

Modelling and control of discrete event manufacturing flow lines

J.A.W.M. van Eekelen

Modelling and control of discrete event manufacturing flow lines

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 8 januari 2008 om 16.00 uur

door

Jozef Antonius Wilhelmus Maria van Eekelen
geboren te 's-Hertogenbosch

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. J.E. Rooda

en

prof.dr. H. Nijmeijer

Copromotor:

dr.ir. A.A.J. Lefeber

A catalogue record is available from the Eindhoven University of Technology Library

ISBN: 978-90-386-1175-4

Cover design: Jelle Stienstra

Reproduction: Universiteitsdrukkerij Technische Universiteit Eindhoven

© Copyright 2007, J.A.W.M. van Eekelen

Preface

During my masters project in the early 2000s, I became really enthusiastic about scientific research. This has been the reason that I decided to continue the research on modelling and control of manufacturing systems in a Ph.D. project. This dissertation is the fruit of a four years' study on the subject, performed within the Systems Engineering group at the Eindhoven University of Technology. I hope that this thesis will find its way within the Systems Engineering group, as a reference for future research and maybe as course material for graduate students.

The opportunity to perform research in an open academic environment made me a privileged man. Being able to think freely about fundamental issues in manufacturing engineering improved my knowledge on several subjects and made my mind much riper. On the other hand, such a project is so personal, that the research matters often kept haunting me in the evenings, nights, weekends, ... Nevertheless, I am proud to have passed the scientific and mental challenge of this Ph.D. project. Apart from several meetings in the Benelux, I was given the opportunity to present my work at big conferences in San Francisco, Seville, Minneapolis, San Diego and New York, for which I am very grateful to my supervisors.

In 2003, I succeeded Joost Vervoort in this project. This research project had been initiated together with Ph.D. student Bas Roset, under the supervision of Koos Rooda, Henk Nijmeijer and my coach Erjen Lefeber. Thank you all for the discussions, inspiration and contributions. Many thanks go to Koos Rooda and Mieke Lousberg for facilitating my research project in such a way that I only had to care about the scientific part, without the administrative and financial paperwork. I also would like to thank my colleagues of the Systems Engineering group for their expertise, the stimulating working environment and many social moments. My office mates Roel van den Berg, Bas Roset and Mihály Petreczky are thanked for the lively discussions on all kinds of subjects, good and bad coffee, sick jokes and other distracting activities.

The research presented in this thesis has been performed in great cooperation with my coach Erjen Lefeber. We worked together for about seven years, first in my masters project and then in this Ph.D. project, sharing our thoughts on both scientific and non-scientific matters. Erjen, I really appreciate your listening, opinion, support and understanding.

Several bachelor and master students participated in this research. Thank you all, Stefan Forschelen, Willem de Koning, Jasper van de Loo, Toby Luiten, Sven Platschorre, Roy Smolders, Sander Verhoeven, Dennis Wetjens and Dirk van Zwieten for your contributions.

I would like to thank the external members of the Doctorate Committee (professor Dieter Armbruster, professor Bart De Schutter, professor Paolo Valigi and doctor Ivo Adan) for carefully reading this dissertation and giving some very useful comments and suggestions.

Finally, I would like to thank my family and friends for their support, love and many enjoyable moments. I really regret that I cannot present this dissertation to my father anymore, but I sincerely hope that, looking from above, he is as proud of me as I am of him. Thank you mama and Bob for always being there for me, giving good advises and supporting me unconditionally. A special word of thanks goes to Cecile Utens for her love and support during my Ph.D. project and the beautiful moments we shared together. Good luck with your own research career!

Joost van Eekelen
November 2007

Summary

Over the last decades complexity of products and production processes has increased tremendously, moving towards high-tech production systems: manufacturing expensive products with even more expensive resources. Failures or mistakes have therefore become expensive too and need to be avoided to fulfill the manufacturers' targets: generating products while maximizing the profit. In general, this is to be achieved by keeping vast control over the manufacturing processes, resources, stocks and labour. Easier said than done, since controlling all phenomena that occur in a manufacturing system is very expensive, if possible at all. Therefore, specific parts of a manufacturing facility are modelled to reduce its complexity. With the models, predictions of future behavior of the system can be made. Moreover, different control strategies can be tested offline at low risks, before implementing them on the real production system.

