time-units.
At the start of mode $(1,2)$, we have $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \Big(50 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right), \frac{250}{3}, x_2^{(k)} - \frac{250}{3}, 0\Big)$. In the beginning of this mode, workstation A needs a setup before it can process the jobs for step 1 which takes 50 time-units. During this setup, workstation B continues processing the jobs for step 2 and buffer 1 increases with rate λ. At the end of this setup (at $t_{12}^{(k)} + 50$), the buffer contents are $\left(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, x_4^{(k)}\right) = \Big(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right), 0, x_2^{(k)}, 0\Big)$. If workstation A processes the jobs for step 1 at maximal rate, buffer 1 effectively reduces at a rate of $(\mu_1 - \lambda =) \frac{7}{3}$ jobs per time-unit. Therefore, clearing buffer 1 takes $\frac{3}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)$ time-units, during which $\frac{10}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)$ jobs are being processed by workstation A. During this mode, workstation B processes the jobs for step 2 at maximal rate μ_2, with $\mu_1 = 2\mu_2$. Therefore at the end of this mode, buffer 2 contains $\frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)$ jobs and buffer 3 contains $x_2^{(k)} + \frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)$ jobs.

For $k \geq 3$, it follows that the buffer contents are:

$$\begin{aligned} x_1^{(k+1)} &= 0; \\ x_2^{(k+1)} &= \frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right); \\ x_3^{(k+1)} &= x_2^{(k)} + \frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right); \\ x_4^{(k+1)} &= 0. \end{aligned} \tag{4.19}$$

To prove the convergence of the controller, it needs to be shown that (4.18) holds when $k \to \infty$.

Before this proof is given, new variables are introduced to simplify the analysis. As mentioned earlier, it is assumed that for $k \geq 3$ the buffer contents are $x_1^{(k)} = 0$, $x_2^{(k)} \geq 500$, $x_3^{(k)} \geq 1000$ and $x_4^{(k)} = 0$. Instead of using the buffer values of buffers 2 and 3, another variables are introduced such that the value zero corresponds with these buffer contents.

$$\begin{aligned} y_2^{(k)} = \frac{1}{5}x_2^{(k)} - 100 \quad &\rightarrow \quad x_2^{(k)} = 5y_2^{(k)} + 500; \\ y_3^{(k)} = \frac{1}{10}x_3^{(k)} - 100 \quad &\rightarrow \quad x_3^{(k)} = 10y_3^{(k)} + 1000. \end{aligned}$$

With these new variables, $x_2^{(k)} = 500$ corresponds with $y_2^{(k)} = 0$ and $x_3^{(k)} = 1000$ corresponds with $y_3^{(k)} = 0$. Using these new variables for (4.19), the following equations for $y_2^{(k+1)}$ and $y_3^{(k+1)}$ are obtained:

$$\begin{aligned} y_2^{(k+1)} &= \frac{1}{5}x_2^{(k+1)} - 100 \\ &= \frac{1}{5}\left[\frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)\right] - 100 \\ &= \frac{1}{7}\left(100 + \frac{3}{10}\left[10y_3^{(k)} + 1000\right] + \max\left(\frac{3}{5}\left[5y_2^{(k)} + 500\right], \frac{3}{10}\left[10y_3^{(k)} + 1000\right]\right)\right) - 100 \\ &= \frac{1}{7}\left(3y_3^{(k)} + 3\max\left(y_2^{(k)}, y_3^{(k)}\right) + 700\right) - 100 \\ &= \frac{3}{7}y_3^{(k)} + \frac{3}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right) \\ &\leq \frac{3}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right) + \frac{3}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right) \\ &\leq \frac{6}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right). \end{aligned}$$

With $x_2^{(k)} \geq 500$ and $x_3^{(k)} \geq 1000$ at $t_{43}^{(k)}$ for $k \geq 3$, it is known that $y_2^{(k)} \geq 0$. This means that $y_2^{(k+1)}$ can also be written as:

$$0 \leq y_2^{(k+1)} \leq \frac{6}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right). \tag{4.20}$$

Also, for $y_3^{(k+1)}$:

$$\begin{aligned}
y_3^{(k+1)} &= \frac{1}{10}x_3^{(k+1)} - 100 \\
&= \frac{1}{10}\left[x_2^{(k)} + \frac{5}{7}\left(100 + \frac{3}{10}x_3^{(k)} + \max\left(\frac{3}{5}x_2^{(k)}, \frac{3}{10}x_3^{(k)}\right)\right)\right] - 100 \\
&= \frac{1}{10}\Bigg(\left[5y_2^{(k)} + 500\right] + \frac{5}{7}\Bigg(100 + \frac{3}{10}\left[10y_3^{(k)} + 1000\right] + \\
&\qquad\qquad \max\left(\frac{3}{5}\left[5y_2^{(k)} + 500\right], \frac{3}{10}\left[10y_3^{(k)} + 1000\right]\right)\Bigg)\Bigg) - 100 \\
&= \frac{1}{10}\left(5y_2^{(k)} + 500 + \frac{5}{7}\left(3y_3^{(k)} + 3\max\left(y_2^{(k)}, y_3^{(k)}\right) + 700\right)\right) - 100 \\
&= \frac{1}{2}y_2^{(k)} + \frac{3}{14}y_3^{(k)} + \frac{3}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right) \\
&\leq \frac{1}{2}\max\left(y_2^{(k)}, y_3^{(k)}\right) + \frac{3}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right) + \frac{3}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right) \\
&\leq \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right).
\end{aligned}$$

With $x_2^{(k)} \geq 500$ and $x_3^{(k)} \geq 1000$ at $t_{43}^{(k)}$ for $k \geq 3$, it is known that $y_3^{(k)} \geq 0$. This means that $y_3^{(k+1)}$ can also be written as:

$$0 \leq y_3^{(k+1)} \leq \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right). \tag{4.21}$$

With (4.20) and (4.21), we can determine $y_2^{(k+2)}$ and $y_3^{(k+2)}$:

$$0 \leq y_2^{(k+2)} \leq \frac{6}{7}\max\left(\frac{6}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right), \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right)\right) \leq \frac{6}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right), \tag{4.22}$$

$$0 \leq y_3^{(k+2)} \leq \frac{13}{14}\max\left(\frac{6}{7}\max\left(y_2^{(k)}, y_3^{(k)}\right), \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right)\right) \leq \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right). \tag{4.23}$$

From (4.22) and (4.23), it follows that:

$$0 \leq \max\left(y_2^{(k+2)}, y_3^{(k+2)}\right) \leq \frac{13}{14}\max\left(y_2^{(k)}, y_3^{(k)}\right). \tag{4.24}$$

From (4.24), we can conclude that:

$$\lim_{k\to\infty} y_2^{(k)} = \lim_{k\to\infty} y_3^{(k)} = 0. \tag{4.25}$$

With the fact that $y_2^{(k)} = 0$ corresponds with $x_2^{(k)} = 500$ and $y_3^{(k)} = 0$ corresponds with $x_3^{(k)} = 1000$, this last conclusion shows that (4.18) holds which proves the convergence of the controller towards the desired periodic orbit.

4.7 Simulation experiments

In previous sections, a non-distributed controller has been designed which will make the system converge from any arbitrary initial condition towards the desired periodic orbit. Also, it is proven that the derived feedback policy guarantees convergence of the system to the desired periodic orbit. In this section, the resulting responses of the controlled system are shown by performing simulations. As the non-distributed controller has been derived using continuous values of the buffer contents, first continuous simulations are performed to shown the stable closed-loop dynamics, as well as the convergence of the system to the desired system behavior. Also a discrete event model is presented, as in reality buffer contents can only have natural values. First, the results of a deterministic simulation are compared to the results of a continuous simulation. After that, a more interesting simulation is performed using stochastic system parameters. Finally, the improvements of this derived feedback policy is compared with other policies.

Continuous simulation

As the non-distributed controller has been designed using continuous values of the buffer contents, first continuous simulations are performed to shown the stable closed-loop dynamics, as well as the convergence of the system towards the desired periodic orbit. The continuous model is implemented in Matlab and is presented in Appendix A.1.

In the first simulation, a case with an empty system is considered and where the controller for workstation A and B is initiated in respectively "processing step 1" and "processing step 2". For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (1, 2, 0, 0, 0, 0, 0, 0)$, the resulting responses of the controlled system are shown. In Figure 4.11 and Figure 4.12, the resulting closed-loop dynamics, as well as the behavior of each workstation are presented. In Figure 4.13, the amount of work in the system and the weighted wip-level for this case are shown.

The simulation results of workstation A for this case are shown in Figure 4.11. When the graph in the right hand side of this figure is passed through like the periodic orbit of Figure 4.1 from light grey to black, then it is clear that the controller will make the behavior of workstation A converge towards the desired periodic orbit of Figure 4.1. Once workstation A operates according to this desired periodic orbit, workstation A reaches desired steady state behavior. This can be seen in the left hand side of Figure 4.11, where the buffer contents of buffers 1 and 4 are shown during this convergence.

The same analysis holds for workstation B, the controller will make the behavior of this workstation converge towards the desired periodic orbit of Figure 4.1 which is presented in the right hand side of Figure 4.12. Once workstation B operates according to this desired periodic orbit, workstation B also reaches desired steady state behavior. This can be seen in the left hand side of Figure 4.12, where the buffer contents of buffers 2 and 3 are shown during this convergence.

In Figure 4.13, simulation results of the amount of work and the weighted wip-level are shown. The amount of work in the system settles down exactly to the amount of work of the desired periodic system behavior, which is presented in Figure 4.4. When the system operates in

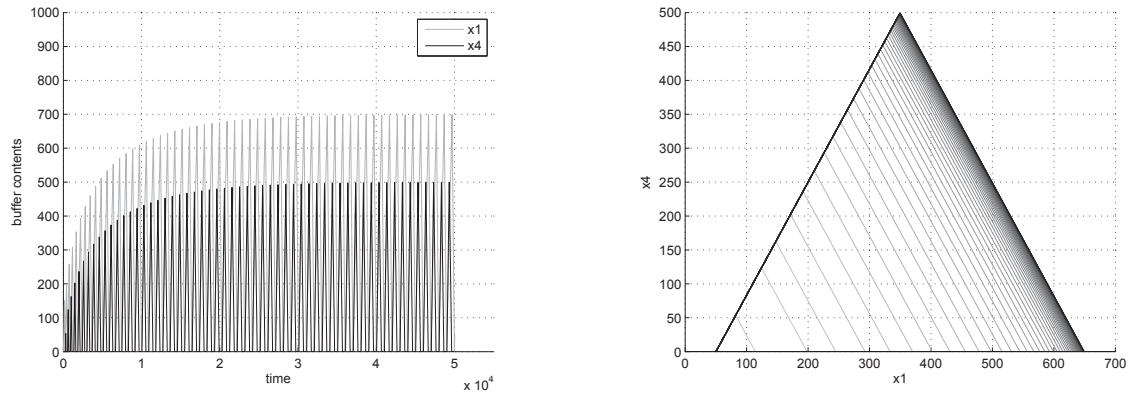


Figure 4.11: Continuous simulation results of workstation A for an empty system.

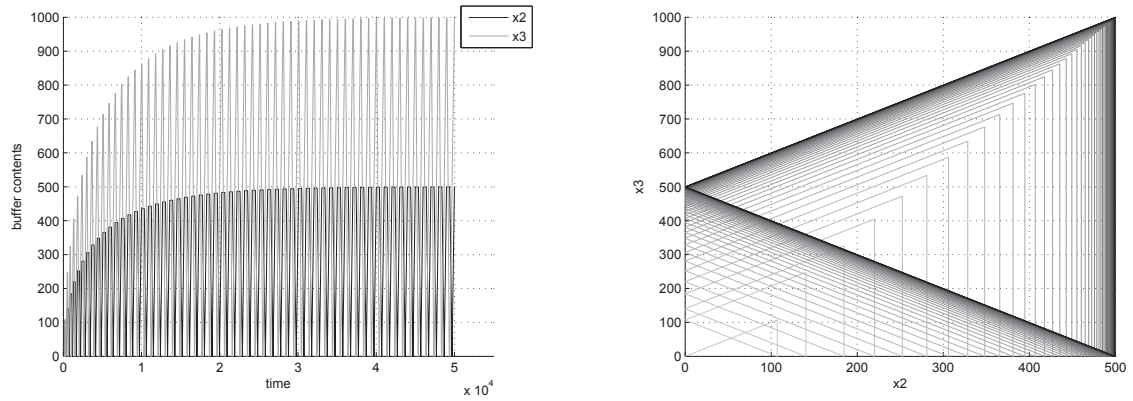


Figure 4.12: Continuous simulation results of workstation B for an empty system.

steady state, the mean amount of work in the system equals 1695 time-units, as should be the case (see (4.1)). Also, the weighted wip-level of the system settles down exactly to the weighted wip-level of the desired periodic system behavior. When the system operates in steady state, the weighted wip-level equals 3150, as should be the case (see (3.26)).

In the second simulation, a case with initially 1000 jobs in each buffer is considered and where the controller for workstation A and B is initiated in respectively "setting up for processing step 4" and "setting up for processing step 3". For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (4, 3, 50, 50, 1000, 1000, 1000, 1000)$, the resulting responses of the controlled system are shown. In Figure 4.14 and Figure 4.15, the resulting closed-loop dynamics, as well as the behavior of each workstation are presented. In Figure 4.16, the amount of work in the system and the weighted wip-level for this case are shown.

The controller will make the behavior of each workstation converge towards the desired periodic orbit of Figure 4.1. In steady state, the system operates according to desired periodic system behavior of Figure 4.4.

In Figure 4.16, simulation results of the amount of work and the weighted wip-level are presented. The amount of work in the system converges exactly to the amount of work of

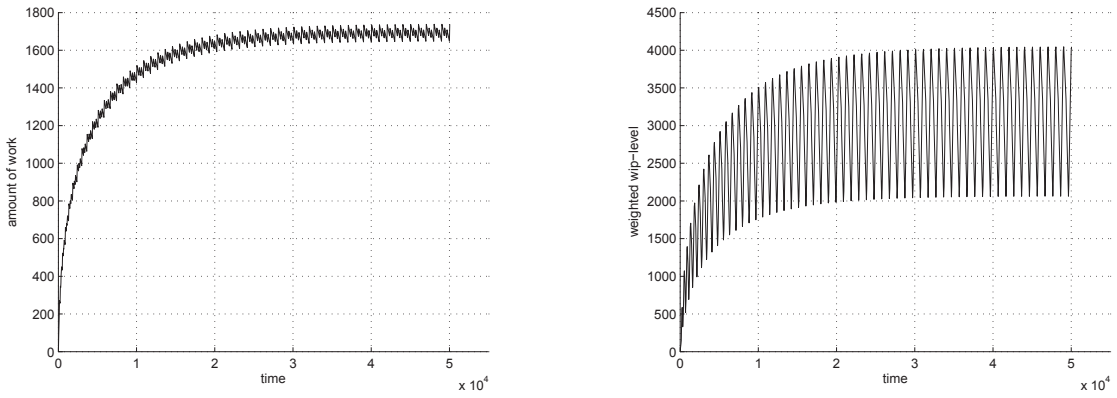


Figure 4.13: Continuous simulation results of amount of work and weighted wip-level for an empty system.