In this dissertation, manufacturing flow lines are modelled using several modelling paradigms. These are divided into three groups: discrete event models, continuous models and hybrid models. The presented modelling methods are used throughout the remainder of the thesis and therefore the survey is by no means an attempt to give a complete overview of the whole area of modelling manufacturing systems.

A state space representation of a manufacturing workstation is introduced, which is finite dimensional, can be measured instantaneously and does not contain any information about production or control policy. This state space representation is used in the coupling of different model paradigms, facilitating the use of analysis techniques in both time domain and event domain. In addition, the introduced state space representation is used in the development of a continuous time receding horizon state feedback controller. For a flow line of multiple workstations, each with its own buffer capacity and process time, and for an event based control horizon, an optimal production schedule controller is developed. This state feedback controller provides optimal schedules, even when unexpected disturbances occur.

Switching servers are found in a wide variety, like in manufacturing industries, traffic networks and call-centers. Switching servers process multiple types of jobs, with a switchover time involved. A hybrid fluid model is used to describe the dynamics of switching servers. Continuous

dynamics is used to describe the evolution of buffer levels. The discrete event part of the model describes the switches between the product types. For a workstation processing two product types that arrive at constant rates, optimal switching policies with respect to minimal time averaged weighted work in process levels are defined for situations with and without maximum buffer level capacities. An important insight is the possible appearance of a slow-mode in optimal process cycles. During a slow-mode, a buffer is empty and products are served at their arrival rate, instead of switching to the other product type. The slow-mode represents a trade-off between losing capacity due to serving products at a lower rate than the maximum rate and losing capacity due to relatively often switching in time. Conditions on the appearance of a slow-mode in optimal process cycles are derived explicitly. In a manufacturing network, arrivals of products at a workstation are in general not at a constant rate. For a switching server processing two product types and with piecewise constant periodic arrival rates (on/off), optimal system behavior with respect to work in process levels is defined. The total optimization problem then splits into several subproblems, which need to be solved separately.

The optimal mean work in process level for a single switching server is also a lower bound on the mean work in process level that can be realized for a switching server flow line in which that server resides. For a flow line consisting of two switching servers, each processing two product types, it is investigated under which conditions this lower bound can actually be achieved. An important conclusion is that, in certain cases, workstations in a flow line need a synchronization mechanism to get to the desired process cycles.

For the single switching server and the switching server flow lines with constant arrival rates of products, state feedback controllers are proposed to steer the trajectory of the switching server (flow line) to the determined optimal process cycle from arbitrary feasible starting point. Contrary to many methods proposed in literature, in this research first the desired (optimal) system behavior is defined, regardless of any control policy. Then a control policy is formulated to achieve this desired behavior.

It is questionable whether optimal system behavior should be looked for. The studies in this thesis show that optimal process cycles can be determined for a rather small class of workstations. Taking more than two product types into account or more than two workstations leads to very complex optimization problems. Apart from the complexity, it is not even known whether optimal cyclic behavior for these larger systems exists. Manufacturers might not be interested in the theoretically optimal solution. Often, a better solution than the current solution will do. Moreover, a manufacturer might prefer suboptimal solutions over optimal solutions when the suboptimal solution handles disturbances or uncertainties better.

This dissertation can serve as a starting point for further research on modelling and control of manufacturing networks. The introduced state space representation topic can be extended for larger networks and other manufacturing resources. In addition, other research areas can be linked to this research, including stochastic behavior and effective process times.