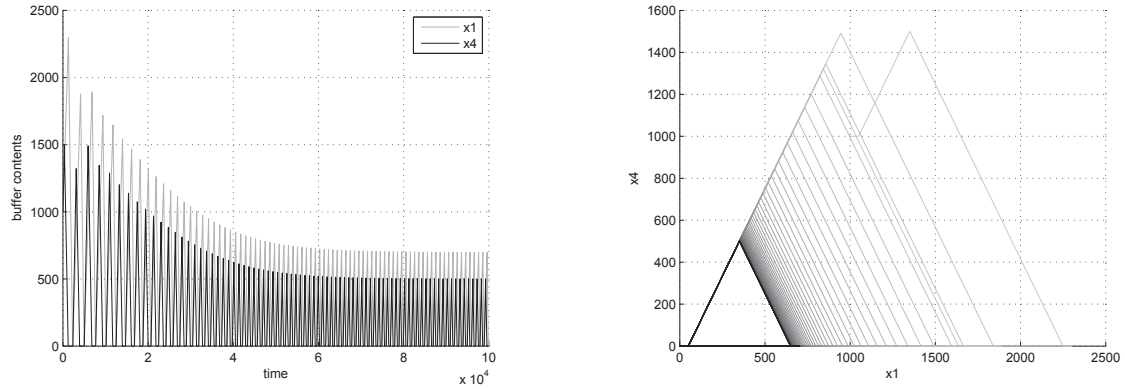


Figure 4.14: Continuous simulation results of workstation A for a 'full' system.

the desired periodic system behavior, which is presented in Figure 4.4. Also, the weighted wip-level of the system converges exactly to the weighted wip-level of the desired periodic system behavior. When the system operates in steady state, the mean amount of work equals 1695 time-units and the weighted wip-level equals 3150, as should be the case.

Discrete event simulation

As in reality buffer contents can only have natural values, in this section a discrete event model is presented. This discrete event model is made using the specification language χ. This χ-script is presented in Appendix A.2.

Before a more interesting simulation is performed with stochastic system parameters, first deterministic simulations are performed to make sure that this discrete event model is correct. This verification is made by comparing the results of the discrete event model with the results of the continuous model using the same deterministic system parameters. Although the same system parameters are used, some differences are possible due to the difference — continuous and discrete — in job-type flow. But, the resulting responses of the controlled system obtained with the discrete event model are the same as that with the continuous

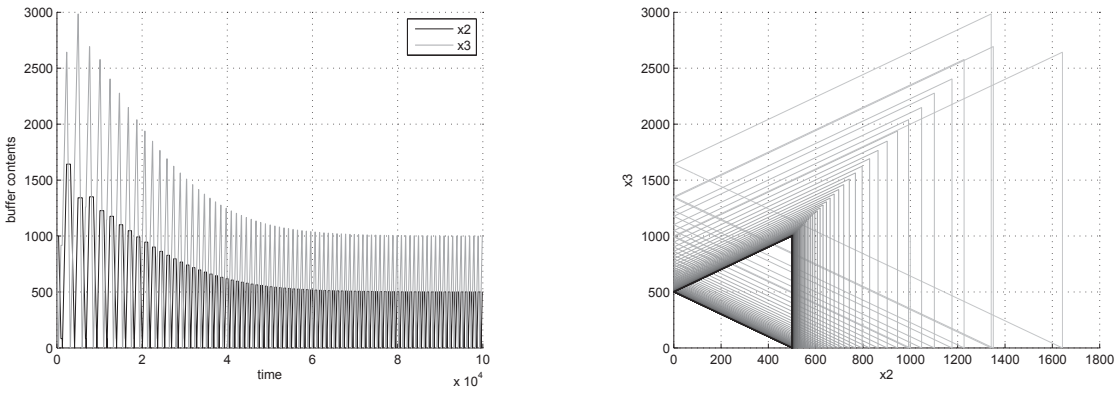


Figure 4.15: Continuous simulation results of workstation B for a 'full' system.

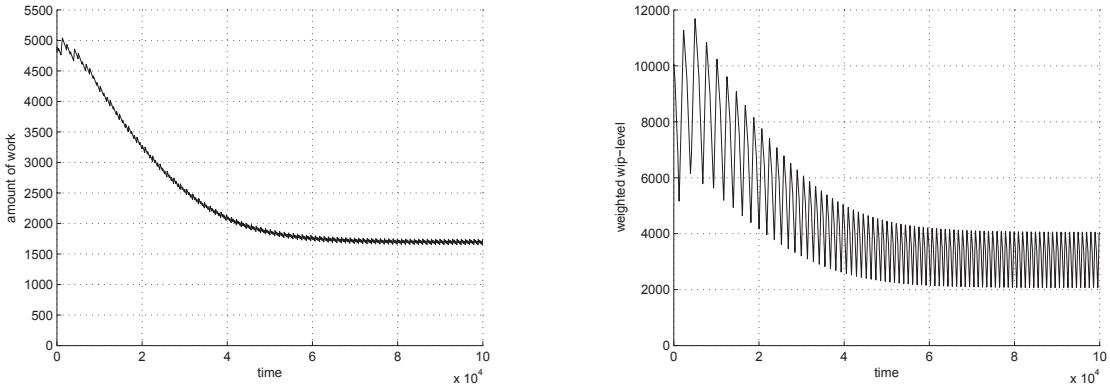


Figure 4.16: Continuous simulation results of amount of work and weighted wip-level for a 'full' system.

model which proves correctness of the discrete event model. For both models, the resulting responses of the controlled deterministic system with an initial system state $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (4, 2, 0, 50, 500, 500, 500, 500)$ are shown in Figure 4.17. The overall system behavior is the same for both models which proves correctness of the models.

After this verification, more interesting simulations can be performed using stochastic system parameters. All previous simulations are obtained with deterministic system parameters. In the next simulations, all the system parameters are not fixed but they are made stochastic by drawing them from independent exponential distributions. All the process times, setup times and the inter arrival time λ into buffer 1 are made stochastic. The process times for step 1 are drawn from an exponential distribution with mean 0.3, process times for step 2 are drawn from an exponential distribution with mean 0.6, process times for step 3 are drawn from an exponential distribution with mean 0.3 and the process times for step 4 are drawn from an exponential distribution with mean 0.6. Also, all the setup times are drawn from independent exponential distributions with mean 50 and the inter arrival times are drawn from an exponential distribution with mean 1. The case with initially a lot of jobs in the buffers — the case where each buffer contains initially 1000 jobs and where the controller for workstation A and B is initiated in respectively "setting up for processing step 4" and

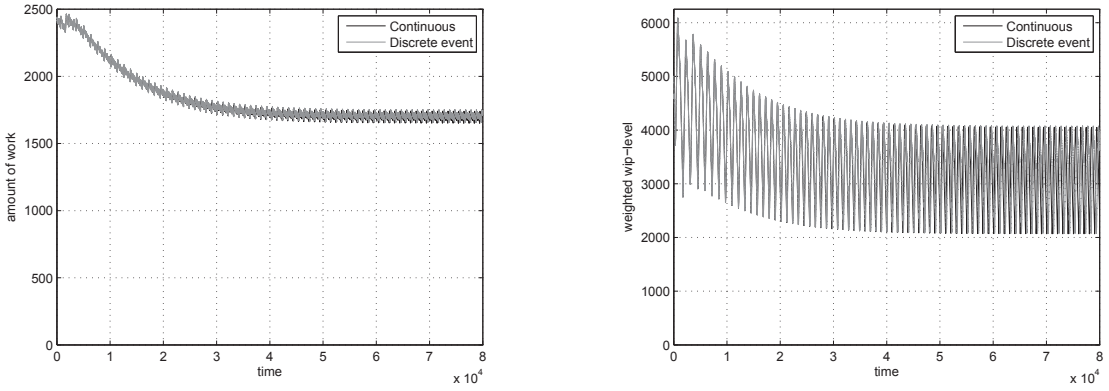


Figure 4.17: Comparison continuous model with discrete event model.

"setting up for processing step 3" — is considered again. For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (4, 3, 50, 50, 1000, 1000, 1000, 1000)$, a resulting response of the controlled system is given in Figure 4.18 and Figure 4.19.

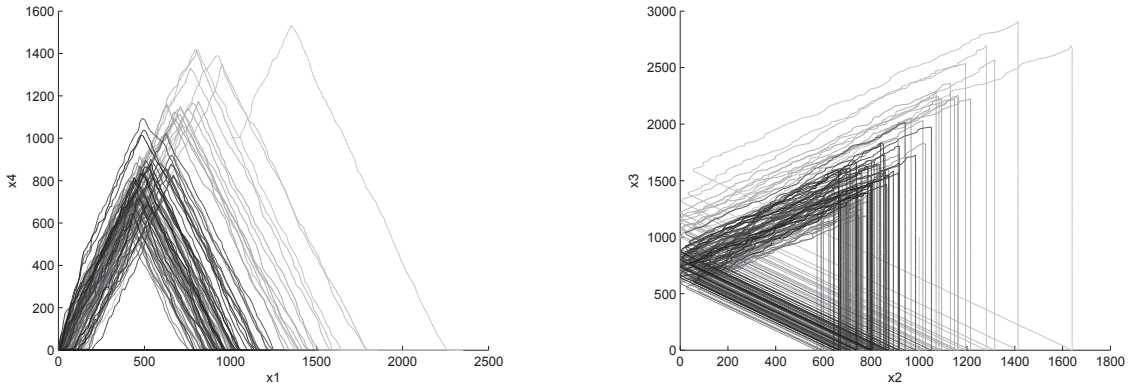


Figure 4.18: Discrete event simulation results of each workstation for a stochastic system.

Due to the stochastic system parameters, the designed controller can not exactly make the system converge towards the desired periodic orbit of Figure 4.1. The cycle period and the number of jobs present in the system vary and are not periodically which means that workstations are not always able to supply the buffers for the next step on time. As a consequence the cycle period and the mean number of jobs present in the system are likely to increase. While looking at Figure 4.19 this is the case, the mean amount of work and the weighted wip-level of the system in steady state are higher than the cases using deterministic system parameters. Although the system behavior varies over time due to this stochasticity, the designed controller will make the system converge to steady state which results in a system behavior with a small mean number of jobs in the system. This is proven in the next section, where the resulting responses of the derived feedback policy are compared with other policies.

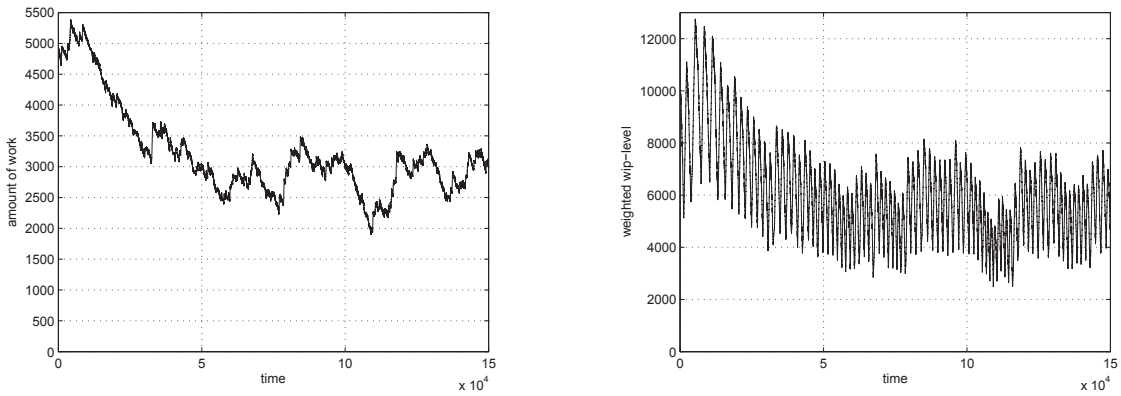


Figure 4.19: Discrete event simulation results of amount of work and weighted wip-level for a stochastic system.

4.8 Comparison of derived feedback policy with other policies

In the previous section, simulations are performed to check the designed non-distributed controller both for cases with deterministic as well as for cases with stochastic inter arrival times, process times and setup times. For cases with deterministic system parameters, we can conclude that the derived feedback policy will always make the system converge from any initial condition towards the desired periodic orbit which results in desired periodic system behavior with the smallest mean number of jobs in the system. This is proven in previous sections. Also for cases with stochastic system parameters, the derived feedback policy will make the system converge from any initial condition towards a periodic orbit which closes almost with the desired periodic orbit. That the derived feedback policy results in desired system behavior for stochastic cases is shown in this section by comparing the resulting responses of the controlled system with the derived feedback policy with other policies. Also, the improvements of this derived feedback policy are shown.

Clearing policies

As mentioned earlier, it was shown analytically that using a certain clearing policy results in an unstable reentrant system. Even though each machine in each workstation has enough capacity to process all jobs, $\frac{\lambda}{\mu_1} + \frac{\lambda}{\mu_4} < 1$ and $\frac{\lambda}{\mu_2} + \frac{\lambda}{\mu_3} < 1$, it has been shown in [KS90] and [PJK94] that since $\frac{\lambda}{\mu_2} + \frac{\lambda}{\mu_4} > 1$ and setup times are all positive, using a clearing policy — process the jobs in a buffer until it is empty, then switch to another buffer — for both workstations results in an unstable system. This instability is not only under continuous job-type flow but also if the job-types are discrete [PJK94]. A machine is processing jobs from a buffer too long, this results in starvation of other machines and therefore a waste of their capacity. Due to this waste the effective capacity of the other machines is not sufficient anymore, resulting in an unstable system. This observation has led to the development of so-called buffer regulators [PJK94] or gated policies which is presented in next section.

A system is unstable if the total number of jobs in the system explodes as time evolves. To make this more clear, Figure 4.20 and Figure 4.21 show the discrete event simulation results

of an unstable system controlled by a clearing policy with initial system state $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (4, 3, 50, 50, 1000, 1000, 1000, 1000)$.