Contents

Preface	v
Summary	vii
1 Introduction	1
1.1 Manufacturing systems	1
1.2 Model and control framework	5
1.3 Objective and contributions	7
1.4 Outline of this thesis	9
2 Modelling manufacturing systems	11
2.1 Discrete event models	13
2.2 Continuous models	23
2.3 Hybrid models	30
2.4 Case study: modelling a flow line	40
2.5 Summary	47
3 Coupling event domain and time domain models of manufacturing systems	49
3.1 (State space) Dynamical systems	50
3.2 Modelling a workstation	53
3.3 Maps between states and signals	59
3.4 Summary	64

4	Receding horizon control using (multi-parametric) linear programming	65
4.1	Model predictive control and (multi-parametric) linear programming	66
4.2	Deriving a controller for a single workstation	71
4.3	Flow line: interconnection and case study	84
4.4	Summary	90
5	A switching server	93
5.1	Characteristics and dynamics of two queues switching server	95
5.2	Optimal process cycle of two queues switching server	98
5.3	State feedback control of switching server	105
5.4	Finite buffer capacities: effects on optimal process cycle and controller	106
5.5	Case study: controller implementation	109
5.6	Discrete event analysis	117
5.7	On multiple product types and optimal process cycles	119
5.8	Piecewise constant arrival rates	121
5.9	Summary	136
6	Flow lines of switching servers	139
6.1	Characteristics and dynamics of flow lines of switching servers	140
6.2	Process cycles and controllers for switching server flow line	143
6.3	General problem of two workstations flow line	161
6.4	Larger flow lines	164
6.5	Summary	169
7	Conclusions and recommendations	173
7.1	Conclusions	173
7.2	Recommendations for further research	177
	References	179
	A Proofs	187
	B Matlab and χ models	235
	Nederlandse samenvatting	251
	Curriculum vitae	255

Introduction

1.1 Manufacturing systems

Manufacturing systems exist in a wide variety. But what exactly is a manufacturing system? Hopp and Spearman [61] use the following definition:

“A manufacturing system is an objective-oriented network of processes through which entities flow”.

The *objective* of the manufacturing system usually is twofold: generating products and making money. However, in academic studies often different objectives are chosen, such as minimal inventory, high customer satisfaction or smooth production. Eventually, all academic objectives can be translated into some money related equivalent. The *network of processes* refers to the lay-out of a factory and the physical processes that take place, like drilling, welding, lithography, etc. Common network topologies are presented in Section 1.1.2. The term *entities* in the definition refers to the kind of goods that are produced. Is it the product of mass-fabrication that is hard to distinguish as an individual item? Is it a batch of breads in a bakery? Or is it a customized car from a car manufacturer? In each case, the *entity* is different. More on manufacturing entities is presented in Section 1.1.1. Finally, the verb *flow* in the definition describes how materials and information are processed. In modelling manufacturing systems, this flow characterization has a great influence on the model type that is suitable for the specific factory.

A common factor of all manufacturing systems is that they convert material into products. This conversion process in general consists of a multitude of process/production steps. At the start

of the manufacturing process, raw material is used (e.g. steel, water, oil) or the product from another manufacturing process (e.g. at an assembly station). At the end of the manufacturing process, products leave the system and are either sold or used as starting point for another manufacturing process. During all manufacturing processes, value is added to the products. This relates to the general objective of making money, as mentioned before.

1.1.1 Lots and products

The items that are manufactured are called *lots*. Often a *lot* is the same as a *product*, for example in car manufacturing industry, where cars are produced and transported in the facility on a one-by-one basis. (With the current mass-customization of cars, it often occurs that a car manufacturer does not produce two cars that are exactly the same in one day.)

It is also possible that a lot consists of multiple products. An example of this is the semiconductor industry, in which a pod with wafers travels through the plant. The pod represents the lot, while the wafers are the actual products. An even more complicating aspect is that not all pods contain the same number of wafers during production.

Another example is a bakery, in which individual breads are the lots for a cutter, but a batch of breads represents a lot for the oven. This example shows that even at different production steps within a manufacturing system the term *lot* may refer to different amounts of products.

In this thesis, the term *lot* is used as the manufacturing entity, although in most cases it can be interchanged with *product* freely.

1.1.2 Manufacturing system topologies

As stated in the definition of a manufacturing system, lots flow through a *network of processes*. The processes are physically arranged in a certain manner on the factory floor, the so-called *lay-out* of the factory. Many different lay-out structures exist. The most basic concepts are called *job-shop*, *flow line* or *continuous flow processes*, cf. [26].