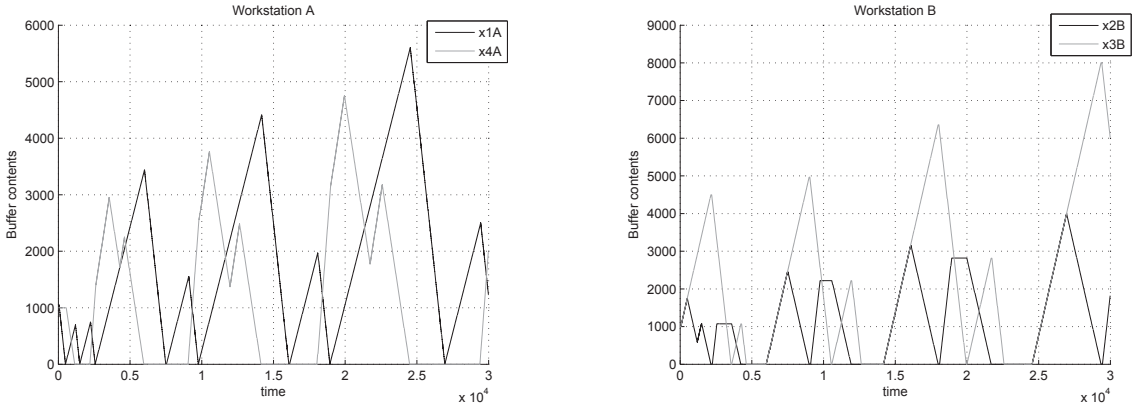


Figure 4.20: Discrete event simulation results of each workstation for an unstable system.

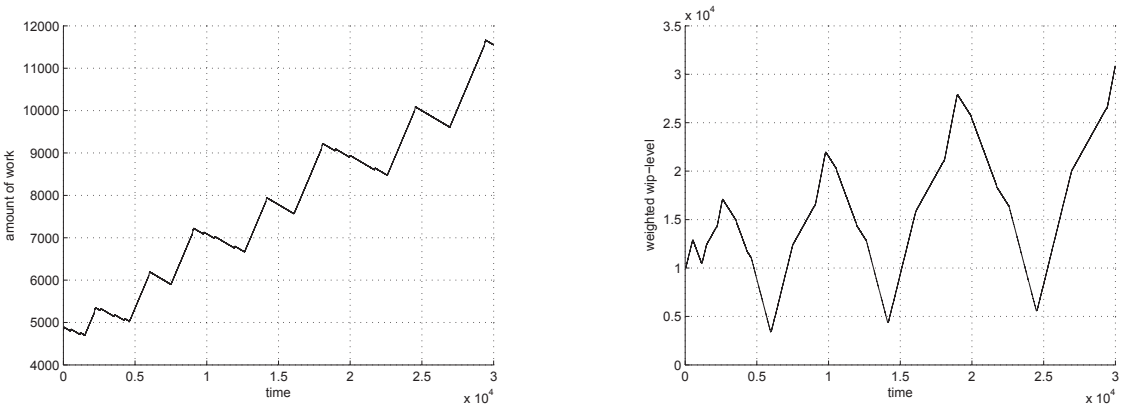


Figure 4.21: Discrete event simulation results of amount of work and weighted wip-level for an unstable system.

Looking at Figure 4.20, we can conclude that the total number of jobs in the system explodes as time evolves which results in an unstable system. In the next section, a gated policy is used which will control the system to a periodic orbit.

Gated policies

Gated policies or buffer regulators will make the system converge to steady state system behavior. In this section, the resulting responses of the controlled system with buffer regulators are compared with the results of the derived feedback policy. The main idea behind buffer regulators is that each buffer in the system contains a gate, so that the buffer is split into two parts. A part before the gate (a regulator buffer) and a part after the gate (a regulated buffer), see also Figure 4.22. Instead of switching depending on the total buffer contents in a buffer, switching is now determined based on the buffer contents after the gate. As a result, a workstation might now switch earlier — the regulated buffer is emptied while the regulator

buffer can contain some jobs — avoiding long periods of processing the jobs for one step which was the case when a clearing policy is used.

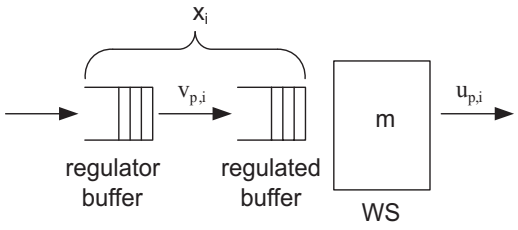


Figure 4.22: A regulator buffer constrains the input of jobs available to workstation m.

To create a stable system, the arrival rate of jobs ($v_{p,i}$) into the regulated buffer from the regulator buffer has to satisfy the following condition [PJK94].

$$\sum_{b(p,i)\in\{B_m\}} \frac{v_{p,i}}{u_{p,i}} < 1. \tag{4.26}$$

With: $b_{p,i}$ = job-type p stored in buffer i that works with workstation m,
B_m = set of buffers that works with workstation m,
$v_{p,i}$ = arrival rate of job-type p into the regulated buffer from the regulator buffer,
$u_{p,i}$ = process rate for job-type p of buffer i at workstation m.

In this project, the system only processes the jobs at maximal rate to obtain desired periodic system behavior. With an arrival rate $v_{p,i}$ equal to $\lambda = 1\left[\frac{\text{job}}{\text{time-unit}}\right]$, condition (4.26) is satisfied. For this case, simulations are performed to compare the results with the derived feedback policy.

With buffer regulators, the reentrant manufacturing system of Figure 2.1 becomes the system presented in Figure 4.23. Note that buffer 1 does not need a buffer regulator, the inter arrival rate λ into the buffer regulator of buffer 1 is the same as the arrival rate into the regulated buffer. The model of this system is made using the specification language χ. This χ-script is presented in Appendix A.3.

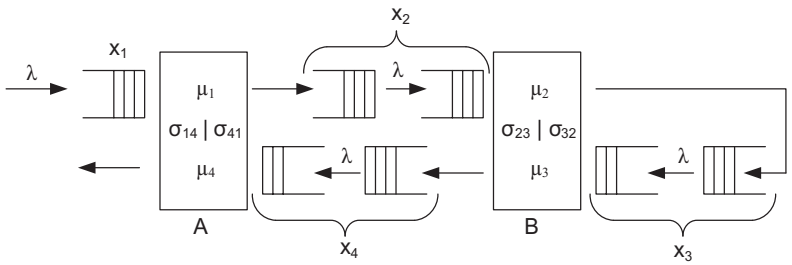


Figure 4.23: Reentrant manufacturing system with buffer regulators.

First, simulations with deterministic system parameters are performed. The results of the derived feedback policy is compared with the results of the buffer regulators. For both policies, a case with empty buffers is considered and where the controller for workstation A and B is

initiated in respectively "processing step 1" and "processing step 2". For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (1, 2, 0, 0, 0, 0, 0, 0)$, the resulting responses of the controlled system are shown in Figure 4.24.

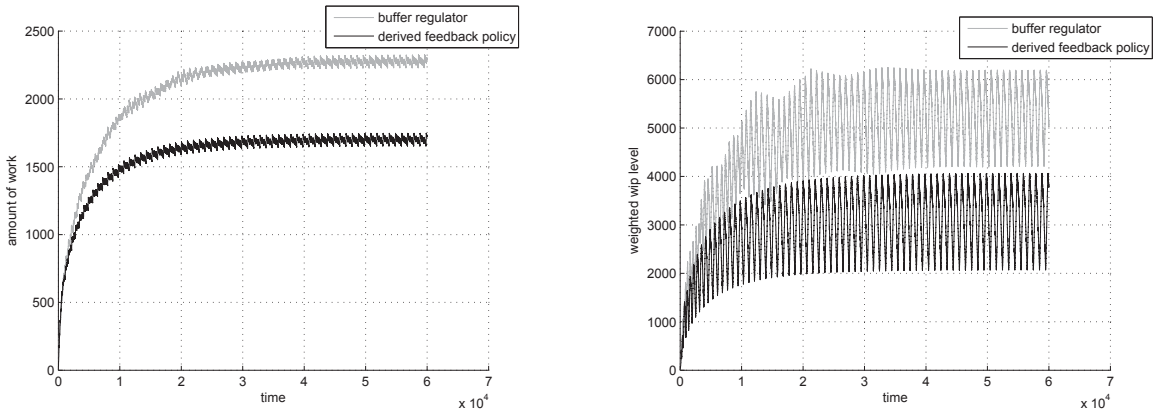


Figure 4.24: Discrete event simulation results of amount of work and weighted wip-level.

Looking at Figure 4.24, we can see that both policies will make the system converge towards a stable steady state system behavior. The derived feedback policy will make the system converge towards a steady state system behavior with a smaller mean amount of work in the system and a smaller weighted wip-level of the system than the case when buffer regulators are used. From this chapter, it is known that the derived feedback policy will make the system converge towards steady state system behavior. For buffer regulators, we only know that we can achieve stable steady state system behavior. But we can not guarantee that the resulting system behavior is desired. The conclusion is that the derived feedback policy guarantees convergence of the system towards periodic system behavior with the smallest mean number of jobs in the system.

In the next simulation, a case with stochastic system parameters and where each buffer contains initially 1000 jobs is considered. When buffer regulators are used, all these 1000 jobs are initially stored in the regulator buffer of each buffer. All process times, setup times and the inter arrival rate into buffer 1 are made stochastic by drawing them from independent exponential distributions. The controller for workstation A and B is initiated in respectively "setting up for processing step 4" and "setting up for processing step 3". For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (1, 2, 0, 0, 1000, 1000, 1000, 1000)$, the resulting responses of the controlled system are shown. In Figure 4.25, the amount of work in the system and the weighted wip-level for this case are shown.

Although the system behavior varies over time due to the stochasticity, both policies will make the system converge to a stable steady state system behavior. Looking at Figure 4.25, it can be concluded that the derived feedback policy will make the system converge towards a steady state system behavior with a smaller mean amount of work in the system and a smaller weighted wip-level of the system than the case when buffer regulators are used. From previous sections, it is known that the derived feedback policy will make the system converge towards steady state system behavior. For buffer regulators, it is not known when the system reaches steady state system behavior.

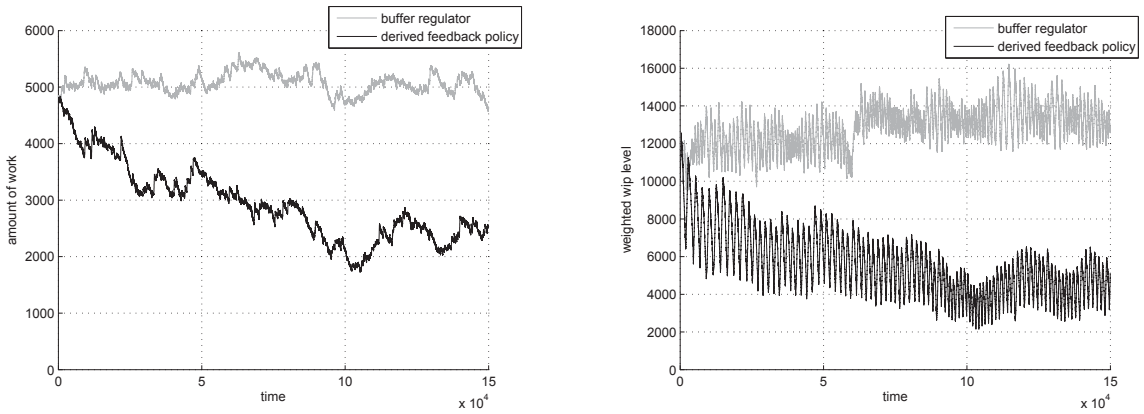


Figure 4.25: Discrete event simulation results of amount of work and weighted wip-level.

At the end of this chapter, we can conclude that the derived non-distributed controller guarantees convergence of the system towards desired periodic system behavior from any arbitrary initial condition.

Chapter 5

Distributed controller

In Chapter 4, a non-distributed controller is designed which will make the system converge from any arbitrary initial condition towards the desired periodic system behavior. This controller is non-distributed, each workstation needs to have global state information for determining when to switch. For manufacturing systems, this so-called "global policy" is feasible, maybe also for some urban traffic networks. However, for other networks, e.g. communication networks or computer networks, this global information might not be available. For these networks, a distributed controller is designed.
As already mentioned in the Introduction, this chapter deals with the third research objective, which was formulated as follows:

Research objective 3:
Determine a distributed controller (local policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

With this feedback controller, each workstation needs only local state information. In the specific manufacturing system, workstation A does not require information about the state at workstation B to determine its next task and workstation B does not require information about the state at workstation A. In this chapter, a distributed controller is designed which achieves a similar result as the non-distributed controller.

5.1 Derivation of the distributed controller

The non-distributed controller of previous chapter is taken as a starting point for deriving a distributed controller.

- If the system operates in mode $(1, 3)$, switch immediately to mode $(4, 3)$.
- If the system operates in mode $(4, 3)$, stay in this mode until buffer 3 is empty. Then switch to mode $(4, 2)$.
- If the system operates in mode $(4, 2)$, stay in this mode until either $x_2 \leq \frac{250}{3}$ and $x_4 = 0$. Then switch to mode $(1, 2)$.

- If the system operates in mode $(1, 2)$, stay in this mode until buffer 1 is empty. Then switch to mode $(4, 3)$.

Before a distributed controller is derived, first the above non-distributed controller description is defined for each workstation separately. After that, the description can be adapted to a distributed controller.

First, the non-distributed controller is described for each workstation separately. Lets consider workstation A. From the desired periodic system behavior in Figure 4.4, it is known that the system always processes the jobs at maximal rate and it cyclically visits the modes $(4, 3)$, $(4, 2)$ and $(1, 2)$. If workstation A is processing the jobs for step 4, it continues processing the jobs until both $x_2 \leq \frac{250}{3}$ and $x_4 = 0$. Then the controller switches workstation A to step 1. The controller description for step 4 of workstation A can be defined as: "If workstation A is processing the jobs for step 4, continue processing the jobs until both $x_2 \leq \frac{250}{3}$ and $x_4 = 0$. Then switch to step 1." If workstation A is processing the jobs for step 1, continue processing the jobs until buffer 1 is empty. The controller description for step 1 of workstation A can be defined as: "If workstation A is processing the jobs for step 1, continue processing the jobs until $x_1 = 0$. Then switch to step 4."
Lets consider now workstation B. If workstation B is processing the jobs for step 3, it continues processing the jobs until buffer 3 is empty. The controller switches then the workstation to step 2. The controller description for step 3 of workstation B can be defined as: "If workstation B is processing the jobs for step 3, continue processing the jobs until buffer 3 is empty. Then switch to step 2." If workstation B is processing the jobs for step 2, according to the non-distributed controller description, the controller will only switch again to step 3 when workstation A finishes step 1. The controller description for step 2 of workstation B can be defined as: "If workstation B is processing the jobs for step 2, continue processing the jobs until buffer 1 is empty. Then switch to step 3."