Job-shops are resource oriented: lots have a non-fixed route through the facility. Job-shops are often used for low-volume, highly customized products, in which each product has its own recipe and requirements. Examples of job-shops are custom furniture shops, commercial print services, ship yards, etc.

Flow lines are product oriented manufacturing systems in which products have a fixed recipe and route. Especially for high-volume, highly standardized products a flow line is often used. Examples of flow lines can be found in (parts of the) car industry, food processing facilities, mass textile fabrication, etc. A special form of flow lines emerges when products have to travel a certain part of the flow line more than once. In that case a *reentrant flow line* is obtained. In this thesis, the focus is on flow lines without reentrant behavior.

Continuous flow processes are facilities in which material literally flows through the factory.

The route of products is fixed and in general cannot be changed very easily. Examples are oil refineries, beer breweries, gas installations, steel producers, etc.

Zooming in on the physical lay-out of a manufacturing system, one encounters *workstations*. A workstation is a manufacturing resource consisting of a buffer and a number of parallel competing machines. The buffer has either infinite or finite storage capacity and is used to store incoming lots from other workstations. The machines are fed with products from the buffer and send them to the next workstation after completion of the production step. In this thesis it is assumed that a workstation contains only one machine, unless indicated otherwise. In addition, the buffer is assumed to work on a FIFO basis (first-in-first-out). This means that the order in which lots are taken from the buffer is the same as the order in which the lots arrived at the buffer. A schematic picture of a flow line consisting of four workstations is presented in Figure 1.1. The buffers are denoted by B and the machines by M . These capitals are used throughout the remainder of this thesis. Note that the term *downstream* indicates the direction in which the lots flow and *upstream* denotes the opposite direction.

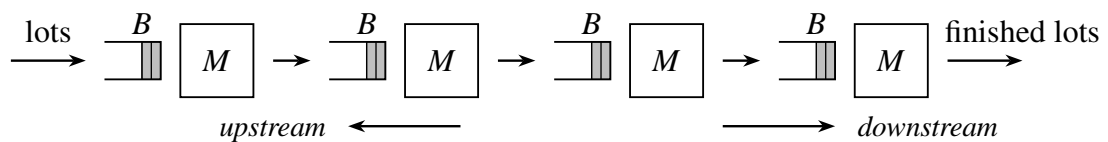


Figure 1.1: Example of a flow line.

Taking a closer look at the machine itself in a workstation, different types of machines exist. The most commonly used machines are *single-lot machines*, *batch machines* or *conveyors*. Single-lot machines are machines that process only one lot at a time. After completion of the production step, the lot is sent away and only then the next lot can be taken from the buffer. Single-lot machines are mostly considered in this research. Batch machines are machines that process a number of lots in parallel. This number of lots can be fixed or variable. An example of a batch machine is the aforementioned oven in a bakery. A number of breads is taken from the preceding buffer and baked simultaneously. After baking, the breads leave the oven at the same time. A conveyor is a machine that is able to process or transport multiple lots simultaneously. Lots do not have to start on a conveyor at the same time instant. Examples of conveyors are transport belts for suitcases on airports or pasteurizers in a beer brewery, where filled bottles move slowly through a heated area.

1.1.3 Common performance measures

Manufacturers may have different objectives for their production facility. Sometimes, a manufacturer wants to produce as many products as possible. Others just want to meet a certain customer demand with the lowest possible amount of lots in the system, while again others just want to keep the response time to their market low. Some basic quantities are defined here (taken from [93]) that characterize in some sense the performance of a manufacturing system.

Throughput δ of a manufacturing system denotes the number of lots per time unit that leave the system. In a steady-state situation, the mean throughput equals the mean number of jobs that enter the system.

Flow time ϕ of a lot denotes the time a lot is in the manufacturing system. It includes all process times, waiting times, transportation times, etc.

Work in process (*wip*) level w denotes the total number of lots in the manufacturing system. In addition to momentaneous wip levels, one is often interested in *mean* wip levels. The mean wip level is the time averaged value of momentaneous wip levels. Especially in manufacturing systems where variability occurs (in practice all manufacturing systems), the mean wip level is often more important than the momentaneous wip level.