The non-distributed controller description for each workstation separately is defined as:

- Controller for workstation A:
  - If processing step 1, continue until $x_1 = 0$. Then switch to step 4.
  - If processing step 4, continue until both $x_4 = 0$ and $x_2 \leq \frac{250}{3}$. Then switch to step 1.
- Controller for workstation B:
  - If processing step 2, continue until $x_1 = 0$. Then switch to step 3.
  - If processing step 3, continue until $x_3 = 0$. Then switch to step 2.

Both workstations — step 4 of workstation A and step 2 of workstation B — need global state information. If a distributed controller is needed, the controller description should be adapted. Lets first consider workstation B. According to the proof of convergence in previous chapter, a distributed controller for workstation B exists. The controller needs to make sure that the system behavior still satisfies condition (4.19) for $k \geq 3$. This condition depends only on the buffer contents of buffers 2 and 3, which means that a distributed controller for

workstation B exists. According to the steady state system behavior in Figure 4.4, the system starts setting up to mode $(4,3)$. When the setup to mode $(4,3)$ has completed, workstation B processes all the jobs for step 3 until buffer 3 is empty. Then workstation B switches to step 2 and processes the jobs for step 2 until buffer 2 is empty. At the time moment that buffer 2 becomes empty, workstation A starts processing the jobs for step 1 at maximal rate. In this case, workstation B can continue processing the jobs for step 2 at maximal rate until buffer 1 is empty. With $\mu_1 = 2\mu_2$, workstation B processes the jobs two times slower than workstation A. This means that half of the number of processed jobs of step 1 is stored in buffer 2 and the other half of the number of processed jobs is stored in buffer 3. When $\bar{x}_1$ denotes the number of processed jobs for step 1 during a cycle period and $\bar{x}_2$ denotes the number of jobs in buffer 2 after the setup to step 2 has completed, then it is known that when workstation A finishes step 1, buffer 2 contains at least $\frac{1}{2}\bar{x}_1$ jobs and buffer 3 contains at least $\bar{x}_2 + \frac{1}{2}\bar{x}_1$ jobs. This means that workstation B finishes step 2 when at least $\bar{x}_2 + \frac{1}{2}\bar{x}_1$ jobs have been processed. Workstation B still need global state information. But according to the steady state system behavior in Figure 4.4, it is known that both workstations have to process at least 1000 jobs for every step during a cycle period. Therefore, workstation A finishes step 1 when $x_1 = 0$ and at least 1000 jobs have been processed. Also, workstation B has processed at least 1000 jobs for step 2 when workstation A finishes step 1. At the start of mode $(4,3)$, buffer 3 contains at least 1000 jobs. According to the desired system behavior, the number of processed jobs of step 1 is equal to the number of processed jobs of step 3 during a cycle period, i.e. $\bar{x}_1 = \bar{x}_3$. Therefore, the controller for step 2 of workstation B can be defined as "If processing step 2, continue until $\max\left(1000, \bar{x}_2 + \frac{1}{2}\bar{x}_3\right)$ have been processed. Then switch to step 3." If the system starts initially in mode $(4,2)$ or in mode $(1,2)$, then no jobs have been processed for step 3 and $\bar{x}_3 = 0$.

With the above analysis for step 2, the controller description for each workstation can be defined as:

- Controller for workstation A:
  - If processing step 1, continue until both $x_1 = 0$ and at least 1000 jobs have been processed. Then switch to step 4.
  - If processing step 4, continue until both $x_4 = 0$ and $x_2 \leq \frac{250}{3}$. Then switch to step 1.
- Controller for workstation B:
  - Let $\bar{x}_2$ denote the number of jobs in buffer 2 when the setup to step 2 has completed and let $\bar{x}_3$ denote the number of processed jobs for step 3 in previous mode. If processing step 2, continue until $\max(1000, \bar{x}_2 + \frac{1}{2}\bar{x}_3)$ jobs have been processed.
  - If processing step 3, continue until $x_3 = 0$. Then switch to step 2.

Note, that step 1 and step 2 are finished when at least 1000 jobs have been processed respectively. In case when $x_1 = 0$ and workstation A has not processed at least 1000 jobs for step 1, workstation A is waiting until buffer 1 receives a job from the inter arrival rate λ. This job is then send to workstation A, where it is processed at maximal rate. Workstation A repeats these actions until $x_1 = 0$ and at least 1000 jobs have been processed. Workstation A switches then to step 4. The same analysis holds for step 2 of workstation B. Whereas, step 3 and step

4 can not always process at least 1000 jobs during a cycle period. If the system operates in the feasible domain, both workstations will always process at least 1000 jobs for step 3 and step 4. But when the system operates outside the feasible domain, we can not guarantee that both workstations process at least 1000 jobs for steps 3 and 4. Consider the case that buffer 3 is empty and workstation B has not processed at least 1000 jobs. In this case, workstation B will never process at least 1000 jobs, because no jobs are entering buffer 3. The controller can better switch to step 2 instead of waiting until 1000 jobs have been processed which will never happen. Therefore, the controller of workstation B switches only to step 2 when buffer 3 is empty. In case when $x_4 = 0$, $x_3 = 0$ and workstation A has not processed at least 1000 jobs for step 4, then workstation A can better switch to step 1 instead of waiting for a long time until at least 1000 jobs have been processed.

Only step 4 of workstation A needs global state information. But this controller can also be implemented in a distributed way. Consider the case when $x_4 = 0$ and $x_2 > \frac{250}{3}$. According to the controller description, workstation A has to idle until $x_2 \leq \frac{250}{3}$. However, instead of first idling and then processing step 1, workstation A can also first switch to step 1, processing the jobs and then idle for the same duration. As long as workstation B finishes processing the jobs for step 2 at the same time as in the non-distributed controller description, condition (4.19) still holds. Therefore, the (possible) idle duration of workstation A can be shifted without changing the behavior of workstation B. According to the steady state system behavior in Figure 4.4, it is allowed to switch to step 1 when $x_4 = 0$ and $x_1 \geq 650$. In case when $x_4 = 0$ and $x_1 < 650$, workstation A idles at the end of step 4 until $x_1 \geq 650$. This idle duration is relatively short when the system operates in the feasible domain. In cases when the system operates outside the feasible domain, it is better to switch to step 1 instead of idling for a relatively long duration which results in a waste of capacity. For example, consider the case when the system is initially empty and the controller for workstation A is initiated in "processing step 4". Workstation A has to idle first for at least 650 time-units — after that $x_4 = 0$ and $x_1 \geq 650$ — before it can switch to step 1. For this case, workstation A can better switch to step 1, finishing the setup and processing the jobs. In that case, workstation B can also process the jobs for step 2, instead of idling until buffer 2 receives a job from step 1.
Therefore, workstation A has not only finished step 4 when $x_4 = 0$ and $x_1 \geq 650$. Step 4 is also finished when $x_4 = 0$ and workstation A has processed at least an amount of jobs for step 4. It is known that the system finishes mode $(4, 3)$ when buffer 3 is empty. Buffer 4 contains then at least $\frac{1}{2}\bar{x}_3$ jobs with $\bar{x}_3$ the number of processed jobs of step 3 during a cycle period. If workstation A has processed at least $\frac{1}{2}\bar{x}_3$ jobs for step 4, then workstation A continues processing the jobs for step 4 in mode $(4, 2)$ until buffer 4 is empty. Workstation A still need global state information. But according to desired system behavior, the number of processed jobs of step 1 is equal to the number of processed jobs of step 3 during a cycle period, i.e. $\bar{x}_1 = \bar{x}_3$. Therefore, the controller for step 4 of workstation A can be defined as "If processing step 4, continue until $x_4 = 0$ and either at least $\frac{1}{2}\bar{x}_1$ jobs have been processed or buffer 1 contains at least 650 jobs. Then switch to step 1." The controller for step 4 of workstation A is now a distributed controller, it contains only local state information.

The non-distributed controller in previous chapter is now implemented in a distributed way. Each workstation has a separate controller where both controllers need only local state information.

- Controller for workstation A:

- If processing step 1, continue until both $x_1 = 0$ and at least 1000 jobs have been processed. Then switch to step 4.
- Let $\bar{x}_1$ denote the number of processed jobs for step 1 in previous mode. If processing step 4, continue until $x_4 = 0$ and either at least $\frac{1}{2}\bar{x}_1$ jobs have been processed or buffer 1 contains at least 650 jobs. Then switch to step 1.

• Controller for workstation B:

- Let $\bar{x}_2$ denote the number of jobs in buffer 2 when the setup to step 2 has completed and let $\bar{x}_3$ denote the number of processed jobs for step 3 in previous mode. If processing step 2, continue until $\max(1000, \bar{x}_2 + \frac{1}{2}\bar{x}_3)$ jobs have been processed.
- If processing step 3, continue until $x_3 = 0$. Then switch to step 2.

In next section, simulations are performed to check the performance of the distributed feedback in closed-loop with the reentrant manufacturing system as depicted in Figure 2.1, both for deterministic as well as for stochastic system parameters.

5.2 Simulation experiments

The reentrant manufacturing system as depicted in Figure 2.1 in closed-loop with the distributed controller is modeled using the specification language χ. This discrete event model is presented in a χ-script in Appendix B.

First, discrete event simulations are performed to make sure that this discrete event model is correct. This verification is made by comparing the results of this discrete event model with the results of the continuous model in previous chapter using the same deterministic system parameters. In previous chapter, this verification was also made for the non-distributed controller. Although the same deterministic system parameters are used, some differences are possible. On the one hand, differences in job-type flow — continuous and discrete — are possible and on the other hand, differences in feedback — non-distributed and distributed — are possible. For both models, the resulting responses of the controlled deterministic system with an initial system state $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (4, 3, 50, 50, 1000, 1000, 1000, 1000)$ are shown in Figure 5.1.

From Figure 5.1, it is clear that both simulations start from the same initial system state and they will end up in the same steady state. Only, during the transient phase, i.e. during the convergence from the initial system state towards the steady state, the resulting responses of the controlled system are different. These differences are caused due to the difference in job-type flow and feedback. Although the transient phase of both models is not exactly the same, the overall system behavior is the same for both models which proves correctness of the discrete event model.

Looking better at the resulting responses of the controlled system in Figure 5.1, it is clear that the non-distributed controller from previous chapter will make the system converge faster towards the desired periodic system behavior than the case when the distributed controller is used. Also, the mean amount of work in the system and the weighted wip-level of the system during this convergence are smaller when the non-distributed controller is used. Not only for

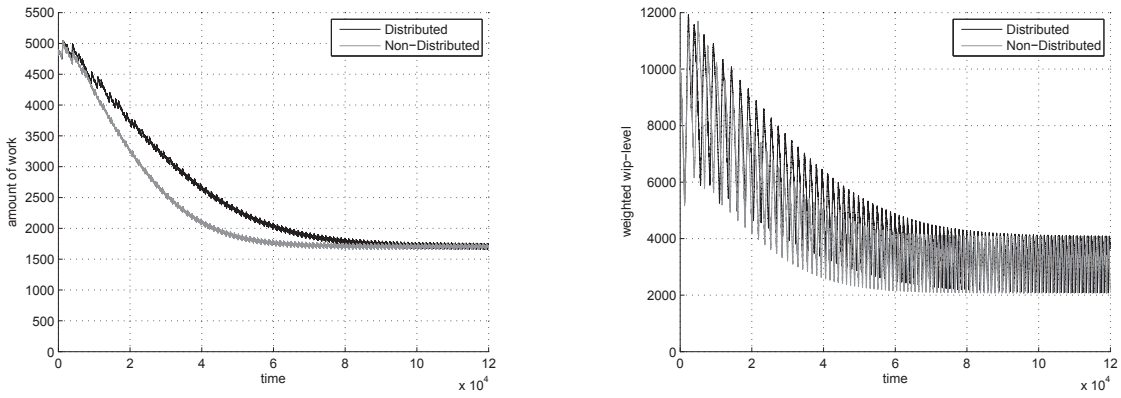


Figure 5.1: Simulation results: distributed feedback and non-distributed feedback.

this initial system state, but for any arbitrary initial system state these two conclusions can be drawn. This means that when the system can controlled by both feedbacks, a non-distributed controller is preferred.

After this verification, more interesting simulations can be performed using stochastic system parameters. Previous simulation results are obtained with deterministic system parameters. In next simulation, all the system parameters are not fixed but they are made stochastic by drawing them from independent exponential distributions. All the process times, setup times and the inter arrival time λ are made stochastic. The case with initially 1000 jobs in each buffer and where the controller for workstation A and B is initiated in respectively "processing step 1" and "processing step 2" is considered again. For this initial system state, i.e. for $(\mathbf{m}, \boldsymbol{x_0}, \mathbf{x}) = (m_A, m_B, x_0^A, x_0^B, x_1, x_2, x_3, x_4) = (1, 2, 0, 0, 1000, 1000, 1000, 1000)$, the resulting responses of the controlled system are given in Figure 5.2.

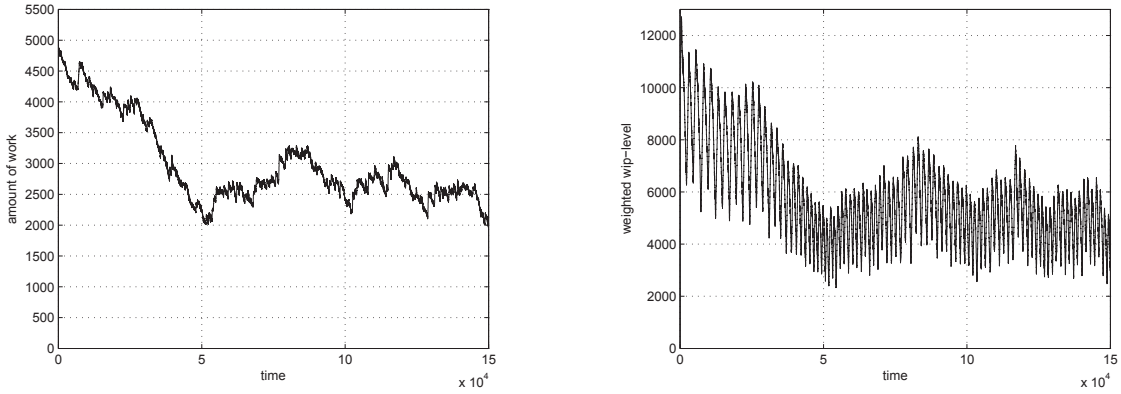


Figure 5.2: Discrete event simulation results of the system controlled by a distributed feedback.

Due to the stochastic system parameters, the designed controller can not exactly make the system converge towards the desired periodic orbit of Figure 4.1. The cycle period and the number of jobs present in the system vary and are not periodically which means that workstations are not always able to supply the buffers for the next step on time. As a consequence the cycle period and the mean number of jobs present in the system are likely

to increase. While looking at Figure 5.2 this is the case, the mean amount of work and the weighted wip-level of the system in steady state are higher than the cases using deterministic system parameters where the mean amount of work equals 1695 time-units and the weighted wip-level equals 3150. Although the system behavior varies over time due to this stochasticity, the designed controller will make the system converge towards steady state behavior with a small mean number of jobs in the system. In case when buffer regulators are used, the mean amount of work in the system (5000 time-units) and the weighted wip-level in the system (13000) are not reduced during the simulation. It remains constant instead of converging towards the desired periodic orbit, see also Figure 4.25.

The non-distributed controller in previous chapter has been implemented successfully in a distributed way. The distributed controller also guarantees convergence of the system from any arbitrary initial condition towards the desired periodic orbit, both for deterministic as well as stochastic system parameters.

Chapter 6

Conclusions and recommendations

In this project, the control of a reentrant manufacturing system with setup times as introduced by Kumar and Seidman is investigated. Most literature on this control problem has one thing in common: first a policy (or a class of policies) is proposed, and then the resulting behavior of the system under this policy (these policies) is considered. Sometimes the system behavior is optimized over the class of considered policies. A strength of these results is that they can be applied to general networks. A drawback however is that it is usually unclear if the presented policies result in optimal system behavior, or what to do to obtain prescribed or desired system behavior. Therefore, an other approach is followed in this project which guarantees desired system behavior. First, the desired system behavior is determined and then a feedback policy is *derived* which achieves this desired behavior. The way to derive this feedback policy has been illustrated extensively in this report. In this chapter, the conclusions of this project are summarized, based on the research objectives that have been formulated in Chapter 1.

The first research objective was formulated as follows:

Research objective 1:
Determine the desired periodic system behavior.

The approach followed in this project starts from desired system behavior. This implies that we first need to define this desired system behavior. For manufacturing systems, this would typically be behavior for which the mean amount of jobs in the system is minimal. From Little's law, it is known that this results in the smallest cycle period. After determining this minimal cycle period, several periodic cycles exist which minimizes the mean amount of jobs in the system. Therefore, we also consider the mean amount of work in the system. The amount of work or weight is associated with holding costs for storing a job in a buffer. In the specific reentrant manufacturing system, we are dealing with increasing weights which means that more value is added to the jobs after every step during a production cycle. For this specific reentrant system, an periodic cycle is determined with respect to minimal weighted work in progress level which minimizes the cycle period and the mean amount of jobs in the system. In Chapter 3, the desired behavior of the reentrant manufacturing system as introduced by Kumar and Seidman is determined and proven step by step.

The second research objective was formulated as follows:

Research objective 2:
Determine a non-distributed controller (global policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

After the desired periodic system behavior is determined, a feedback controller is designed — based on Lyapunov's direct method — which guarantees convergence of the system towards this desired behavior from any initial condition. First, the desired closed-loop behavior of the system is determined. Based on this given desired periodic orbit, an "energy" of the system can be defined by considering the mean amount of work in the system. Depending on the initial condition, either one or more translated desired periodic orbits can be obtained which go through this initial condition. The translated desired periodic orbit with the smallest mean amount of work in the system which go through this initial condition is used as a starting point for the controller. By controlling the system in a way that this "energy" is never increasing, the system stabilizes at a fixed energy level and the controller has made the system converge from the initial condition towards the desired periodic orbit. In Chapter 4, the way to derive this feedback controller from the given desired periodic orbit is illustrated extensively. Also, it is proved analytically, that this designed feedback controller guarantees convergence of the system towards the desired behavior from any initial condition. The derived feedback controller is implemented successfully in a continuous as well as discrete event simulation. After performing several simulations with different initial conditions, the resulting responses of the controlled system always show stable closed-loop dynamics as well as convergence towards the desired behavior. In comparison with buffer regulators and clearing policies, the derived feedback policy guarantees convergence of the system from any initial condition towards desired behavior with a small mean amount of jobs in the system.

The third research objective was formulated as follows:

Research objective 3:
Determine a distributed controller (local policy) which makes the specific manufacturing system converge towards the desired system behavior from any initial condition.

The designed controller is a non-distributed controller, each workstation needs to have global state information for determining when to switch. For manufacturing systems, this so-called "global policy" is feasible, maybe also for some urban traffic networks. However, for other networks, e.g. communication networks or computer networks, this global information might not be available. For these networks, the designed non-distributed controller is implemented successfully in a distributed way, i.e. such that each workstation only requires local state information for determining when to switch. The way to derive this distributed controller is presented in Chapter 5. The performance of the designed distributed controller is analyzed by means of a discrete event simulation. The resulting responses of the controlled system always show stable closed-loop dynamics as well as convergence towards the desired behavior.

In cases, when the system can be controlled by both derived controllers, the non-distributed controller is preferred above the distributed controller. Although, both derived controllers will make the system converge towards the same desired behavior, the non-distributed controller will make the system converge faster towards this desired behavior. Also, the weighted amount of work and the mean number of jobs in the system during the transient phase — the phase between the initial condition and the desired behavior — are smaller.

The fourth research objective was formulated as follows:

Research objective 4:
Check if the developed feedback controllers work for systems with stochastic system parameters by performing simulations.

Even though the feedback policies have been derived for deterministic systems, the derived feedback policies can also be applied in cases with stochastic system parameters. Instead of fixed system parameters, all process times, setup times and the inter arrival time into the first buffer are made stochastic by drawing them from independent exponential distributions. Also, for this case with stochastic settings, the performance of the derived controllers is analyzed by performing discrete event simulations. Due to the stochastic system parameters, the designed controller can not exactly make the system converge towards the desired periodic orbit. The cycle period and the number of jobs present in the system vary and are not periodically which means that workstations are not always able to supply the buffers for the next step on time. As a consequence the cycle period and the mean number of jobs present in the system are likely to increase. Although the system behavior varies over time due to this stochasticity, both feedback policies will make the system converge towards desired behavior with a small mean number of jobs in the system.

Recommendations

In this report, the feedback control strategy has only been applied successfully to a reentrant manufacturing system with setup times consisting of two workstations and processing a single job-type. In this section, some recommendations for further research will be defined.

Controller design

The desired periodic orbit which is used as a starting point for the controller design is derived using real buffer contents. In reality, buffer contents can only have natural values. This difference in job-type can result in small changes of the desired periodic orbit of the system. In further research, a desired periodic orbit of the system can be derived using buffer contents which can only have natural values.

Lyapunov function

The controller design is based on Lyapunov's direct method. Based on the derived desired periodic orbit, an "energy" of the system can be defined by considering the mean amount of work in the system. By continuously dissipating this energy, which is defined in the Lyapunov function candidate, the derived feedback controller guarantees convergence of the system to the desired periodic orbit in which the energy of the system stays constant. In further research, an other Lyapunov function candidate can be derived which makes the system converge faster to the desired periodic orbit.

Extended systems

The feedback control strategy is successfully applied to a reentrant manufacturing system consisting of two workstations. Also for systems consisting of a single workstation, this feedback control strategy is applied successfully [LR06a]. However, for systems consisting of three or more workstations, this feedback control strategy is not applied yet. In further research, this feedback control strategy can be applied to systems with several workstations. Furthermore, it is maybe possible to define a general method for these systems to make this feedback control strategy common applicable.

Bibliography

[BG96] S.X. Bai and S.B. Gershwin. Scheduling manufacturing systems with work-in-progress inventory control: Reentrant systems. *OR Spectrum*, 18(4):187–195, December 1996.

[Bis97] C.F.G. Bispo. Re-entrant flow lines. Technical report, Graduate school of industrial administration and the robotics institute, Carnegie Mellon University, Pittsburgh, PA 15213, September 1997. http://users.isr.ist.utl.pt/ cfb/thesis.pdf.

[DYZ97] J.G. Dai, D.H. Yeh, and C. Zhou. The QNET method for re-entrant queueing networks with priority disciplines. *Operation Research*, 45(4):610–623, July-Augustus 1997.

[ELR06] 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.

[KK01] S. Kumar and P.R. Kumar. Queueing network models in the design and analysis of semiconductor wafer fabs. *IEEE Transactions on robotics and automation*, 17(5):548–561, October 2001.

[KM95] P.R. Kumar and S.P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, 40(2):251–260, February 1995.

[KS90] P.R. Kumar and T.I. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35(3):289–298, March 1990.

[Kum93] P.R. Kumar. Re-entrant lines. *Queueing Systems: Theory and Applications: Special Issue on Queueing Networks*, 13:87–110, May 1993.

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

[LK91] S.H. Lu and P.R. Kumar. Distributed scheduling based on due dates and buffer priorities. *IEEE Transactions on Automatic Control*, 36(12):1406–1416, December 1991.

[LR06a] E. Lefeber and J.E. Rooda. Control of a reentrant manufacturing system with setup times: the Kumar-Seidman case. SE Report 2006-04, Eindhoven University of Technology, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven, The Netherlands, 2006. http://se.wtb.tue.nl/sereports.

[LR06b] E. Lefeber and J.E. Rooda. Controller design for switched linear systems with setups. *Physica A*, 363(1):48–61, February 2006.

[NK96] Y. Narahari and L.M. Khan. Performance analysis of scheduling policies in reentrant manufacturing systems. *Comput. Oper. Res.*, 23(1):37–51, January 1996.

[PJK94] J.R. Perkins, C. Humes Jr, and P.R. Kumar. Distributed scheduling of flexible manufacturing systems: Stability and performance. *IEEE Transactions on robotics and automation*, 10(2):133–141, 1994.

[Sav03] A.V. Savkin. Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems. *Proceedings of the 42nd IEEE conference on Decision and Control*, 5:5468–5471, December 2003.

[ULMV92] R. Uzsoy, C.Y. Lee, and L.A. Martin-Vega. A review of production planning and scheduling models in the semiconductor industry, part I. *IIE Transactions*, 24(4):47–60, September 1992.

Appendix A

Non-distributed Controller

A.1 Continuous Matlab-script

This script is made to perform continuous simulations of which the results are presented in Chapter 4. The script consists of a main script which invokes a system function and a save function.

Main script

The main script contains the non-distributed controller. Given an initial system state, this script determines the next time moment in which the state of the system, except the buffer contents, changes. In case when the system fulfills the conditions to switch to another system mode or finishes a setup, the state of the system changes. This time moment is sent as a time step with the current system state to the system function `delta.m`, which determines the new system state. This new system state contains the recalculated buffer contents and the remaining setup times which is sent back to the main script. Every time, when this new system state is sent back to the main script, this system state and the current time is sent to the save function `savedata.m`, which saves all the data. These data can be used to show the resulting responses or to analyze the performance of the controlled system. These actions are repeated for every time moment in which the system state, except the buffer contents, changes until the number of cycle periods is exceed.

```
lambda = 1;                                                % inter arrival rate

mu1 = 1/0.3;                                               % process time for step 1
mu4 = 1/0.6;                                               % process time for step 4
mu2 = 1/0.6;                                               % process time for step 2
mu3 = 1/0.3;                                               % process time for step 3

s41 = 50;                                                  % setup time to switch from step 4 to step 1
s14 = 50;                                                  % setup time to switch from step 1 to step 4
s32 = 50;                                                  % setup time to switch from step 3 to step 2
s23 = 50;                                                  % setup time to switch from step 2 to step 3

time = 0;                                                  % start time of the simulation

input('Give an initial system state for the reentrant manufacturing system with setup times');
modeA = input('modeA (type 1 for step 1 or 4 for step 4 ): ');  % Give an initial mode for workstation A
modeB = input('modeB (type 2 for step 2 or 3 for step 3 ): ');  % Give an initial mode for workstation B
tauA  = input('tauA: ');                                   % Give the initial setup for workstation A
tauB  = input('tauB: ');                                   % Give the initial setup for workstation B
```

```
x1A   = input('x1A: ');                                 % Give the initial buffer contents of buffer 1
x2B   = input('x2B: ');                                 % Give the initial buffer contents of buffer 2
x3B   = input('x3B: ');                                 % Give the initial buffer contents of buffer 3
x4A   = input('x4A: ');                                 % Give the initial buffer contents of buffer 4

state = [time modeA modeB x1A x4A x2B x3B];             % define the initial state

for k = 1:50                                            % define the number of cycle periods
    k = k + 1;

    if modeA == 1 && modeB == 3                         % if the system operates initially in mode (1,3), switch to mode (4,3)
        modeA = 4;
        tauA = s14;
    end

    if modeA == 4 && modeB == 3                         % if the system operates in mode (4,3), stay in this mode until buffer 3
        while x3B > 0                                   %   is empty
            if tauA == 0 && tauB == 0                   % if buffer 3 is not empty, finish the setups (if needed) and process
                timestep = x3B/mu3;                     %   the jobs at maximal rate until buffer 3 is empty
            elseif tauA == 0 && tauB > 0
                if x4A > 0
                    timestep = min(x4A/mu4, tauB);
                else
                    x4A = 0;
                    timestep = tauB;
                end
            elseif tauA > 0 && tauB > 0
                timestep = min(tauA, tauB);
            elseif tauA > 0 && tauB == 0
                timestep = min(tauA, x3B/mu3);
            end
            [ time x1A x4A x2B x3B tauA tauB ] = delta(time,modeA,modeB,x1A,x4A,x2B,x3B,tauA,tauB,timestep);
            [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B);
        end
        modeB = 2;                                      % if buffer 3 is empty, switch to mode (4,2)
        tauB = s32;
    end

    if modeA == 4 && modeB == 2                         % if the system operates in mode (4,2), stay in this mode until both
        while x2B > 250/3 || x4A > 0                    %   conditions (x4A=0 AND x2B<=250/3) are met
            if x2B > 250/3 && x4A > 0                   % in case when both conditions are not met, finish the setups (if needed)
                if tauA == 0 && tauB == 0               %   and process then the jobs at maximal rate until one of the two
                    timestep = min(x4A/mu4, (x2B-(250/3))/mu2); %   conditions is met
                elseif tauA == 0 && tauB > 0
                    timestep = min(x4A/mu4, tauB);
                elseif tauA > 0 && tauB > 0
                    timestep = min(tauA, tauB);
                elseif tauA > 0 && tauB == 0
                    timestep = min(tauA, (x2B-(250/3))/mu2);
                end
            elseif x2B <= 250/3 && x4A > 0              % in case when only x2B<=250/3 is met, finish the setups (if needed) and
                if tauA == 0 && tauB == 0               %   process then the jobs for step 4 at maximal rate, while
                    timestep = (x4A/mu4);               %   workstation B idles, until buffer 4 is empty
                    tauB = timestep;
                elseif tauA == 0 && tauB > 0
                    timestep = min(x4A/mu4, tauB);
                elseif tauA > 0 && tauB > 0
                    timestep = min(tauA, tauB);
                elseif tauA > 0 && tauB == 0
                    timestep = tauA;
                    tauB = timestep;
                end
            elseif x2B > 250/3 && x4A <= 0              % in case when only x4=0 is met, finish the setups (if needed) and process
                if tauA == 0 && tauB == 0               %   then the jobs for step 2 at maximal rate, while workstation A idles,
                    timestep = (x2B-(250/3))/mu2;       %   until x2<=250/3
                elseif tauA == 0 && tauB > 0
                    timestep = tauB;
                elseif tauA > 0 && tauB > 0
                    timestep = min(tauA, tauB);
                elseif tauA > 0 && tauB == 0
                    timestep = min(tauA, (x2B-(250/3))/mu2);
                end
            end
            [ time x1A x4A x2B x3B tauA tauB ] = delta(time,modeA,modeB,x1A,x4A,x2B,x3B,tauA,tauB,timestep);
            [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B);
        end
        modeA = 1;                                      % if both conditions (x4A=0 AND x2B<=250/3) are met, switch to mode (1,2)
        tauA = s41;
    end

    if modeA == 1 && modeB == 2                         % if the system operates in mode (1,2), stay in this mode until x1=0
        if x1A == 0                                     % in case when the system is initially empty, switch to mode (1,2)
            if x2B == 0 && x3B == 0 && x4A == 0
                tauA = 50;
                timestep = tauA;
                [ time x1A x4A x2B x3B tauA tauB ] = delta(time,modeA,modeB,x1A,x4A,x2B,x3B,tauA,tauB,timestep);
                [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B);
            else
                %
            end
```

```
        else
            while x1A > 0                                       % in case when x1>0, finish the setups (if needed) and process then
                if tauA == 0 && tauB == 0                       %   the jobs at maximal rate until buffer 1 is empty
                    timestep = (x1A/(mu1-lambda));
                elseif tauA == 0 && tauB > 0
                    timestep = min(x1A/(mu1-lambda), tauB);
                elseif tauA > 0 && tauB > 0
                    timestep = min(tauA, tauB);
                elseif tauA > 0 && tauB == 0
                    if x2B > 0
                        timestep = min(tauA, x2B/mu2);
                    else
                        timestep = tauA;
                    end
                end
                [ time x1A x4A x2B x3B tauA tauB ] = delta(time,modeA,modeB,x1A,x4A,x2B,x3B,tauA,tauB,timestep);
                [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B);
            end
        end
        modeA = 4;                                              % if buffer 1 is empty, switch to mode (4,3)
        modeB = 3;
        tauA = s14;
        tauB = s23;
        timestep = min(tauA,tauB);
        [ time x1A x4A x2B x3B tauA tauB ] = delta(time,modeA,modeB,x1A,x4A,x2B,x3B,tauA,tauB,timestep);
        [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B);
    end
end
```