A relationship exists between throughput δ , flow time ϕ and wip level w for a steady-state situation. This relationship is called Little's law, named after the man who proved the law mathematically [75]. The bars $\bar{}$ indicate mean values. Little's law is given by

$$\bar{w} = \bar{\delta} \cdot \bar{\phi}$$

and although not quite right, the bars are often omitted. Little's law appears often in modelling and analysis of manufacturing systems, as becomes clear in the remaining of this thesis. It can for example be used to compute one of the three quantities if only two of them are measurable. A disguised variant of Little's law appears in models of manufacturing systems with partial differential equations (Section 2.2.2). Little's law is also used for queue length calculations in a variety of applications.

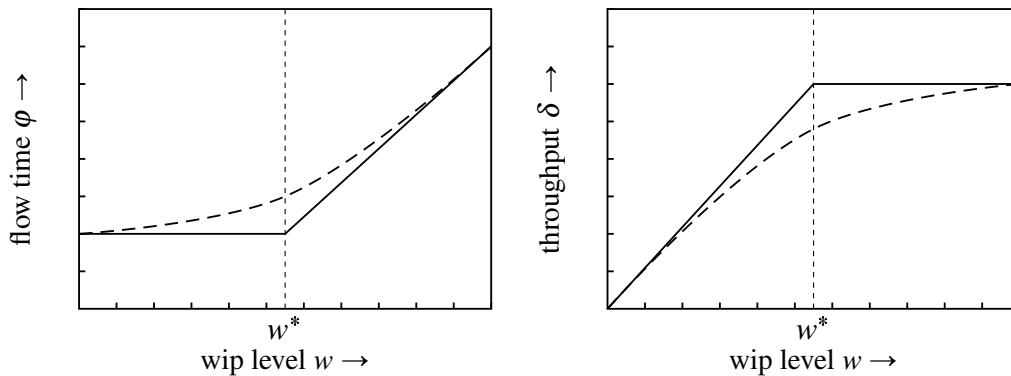


Figure 1.2: Relations between wip, throughput and flow time.

Quantities like throughput and flow time are bounded quantities, e.g. the flow time of lots in a manufacturing system is at least the sum of all process times of production steps and the sum of all transportation times. Even so, the throughput is bounded by the production capacity (in lots per time unit) of the slowest machine. These observations result in Figure 1.2. The figure shows that in a deterministic situation, a critical wip level w^* exists that separates the curves into two parts. Below this critical wip level, the throughput increases as the wip level increases, keeping the flow time constant. Above the critical wip level, the throughput remains constant, whereas the flow time increases. Practical manufacturing systems are never completely deterministic,

resulting in the smoother dashed lines in the figure. Little's law holds for both the deterministic situation and the situation in which variability occurs.

Note that the performance measures throughput, flow time and mean wip all relate to time. Either it is a time averaged quantity or the quantity itself represents a time duration. In the next chapters it is shown that some elegant model types of manufacturing systems are not specified in time, which makes computation and interpretation of these time related performance measures difficult. In Chapter 3 a framework is presented that couples model types that are specified in time and model types that are not specified in time.

Another important notion for manufacturing systems is *stability*. A manufacturing system is called *stable* if all buffer levels in the system remain bounded. This definition is commonly used for manufacturing systems, cf. [25, 98]. Two ingredients are necessary to achieve a stable manufacturing system:

- The system characteristics should meet the product inflow (i.e. is it physically possible to process all incoming products?).
- The production policy must stabilize the system.

For example, a manufacturing system that has enough capacity to process all incoming products, but has a production policy that says to process only all odd numbered products is not stable. On the other hand, a manufacturing system with a production policy that says that all incoming products are to be processed as fast as possible, but with an arrival rate of products that outreaches the production capacity is also unstable. Perkins and Kumar [87] show that having enough production capacity is a necessary and sufficient condition for the existence of a scheduling policy which stabilizes the manufacturing system for all possible initial system conditions.