System function delta.m

```
function [time x1A x4A x2B x3B tauA tauB] = delta(time, modeA, modeB, x1A, x4A, x2B, x3B, tauA, tauB, timestep)

lambda = 1;                                                     % inter arrival rate

mu1   = 1/0.3;                                                  % process time for step 1
mu4   = 1/0.6;                                                  % process time for step 4
mu2   = 1/0.6;                                                  % process time for step 2
mu3   = 1/0.3;                                                  % process time for step 3

time  = time + timestep;                                        % time moment of next event

mAmu1 = 0;                                                      % buffer 2 increases if workstation A is processing the jobs for step 1
mBmu2 = 0;                                                      % buffer 3 increases if workstation B is processing the jobs for step 2
mBmu3 = 0;                                                      % buffer 4 increases if workstation B is processing the jobs for step 3

if modeA == 1                                                   % jobs are always entering buffer 1 with the inter arrival rate lambda
    if tauA > 0                                                 % in case when workstation A is setting up to step 1
        x1A = x1A + lambda*timestep;
    elseif tauA == 0                                            % in case when the setup to step 1 is finished
        if x1A > 0                                              % in case when workstation A processes the jobs for step 1, the nett
           x1A = x1A + (lambda - mu1)*timestep;                 %   rate into buffer 1 is (lambda - mu1) until buffer 1 is empty
           mAmu1 = 1;
        elseif x1A == 0
           x1A = x1A + lambda*timestep;
        else
           x1A = 0;
        end
    end
else
    x1A = x1A + lambda*timestep;                                % in case when workstation A does not operate in mode 1
end

if modeB == 2
    if tauB > 0                                                 % if workstation B is setting up to step 2, the buffer contents of
        if mAmu1 == 1                                           %   buffer 2 only increases when workstation A processes the jobs
            x2B = x2B + mu1*timestep;                           %   for step 1 (mAmu1=1), otherwise buffer 2 remains the same
        else
            x2B = x2B;
        end
    elseif tauB == 0                                            % if the setup to step 2 has completed, buffer 2 will change
        if modeA == 4                                           % if the system operates in mode (4,2), the buffer contents of
            if x2B > 250/3                                      %   buffer 2 decreases until x2<=250/3
                x2B = x2B - mu2*timestep;
                mBmu2 = 1;
            else
                x2B = x2B;
            end
        else                                                    % if the system operates in mode (1,2), the buffer contents of
            if x2B > 0                                          %   buffer 2 increases when workstation A processes the jobs
                if mAmu1 == 1                                   %   step 1 (mAmu1=1), otherwise it decreases until x2=0
                    x2B = x2B + (mu1 - mu2)*timestep;
                    mBmu2 = 1;
                else
                    x2B = x2B - mu2*timestep;
                    mBmu2 = 1;
```

```
            end
        else                                        % in case when x2=0 and workstation A processes the jobs for
            if mAmu1 == 1                           %   step 1, workstation B continues processing the jobs for
                x2B = x2B + (mu1 - mu2)*timestep;   %   step 2, otherwise buffer 2 remains empty
                mBmu2 = 1;
            else
                x2B = 0;
            end
        end
    end
else                                                % in case when the system is not operating in step 2, the buffer
    if mAmu1 == 1                                   %   contents of buffer 2 only increases when workstation A is
        x2B = x2B + mu1*timestep;                   %   processing the jobs for step 1 (mAmu1=1)
    else
        x2B = x2B;
    end
end

if modeB == 3
    if tauB == 0                                    % if the setup to step 3 has completed, the buffer contents of
        if x3B > 0                                  %   buffer 3 decreases until buffer 3 is empty
            x3B = x3B - mu3*timestep;
            mBmu3 = 1;
        else
            x3B = x3B;
        end
    else                                            % if workstation B is setting up to step 3, the buffer contents
        x3B = x3B;                                  %   of buffer 3 remains constant until the setup has completed
    end
else
    if mBmu2 == 1;                                  % if the system is not operating in step 3, the buffer contents
        x3B = x3B + mu2*timestep;                   %   of buffer 3 only increases when workstation B is processing
    else                                            %   the jobs for step 2 (mBmu2=1), otherwise it remains constant
        x3B = x3B;
    end
end

if modeA == 4
    if tauA > 0                                     % if workstation A is setting up to step 4, the buffer contents
        if mBmu3 == 1                               %   of buffer 4 only increases when workstation B is processing
            x4A = x4A + mu3*timestep;               %   the jobs for step 3, otherwise it remains constant
        else
            x4A = x4A;
        end
    elseif tauA == 0                                % if the setup to step 4 has completed, the buffer contents of
        if x4A > 0                                  %   buffer 4 increases when workstation B is processing the
            if mBmu3 == 1                           %   jobs for step 3 with nett rate (mu3 - mu4), otherwise it
                x4A = x4A + (mu3 - mu4)*timestep;   %   decreases until buffer 4 is empty
            else
                x4A = x4A - mu4*timestep;
            end
        else                                        % in case when x4=0 and workstation B is processing the jobs for
            if mBmu3 == 1                           %   step 3, workstation A continues processing the jobs for step
                x4A = x4A + (mu3 - mu4)*timestep;   %   4 and buffer 4 increases with nett rate (mu3 - mu4),
            else                                    %   otherwise it remains empty
                x4A = 0;
            end
        end
    end
else                                                % if the system is not operating in step 4, the buffer contents
    if mBmu3 == 1                                   %   of buffer 4 only increases when workstation B is processing
        x4A = x4A + mu3*timestep;                   %   the jobs for step 3, otherwise it remains constant
    else
        x4A = x4A;
    end
end

if tauA == 0                                        % if workstation A is setting up, continue until the setup has
    tauA = 0;                                       %   been completed
elseif tauA > 0
    tauA = tauA - timestep;
end

if tauB == 0                                        % if workstation B is setting up, continue until the setup has
    tauB = 0;                                       %   been completed
elseif tauB > 0
    tauB = tauB - timestep;
end
```

Save function savedata.m

```
function [state] = savedata(state, time, modeA, modeB, x1A, x4A, x2B, x3B)

state = [state; time modeA modeB x1A x4A x2B x3B];  % for every time step, the current time, the system mode and the
                                                    %   buffer contents of each buffer is saved
```

A.2 Discrete event χ-script

This script is made to perform discrete event simulations of which the results are presented in Chapter 4.
The discrete event model of the specific reentrant manufacturing system is presented in Figure A.1. All the components and channels are explained in this appendix to clarify the functioning of the model.

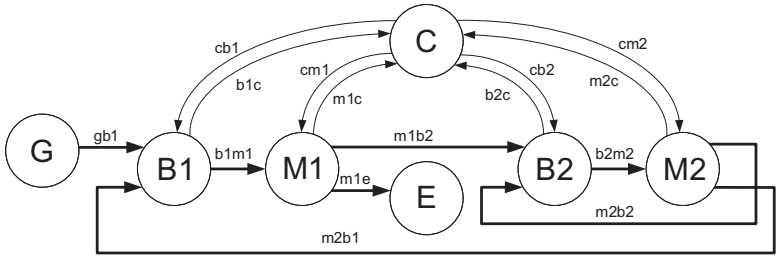


Figure A.1: Discrete event χ-model of the system controlled by a non-distributed controller.

Channels

The bold arrows in Figure A.1 represent the channels through which jobs are transferred. In this model, each job is identified with its arrival time into the system, i.e. `type job = real`. The thin arrows represent the channels through which the controller communicates with the components. The controller sends the initial buffer contents to the corresponding buffers through channels `cb1 and cb2`. If a workstation becomes available, it sends a signal to the controller through channel `m1c or m2c` to ask the controller for their next task. The controller will send a task `(act = nat # real)` through channel `cm1 or cm2` to the workstation in which the remaining setup is given or which step has to be processed at maximal rate. Every time when a job enters or leaves a buffer, the total number of jobs stored in the buffer is sent to the controller through channel `b1c or b2c`.

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

type job = real         // arrival time of a job into the system
   , act = nat # real   // action # remaining setup time
                        // action = 0 : setup, action = 1 : step 1 or step 3, action = 2 : step 2 or step 4
```

Controller

At startup of the model, the controller asks for an initial system state which has to be entered. After that, every initial buffer contents is sent to the corresponding buffer and the production cycle can start. If a workstation is available `(m1av or m2av is true)`, it asks the controller for a task. As the controller is always aware of the state of the system, it can decide what the workstation has to do. If the state satisfies the conditions to perform a setup, the controller tells the workstation to which step it must perform the setup, otherwise it tells a workstation which step has to processed at maximal rate until the conditions are satisfied. Sometimes, it

is possible that a workstation becomes available but it is not allowed to perform a setup to the next step. In that case, the workstation has to idle (`inloopm1 or inloopm2 is false`) for a duration until the conditions are satisfied to perform the setup.

```
proc C( cb1,cb2 : (!nat)^2, b1c,b2c : (?nat)^2, m1c,m2c : ?void
      , cm1,cm2 : !act, save : !file ) =
|[ modeA,modeB,x1A,x4A,x2B,x3B,x1Aold,x4Aold,x2Bold,x3Bold : nat
 , mode : nat # nat
 , s14A,s41A,s23B,s32B :-> real
 , tauA,tauB,s41,s14,s32,s23,Simtime : real
 , m1av,m2av,inloopm1,inloopm2 : bool
 | !"Give an initial system state \n"
 ; !"System mode workstation A (type 1 or 4): " ; ?modeA ; !"\n"
 ; !"System mode workstation B (type 2 or 3): " ; ?modeB ; !"\n"
 ; !"remaining setup time workstation A: " ; ?tauA ; !"\n"
 ; !"remaining setup time workstation B: " ; ?tauB ; !"\n"
 ; !"buffer contents x1A: " ; ?x1A ; !"\n"
 ; !"buffer contents x4A: " ; ?x4A ; !"\n"
 ; !"buffer contents x2B: " ; ?x2B ; !"\n"
 ; !"buffer contents x3B: " ; ?x3B ; !"\n"
 ; !"Simulation time: " ; ?Simtime ; !"\n"
 ; (cb1.0)!x1A ; (cb1.1)!x4A ; (cb2.0)!x2B ; (cb2.1)!x3B
 ; < x1Aold, x4Aold, x2Bold, x3Bold > := < x1A, x4A, x2B, x3B >
 ; mode:= < modeA, modeB >
 ; *[time <= Simtime -> inloopm1 := true
          ; inloopm2 := true
          ; [ true ; b1c.0?x1A -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                ; save! x1A, "\t", x4A, "\t", x2B, "\t", x3B, "\t", time, "\n"
                                ; x1Aold := x1A ; x4Aold := x4A ; x2Bold := x2B ; x3Bold := x3B
            | true ; b1c.1?x4A -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                ; save! x1A, "\t", x4A, "\t", x2B, "\t", x3B, "\t", time, "\n"
                                ; x1Aold := x1A ; x4Aold := x4A ; x2Bold := x2B ; x3Bold := x3B
            | true ; b2c.0?x2B -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                ; save! x1A, "\t", x4A, "\t", x2B, "\t", x3B, "\t", time, "\n"
                                ; x1Aold := x1A ; x4Aold := x4A ; x2Bold := x2B ; x3Bold := x3B
            | true ; b2c.1?x3B -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                ; save! x1A, "\t", x4A, "\t", x2B, "\t", x3B, "\t", time, "\n"
                                ; x1Aold := x1A ; x4Aold := x4A ; x2Bold := x2B ; x3Bold := x3B
            | true ; m1c?      -> m1av := true
                                ; s41A  := exponential(50.0) ; s41:=sample(s41A)
                                ; s14A  := exponential(50.0) ; s14:=sample(s14A)
            | true ; m2c?      -> m2av := true
                                ; s32B  := exponential(50.0) ; s32:=sample(s32B)
                                ; s23B  := exponential(50.0) ; s23:=sample(s23B)
            ]
            ; *[ m2av and inloopm2 -> [ mode = < 1 , 3 > -> mode := < 4 , 3 >
                                      ; tauA := s14
                                | mode = < 4 , 3 > -> [ x3B > 0 -> [ tauB > 0.0  -> cm2!< 0 , tauB >
                                                                   ; tauB := 0.0
                                                                   ; m2av := false
                                                     | tauB <= 0.0 -> cm2!< 2 , 0.0 >
                                                                   ; m2av := false
                                                     ]
                                                     | x3B = 0 -> mode := < 4 , 2 >
                                                                ; tauB := s32
                                                     ]
                                | mode = < 4 , 2 > -> [ x2B > 250/3  -> [ tauB > 0.0  -> cm2!< 0 , tauB >
                                                                          ; tauB := 0.0
                                                                          ; m2av := false
                                                                | tauB <= 0.0 -> cm2!< 1 , 0.0 >
                                                                          ; m2av := false
                                                                ]
                                                     | x2B <= 250/3 -> [ x4A = 0 -> mode := < 1 , 2 >
                                                                                  ; tauA := s41
                                                                       | x4A > 0 -> inloopm2 := false
                                                                       ]
                                                     ]
                                | mode = < 1 , 2 > -> [ x1A > 0 -> [ tauB > 0.0  -> cm2!< 0 , tauB >
                                                                   ; tauB := 0.0
                                                                   ; m2av := false
                                                     | tauB <= 0.0 -> cm2!< 1 , 0.0 >
                                                                   ; m2av := false
                                                     ]
                                                     | x1A = 0 -> [ x2B < 250/3 and x3B = 0 and x4A = 0 -> inloopm2 := false
                                                                  | x2B >= 0 -> mode := < 4 , 3 >
                                                                              ; < tauA , tauB > := < s14 , s23 >
                                                                  ]
                                                     ]
                                ]
               ]
            ; *[ m1av and inloopm1 -> [ mode = < 1 , 3 > -> mode := < 4 , 3 >
                                      ; tauA := s14
                                | mode = < 4 , 3 > -> [ x3B > 0 -> [ tauA > 0.0  -> cm1!< 0 , tauA >
                                                                   ; tauA := 0.0
                                                                   ; m1av := false
                                                     | tauA <= 0.0 -> cm1!< 2 , 0.0 >
                                                                   ; m1av := false
```

```
                                                ]
                                        | x3B = 0 -> mode := < 4 , 2 >
                                                  ; tauB := s32
                                                  ; inloopm1 := false
                                        ]
                                | mode = < 4 , 2 > -> [ x4A > 0 -> [ tauA > 0.0   -> cm1!< 0 , tauA >
                                                                                      ; tauA := 0.0
                                                                                      ; m1av := false
                                                                   | tauA <= 0.0  -> cm1!< 2 , 0.0 >
                                                                                      ; m1av := false
                                                                   ]
                                                       | x4A = 0 -> [ x2B <= 250/3 -> mode := < 1 , 2 >
                                                                                      ; tauA := s41
                                                                   | x2B >  250/3 -> inloopm1 := false
                                                                   ]
                                        ]
                                | mode = < 1 , 2 > -> [ x1A > 0 -> [ tauA > 0.0  -> cm1!< 0 , tauA >
                                                                                    ; tauA := 0.0
                                                                                    ; m1av := false
                                                                   | tauA <= 0.0 -> cm1!< 1 , 0.0 >
                                                                                    ; m1av := false
                                                                   ]
                                                       | x1A = 0 -> [ x2B < 250/3 and x3B = 0 and x4A = 0 -> cm1!< 0 , 1.0 >
                                                                                                             ; m1av := false
                                                                   | x2B >= 0 -> mode := < 4 , 3 >
                                                                              ; < tauA , tauB > := < s14 , s23 >
                                                                   ]
                                                       ]
                                ]
                ]
    ]
 ]|
```

Buffers B1 and B2

Both buffers can always receive jobs from the generator or workstation. Each buffer contains two lists where jobs from a step are stored respectively. At startup of the model, the initial buffer contents are stored in the corresponding buffers with arrival time 0.0 time-units. Every time when a job from a step enters or leaves a buffer, the buffer sends the total number of stored jobs of this step to the controller.

```
proc B1( cb1 : (?nat)^2, b1c : (!nat)^2, gb1,m2b1 : ?job, b1m1 : (!job)^2 ) =
|[ joblist1,joblist4 : job*
 , product : job
 , x1Aini, x4Aini : nat
 | joblist1 := []; joblist4 := []
 ; [true; (cb1.0)?x1Aini -> *[ x1Aini > 0 -> joblist1 := joblist1 ++ [ 0.0 ]; x1Aini := x1Aini - 1 ]]
 ; [true; (cb1.1)?x4Aini -> *[ x4Aini > 0 -> joblist4 := joblist4 ++ [ 0.0 ]; x4Aini := x4Aini - 1 ]]
 ; *[ true; gb1?product  -> joblist1 := joblist1 ++ [product] ; (b1c.0)!len(joblist1)
    | true; m2b1?product -> joblist4 := joblist4 ++ [product] ; (b1c.1)!len(joblist4)
    | len(joblist1) > 0; b1m1.0!hd(joblist1) -> joblist1:=tl(joblist1) ; (b1c.0)!len(joblist1)
    | len(joblist4) > 0; b1m1.1!hd(joblist4) -> joblist4:=tl(joblist4) ; (b1c.1)!len(joblist4)
    ]
]|

proc B2( cb2 : (?nat)^2, b2c : (!nat)^2, m1b2,m2b2 : ?job, b2m2 : (!job)^2 ) =
|[ joblist2,joblist3 : job*
 , product : job
 , x2Bini, x3Bini : nat
 | joblist2 := []; joblist3 := []
 ; [true; cb2.0?x2Bini -> *[ x2Bini > 0 -> joblist2 := joblist2 ++ [ 0.0 ]; x2Bini := x2Bini - 1 ]]
 ; [true; cb2.1?x3Bini -> *[ x3Bini > 0 -> joblist3 := joblist3 ++ [ 0.0 ]; x3Bini := x3Bini - 1 ]]
 ; *[ true; m1b2?product -> joblist2 := joblist2 ++ [product] ; b2c.0!len(joblist2)
    | true; m2b2?product -> joblist3 := joblist3 ++ [product] ; b2c.1!len(joblist3)
    | len(joblist2) > 0; b2m2.0!hd(joblist2) -> joblist2:=tl(joblist2) ; b2c.0!len(joblist2)
    | len(joblist3) > 0; b2m2.1!hd(joblist3) -> joblist3:=tl(joblist3) ; b2c.1!len(joblist3)
    ]
]|
```

Workstations M1 and M2

Every time when a workstation becomes available (`m1c! and/or m2c!`), it asks the controller for a task (`cm1?action and/or cm2?action`). Depending on this task, the controller tells the

workstation which step should be processed (`action.0 > 0 defines the step`) or to which step a setup (`action.0 = 0`) must be performed. If a job is processed at maximal rate, it is sent to the next buffer or to the exit.

```
proc M1( b1m1 : (?job)^2, m1b2,m1e : !job, m1c : !void, cm1 : ?act ) =
|[ action : act , product : job, proctime1, proctime4 :-> real
 | proctime1 := exponential(0.3)
 ; proctime4 := exponential(0.6)
 ; *[ true -> m1c!
             ; cm1?action
             ; [ action.0 = 0 -> delta action.1
               | action.0 = 1 -> b1m1.0?product
                                 ; delta sample(proctime1)
                                 ; m1b2!product
               | action.0 = 2 -> b1m1.1?product
                                 ; delta sample(proctime4)
                                 ; m1e!product
               ]
     ]
 ]|

proc M2( b2m2 : (?job)^2, m2b2,m2b1 : !job, m2c : !void, cm2 : ?act ) =
|[ action : act , product : job, proctime2, proctime3 :-> real
 | proctime2 := exponential(0.6)
 ; proctime3 := exponential(0.3)
 ; *[ true -> m2c!
             ; cm2?action
             ; [ action.0 = 0 -> delta action.1
               | action.0 = 1 -> b2m2.0?product
                                 ; delta sample(proctime2)
                                 ; m2b2!product
               | action.0 = 2 -> b2m2.1?product
                                 ; delta sample(proctime3)
                                 ; m2b1!product
               ]
     ]
 ]|
```

Generator

The generator will always send generated jobs with rate λ to the first list in buffer B1.

```
proc G( gb1 : !job ) =
|[ lambda :-> real, wait: real
 | lambda := exponential(1.0)
 ; *[ true -> gb1!time
             ; wait := sample lambda
             ; delta wait
    ]
]|
```

Exit

The exit E can always receive jobs from workstation m1. Every time when the exit has received a job, the job leaves the system.

```
proc E( m1e : ?job ) =
|[ product : job
 | *[ true; m1e?product -> skip ]
]|
```

Cluster

The cluster System connects all the components by channels as presented in Figure A.1.

```
clus System ( save : ! file ) =
|[ cb1, cb2, b1c, b2c : (-nat)^2
 , gb1, m1b2, m2b1, m2b2, m1e : -job
 , b1m1, b2m2 : (-job)^2
 , m1c , m2c : -void
 , cm1 , cm2 : -act
 |G ( gb1 )
||B1( cb1, b1c, gb1, m2b1, b1m1 )
||M1( b1m1, m1b2, m1e, m1c, cm1 )
||B2( cb2, b2c, m1b2, m2b2, b2m2 )
||M2( b2m2, m2b2, m2b1, m2c, cm2 )
||E ( m1e )
||C ( cb1, cb2, b1c, b2c, m1c, m2c, cm1, cm2, save )
]|

xper = |[ System( fileout( "Nondistributed_systemstates.txt" ) ) ]|
```

A.3 Buffer regulator

The discrete event model of the specific reentrant manufacturing system with buffer regulators is presented in Figure A.2. All the components and channels are explained in this appendix to clarify the functioning of the model.

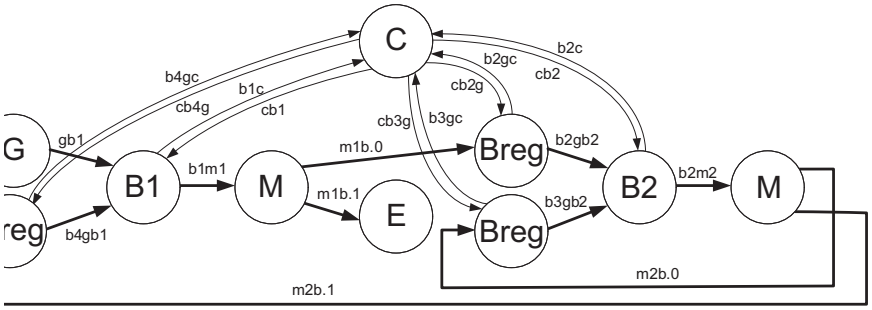


Figure A.2: Discrete event χ-model of the system controlled by buffer regulators.

Channels

The bold arrows in Figure A.2 represent the channels through which jobs are transferred. In this model, each job is identified with its arrival time into the system, i.e. `type job = real`. The thin arrows represent the channels through which the controller communicates with the buffers. The controller can send the initial buffer contents to each buffer and every buffer sends the total number of jobs stored in the buffer to the controller which saves all the data that can be used to analyze the resulting responses of the controlled system.

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

type job = real     // arrival time of a job into the system
```

Generator

The generator will always send generated jobs with rate λ into the first buffer until the simulation time is reached.

```
proc G( gb1: !job ) =
|[ lambda :-> real, wait: real
 | lambda := exponential(1.0)
 ; *[ time <= 150000.0 -> gb1!time
                        ; wait := sample lambda
                        ; delta wait
    ]
]|
```

Controller

For this model, the gated policy is used to control the system instead of the controller. The controller is only aware of the buffer contents of each buffer in the system.
At startup of the model, the controller asks for the initial buffer contents which has to be entered. After that, every initial buffer contents is sent to the corresponding buffer and the production cycle can start. The system contains 3 regulator buffers (b2g, b3g and b4g) — called Breg in Figure A.2 — and two regulated buffers (b1 and b2). Both regulated buffers contain two lists to store the jobs of each step respectively. Every time when a job enters or leaves a buffer, the controller receives the total number of jobs stored in this buffer. All these data are saved in a file until the simulation time is reached. These data can be used to analyze the performance of the system.

```
proc C( b1c,b2c: (?nat)^2, b2gc,b3gc,b4gc: ?nat, cb1,cb2g,cb3g,cb4g: !nat, save: !file ) =
|[ x1A,x2B,x2Bg,x3B,x3Bg,x4A,x4Ag,x1Aold,x4Aold,x2Bold,x3Bold : nat
 | !"buffer contents x1A: " ; ?x1A ; !"\n"
 ; !"buffer contents x4A: " ; ?x4Ag ; !"\n"
 ; !"buffer contents x2B: " ; ?x2Bg ; !"\n"
 ; !"buffer contents x3B: " ; ?x3Bg ; !"\n"
 ; cb1!x1A ; cb4g!x4Ag ; cb2g!x2Bg ; cb3g!x3Bg
 ; x4A:=0; x2B:=0; x3B:=0
 ; < x1Aold, x4Aold, x2Bold, x3Bold > := < x1A, x4A+x4Ag, x2B+x2Bg, x3B+x3Bg >
 ; x1Aold:=x1A; x4Aold:=x4A+x4Ag; x2Bold:=x2B+x2Bg; x3Bold:=x3B+x3Bg
 ;*[ time <= 150000.0 ->
                [ true ; b1c.0?x1A -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b1c.1?x4A -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b4gc?x4Ag -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b2c.0?x2B -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b2c.1?x3B -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b2gc?x2Bg -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                | true ; b3gc?x3Bg -> save! x1Aold, "\t", x4Aold, "\t", x2Bold, "\t", x3Bold, "\t", time, "\n"
                                    ; save! x1A, "\t", x4A+x4Ag, "\t", x2B+x2Bg, "\t", x3B+x3Bg, "\t", time, "\n"
                                    ; x1Aold := x1A ; x4Aold := x4A+x4Ag ; x2Bold := x2B+x2Bg ; x3Bold := x3B+x3Bg
                ]
   ]
]|
```

Gated policy

In this model, the gated policy is used to control the system. This policy contains a switching function and regulator buffers in front of every buffer as depicted in Figure A.2. At startup of the model, the initial buffer contents are stored in the corresponding regulator buffers with arrival time 0.0 time-units. Every regulator buffer can always receive the jobs from a workstation which are stored in a list. These jobs are sent with inter arrival time $1\frac{\text{job}}{\text{time-unit}}$ to the regulated buffer. Every time when a job enters or leaves this list in the regulator buffer,

the controller receives the total number of jobs stored in this list.
The gated policy only switches the system to another mode when a buffer becomes empty. The switching function "next" determines this switching moment and switches the workstation to the next step.

```
proc BufReg( mbg: ?job, cbg: ?nat, bgb: !job, bgc: !nat ) =
|[ x: job, xini: nat, xs: job*, next_job: bool, tnext: real
 | next_job := true
 ; [ true; cbg?xini -> *[ xini > 0 -> xs := xs ++ [ 0.0 ]; xini := xini - 1 ] ]
 ; *[ true; mbg?x   -> xs := xs ++ [x]; bgc!len(xs)
    | len(xs) > 0 and next_job; bgb!hd(xs) -> xs := tl(xs); bgc!len(xs); next_job := false; tnext := time + 1.0
    | not next_job; delta tnext - time     -> next_job := true
    ]
 ]|

 func next( xs: (job*)^2, k: nat ) -> nat =    // k is current step
|[ i,m: nat                                    // m is required step
 | m := k; i := 0                             // i is possible step
 ; *[ i < 2 and len(xs.k) = 0
       -> [ len(xs.i) >  len(xs.m) -> m := i
          | len(xs.i) <= len(xs.m) -> skip
          ]
        ; i := i+1
    ]
 ; ret m
]|
```

Buffers B1 and B2

Both buffers can always receive jobs from the generator or regulator buffers. Each buffer contains two lists (`xs.0 and xs.1`) where jobs from a step are stored respectively. Only the initial amount of jobs for step 1 are stored in the corresponding list with arrival time 0.0 time-units at startup of the model. During the simulation, all the jobs generated by the generator are also stored in this list. All the other lists receive the jobs from their corresponding regulator buffer. The jobs from a list are sent to the workstation until the list is empty. Every time when a job enters or leaves a list, the buffer sends the total number of stored jobs in this list to the controller.

```
proc B1( cb1: ?nat, gb1,b4gb1: ?job, b1m1: (!job)^2, b1c: (!nat)^2 ) =
|[ xs: (job*)^2, x: job, k,xini: nat
 | k := 0; *[ k < 2 -> xs.k := []; k := k+1 ]   // define for each production step a list
 ; k := 0
 ; [ true; cb1?xini -> *[ xini > 0 -> xs.0 := xs.0 ++ [ 0.0 ]; xini := xini - 1 ] ] // initial amount of jobs is stored in the first buffer
 ; *[ true; gb1?x   -> xs.0 := xs.0 ++ [x]; b1c.0!len(xs.0)
    | true; b4gb1?x -> xs.1 := xs.1 ++ [x]; b1c.1!len(xs.1)
    | len(xs.next(xs,k)) > 0; b1m1.next(xs,k)!hd(xs.next(xs,k))
      -> k := next(xs,k); xs.k := tl(xs.k); b1c.k!len(xs.k)
    ]
]|

proc B2( b2gb2,b3gb2: ?job, b2m2: (!job)^2, b2c: (!nat)^2 ) =
|[ xs: (job*)^2, x: job, k: nat
 | k := 0; *[ k < 2 -> xs.k := []; k := k+1 ]   // define for each production step a list
 ; k := 0
 ; *[ true; b2gb2?x -> xs.0 := xs.0 ++ [x]; b2c.0!len(xs.0)
    | true; b3gb2?x -> xs.1 := xs.1 ++ [x]; b2c.1!len(xs.1)
    | len(xs.next(xs,k)) > 0; b2m2.next(xs,k)!hd(xs.next(xs,k))
      -> k := next(xs,k); xs.k := tl(xs.k); b2c.k!len(xs.k)
    ]
]|
```

Workstation M

Every time when a workstation becomes available, it asks the buffer for a job, i.e. (`bm.j?x`). If the job comes from the same list as the previous processed job (i.e. `j = k`), the workstation continues processing the job at maximal rate. If the job does not come from the same list

as the previous processed job (i.e. `not j = k`), the workstation is first setting up to the next step before it can process the job at maximal rate. If a job is processed at maximal rate, it is sent either to the next buffer or to the exit.

```
proc M( bm: (?job)^2, mb: (!job)^2,  t: real^2 ) =
|[ k: nat, x: job
 , setuptime: -> real, setup :real
 , processtime: -> real, process :real
 | k := 0 ; setuptime := exponential(50.0)
 ; *[ j:nat <- 0..2: true; bm.j?x
     -> processtime := exponential(t.j)
      ; [ j = k   // j is step of current job and k is step of previous processed job
          -> process := sample processtime; delta process; mb.k!x
        | not (j = k)
          -> k := j; setup := sample setuptime; delta setup
           ; process := sample processtime; delta process; mb.k!x
        ]
   ]
]|
```

Exit

The exit E can always receive jobs from the workstation. Every time when the exit has received a job, the job leaves the system.

```
proc E( m1b: ?job ) =
|[ x: job
 | *[ true -> m1b?x ]
]|
```

Cluster

The cluster BufferRegulators connects all the components by channels as presented in Figure A.2.

```
clus BufferRegulators() =
|[ gb1, b2gb2, b3gb2, b4gb1: -job
 , b1m1, b2m2, m1b, m2b: (-job)^2
 , b1c, b2c: (-nat)^2
 , cb1, cb2g, cb3g, cb4g, b2gc, b3gc, b4gc: -nat
 | G( gb1 )
|| B1( cb1, gb1, b4gb1, b1m1, b1c )
|| B2( b2gb2, b3gb2, b2m2, b2c )
|| BufReg( m1b.0, cb2g, b2gb2, b2gc )
|| BufReg( m2b.0, cb3g, b3gb2, b3gc )
|| BufReg( m2b.1, cb4g, b4gb1, b4gc )
|| M( b1m1, m1b, <| 0.3, 0.6|> )
|| M( b2m2, m2b, <| 0.6, 0.3|> )
|| E( m1b.1 )
|| C( b1c, b2c, b2gc, b3gc, b4gc, cb1, cb2g, cb3g, cb4g, fileout( "Results_BufferRegulators.txt" ) )
]|

xper = |[ BufferRegulators() ]|
```

Appendix B

Distributed Controller

B.1 Discrete event χ-script

The discrete event model of the specific reentrant manufacturing system controlled by a distributed controller is presented in Figure A.1. For this model, each workstation M only needs local state information. Therefore, the controller is divided into two parts by a dotted line. The controller can decide the next task for a workstation without knowledge of the state of the other workstation. The results of the simulations are presented in Chapter 5. All the components and channels are explained in this appendix to clarify the functioning of the model.

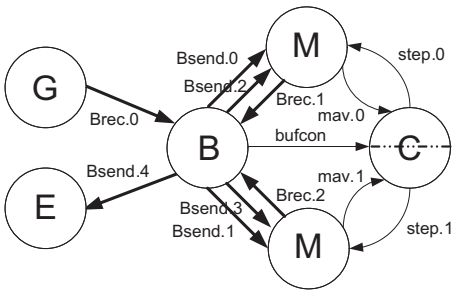


Figure B.1: Discrete event χ-model of the system controlled by a distributed controller.

Channels

The bold arrows in Figure B.1 represent the channels through which jobs are transferred. In this model, each job is identified with its processing step, i.e. `type job = nat`. Step 1 corresponds with job = 0, step 2 corresponds with job = 1, step 3 corresponds with job = 2 and step 4 corresponds with job = 3.
The thin arrows represent the channels through which the controller communicates with the components. If a workstation becomes available, it sends a signal to the controller through channel `mav.0 or mav.1` to ask the controller for their next task. The controller will send the production step as a task through channel `step.0 or step.1` to the workstation. Every time

when a job enters or leaves the buffer, the total number of jobs stored in the buffer is sent to the controller through channel `bufcon`. All these buffer contents are saved with the current time in a file which can be used to analyze the performance of the controlled system.

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

type job = nat
```

Generator

The generator will always send generated jobs with rate λ to the first list in the buffer.

```
proc G( gb : !job ) =
|[ deltagen :-> real, wait: real
 | deltagen := exponential(1.0)
 ; *[ true -> gb!0
            ; wait := sample deltagen
            ; delta wait
    ]
]|
```

Controller

At startup of the model, the initial buffer contents are stored in the list `xs:= xinit` and the production cycle can start. The controller receives a signal from a workstation `(m1av.i?)` when it becomes available. As the controller is always aware of the state of the system, it can decide what the workstation has to do. If the state satisfies the conditions to switch to another step, the controller tells the workstation to which step it must perform the setup, otherwise it tells a workstation which step has to processed at maximal rate until the conditions are satisfied. All these conditions are defined in the function "determinemode" which are equal to the distributed controller description as presented in Chapter 5.

```
proc C( bufcon: ?nat^4, mav: (?void)^2, step: (!nat)^2, xinit: nat^4, save: !file ) =
|[ mode: nat^2, xs,nrprod: nat^4, x2bar: nat, setuptwo: bool
 | xs:= xinit
 ; mode:= <| 1, 2 |>; nrprod:= <|0,0,0,0|>; setuptwo:= false
 ; *[ true -> [ true; bufcon?xs -> skip
              | i:nat<-0..2: true; mav.i?
                -> nrprod.(mode.i-1):= nrprod.(mode.i-1)+1
                 ; [ not setuptwo -> skip
                   | setuptwo and xs.1 <  x2bar -> setuptwo := false
                   | setuptwo and xs.1 >= x2bar -> x2bar := xs.1
                   ]
                 ; < mode, nrprod, setuptwo > := determinemode( mode, nrprod, xs, x2bar )
                 ; [ setuptwo     -> x2bar:= xs.1    // setup to step 2 has been completed and the number of jobs in buffer 2 is defined
                   | not setuptwo -> skip
                   ]
                 ; step.i!mode.i
              ]
            ; save! xs.0, "\t", xs.1, "\t", xs.2, "\t", xs.3, "\t", time, "\n"
    ]
]|

func determinemode( mode: nat^2, nrprod,xs: nat^4,x2bar: nat ) -> nat^2 # nat^4 # bool =
|[ setuptwo: bool
 | setuptwo:= false
 ; [ mode.0=1 and xs.0>0                                              -> skip
   | mode.0=1 and xs.0=0 and nrprod.0< 1000                          -> skip
   | mode.0=1 and xs.0=0 and nrprod.0>=1000                          -> mode.0:= 4; nrprod.3:= 0
   | mode.0=4 and xs.3>0                                             -> skip
   | mode.0=4 and xs.3=0 and nrprod.3< (nrprod.0 div 2) and xs.0< 650 -> skip
   | mode.0=4 and xs.3=0 and nrprod.3< (nrprod.0 div 2) and xs.0>=650 -> mode.0:= 1; nrprod.0:= 0
   | mode.0=4 and xs.3=0 and nrprod.3>=(nrprod.0 div 2)               -> mode.0:= 1; nrprod.0:= 0
  ]
```

```
 ; [ mode.1=2 and nrprod.1< max(1000, x2bar+(nrprod.2 div 2)) -> skip
   | mode.1=2 and nrprod.1>=max(1000, x2bar+(nrprod.2 div 2)) -> mode.1:= 3; nrprod.2:= 0
   | mode.1=3 and xs.2>0                                      -> skip
   | mode.1=3 and xs.2=0                                      -> mode.1:= 2; nrprod.1:= 0; setuptwo:= true
   ]
 ; ret < mode, nrprod, setuptwo >
]|
```

Buffer

The buffer can always receive jobs from the generator or workstation and contains 5 lists. In the first 4 lists, jobs from each step are stored respectively. All jobs for step 1 are stored in the first list, for step 2 in de second list, for step 3 in the third list and for step 4 in the fourth list. All jobs that have been processed for step 4 are stored in the fifth list which are sent to the exit when the exit can receive a job. At startup of the model, the initial buffer contents are stored in the corresponding buffers with arrival time 0.0 time-units. Every time when a job from a step enters or leaves a buffer, the buffer sends the total number of stored jobs from each step to the controller.

```
proc B( Brec: (?job)^3, Bsend: (!job)^5, bufcon: !nat^4, xinit: nat^4 ) =
|[ x: job, xs: nat^5
 | xs := <| xinit.0, xinit.1, xinit.2, xinit.3, 0 |>
 ; *[ true -> [ i:nat<-0..3: true    ; Brec.i?x  -> xs.x:=xs.x+1
              | i:nat<-0..5: xs.i > 0; Bsend.i!i -> xs.i:=xs.i-1
              ]
            ; bufcon!<| xs.0, xs.1, xs.2, xs.3 |>
    ]
]|
```

Workstation

Every time when a workstation becomes available (`mav!`), it asks the controller which step should be processed (`step?mode`). Note that the mode of a workstation represents the processing production step. If the previous processed job is from the same step as the processing job, the workstation continues processing at maximal rate. Otherwise, it is first setting up to the next step before it can process the jobs at maximal rate. If a job is processed at maximal rate, it is sent to the next buffer or to the exit.

```
proc M( bm1,bm2: ?job, mb: !job, mav: !void, step: ?nat ) =
|[ mode,prevmode: nat, x:job
 , setup,proctime13,proctime42:-> real
 , wait,process13,process42: real
 | prevmode := 0
 ; setup := exponential(50.0)
 ; proctime13 := exponential(0.3)
 ; proctime42 := exponential(0.6)
 ; *[ true -> mav!; step?mode
            ; [ mode =prevmode -> skip
              | mode/=prevmode -> wait := sample(setup); delta wait
              ]
            ; prevmode:=mode
            ; [ mode=1 -> bm1?x; process13 := sample(proctime13); delta process13
              | mode=2 -> bm1?x; process42 := sample(proctime42); delta process42
              | mode=3 -> bm2?x; process13 := sample(proctime13); delta process13
              | mode=4 -> bm2?x; process42 := sample(proctime42); delta process42
              ]
            ; mb!x+1
    ]
]|
```

Exit

The exit E can always receive jobs from the buffer. Every time when the exit has received a job, the job leaves the system.

```
proc E( be:?job ) =
|[ x: job
 | *[ true -> be?x ]
]|
```

Cluster

The cluster Distributed connects all the components by channels as presented in Figure B.1. Also, the initial buffer contents of each buffer is given.

```
clus Distributed( save: !file ) =
|[ Brec: (-job)^3, Bsend: (-job)^5, bufcon: -nat^4, mav: (-void)^2, step: (-nat)^2
 | G( Brec.0 )
|| B( Brec, Bsend, bufcon, <|1000,1000,1000,1000|> )
|| M( Bsend.0, Bsend.3, Brec.1, mav.0, step.0 )
|| M( Bsend.1, Bsend.2, Brec.2, mav.1, step.1 )
|| E( Bsend.4 )
|| C( bufcon, mav, step, <|1000,1000,1000,1000|>, save )
]|

xper = |[ Distributed( fileout( "Distributed_Results.txt" ) ) ]|
```