1.2 Model and control framework

The industrial need for advanced control methods for manufacturing systems becomes stronger, in order to make a manufacturing system behave in a pre-determined desired manner. For example, based on the customer demand (due dates of finished products) the production steps in the plant need to be scheduled. A schematic overview of this research objective is shown in Figure 1.3. A large gap exists between on the one hand real industrial manufacturing plants and on the other hand scientific systems and control theory. In order to bridge the gap, models are made of industrial facilities. Models are abstractions of the real system, in which only the important phenomena are incorporated (a more detailed overview of models and modelling techniques is given in Chapter 2). Since models are mathematical representations of the industrial reality, they can be used for analysis, predictions, simulations, testing and controller synthesis. Basically, the model and control framework looks as presented in Figure 1.4. This framework has been adapted from [72, 105, 106].

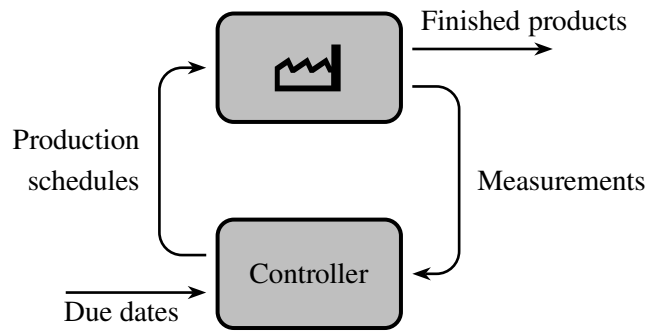


Figure 1.3: Objective of the research: production control of manufacturing systems.

The first step in the framework, arrow ① in Figure 1.4, is to obtain a (mathematical) model of the physical system . Different modelling techniques are possible. Chapter 2 gives an overview of commonly used modelling techniques for manufacturing systems. Inspiration from other application fields can be obtained, for example by comparing a manufacturing system to traffic flow, in which the roads and intersections represent transport and process steps, while cars represent lots. An extensive overview of traffic modelling techniques is presented on Trafficforum [54]. Other application fields that relate to manufacturing systems are railroad networks with trains (lots) and stations (machines) or blood cells (lots) flowing through vessels with junctions (machines).

Once a model of the physical system has been obtained, it can be used for analysis. The model can be validated by exposing it to simulated normal factory circumstances and compare the results to real system behavior. In addition one might derive properties of the model, such as maximum throughput, minimal flow time, etc. One should keep in mind that the properties belong to the *model* rather than the real physical manufacturing system. When working with models for a long time without a view of the physical system, it is tempting to regard the models as the real world. In this thesis, one should keep in mind that all models are abstractions of possible real manufacturing systems. Since the models presented in this thesis have been constructed artificially without linkage to a real physical system, one should constantly be aware that real system behavior differs from the modelled behavior.

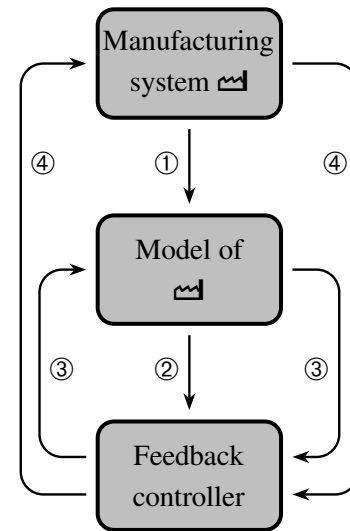



Figure 1.4: Steps in the model and control framework.

The second step in the model and control framework (arrow ② in Figure 1.4) is controller development. Given a desired closed-loop behavior of the manufacturing system, one looks for controllers that achieve this desired behavior. Either a controller is derived from the model, or a controller is proposed based on the insights obtained with the model. If possible, the controller is to be verified mathematically, e.g. a convergence proof to a steady state or limit cycle.

The third step in the model and control framework (loop ③ in Figure 1.4) is to implement and validate the newly obtained controller on the model of the manufacturing system. Validation on the model allows for inspecting the closed-loop behavior without damaging the real manufacturing system. Also different scenarios can be tested that cannot be tested in the factory itself. In this step it is also possible to inspect the closed-loop behavior when disturbances occur in the model, e.g. machine breakdowns, operator unavailability, process time elongations or power cut-offs. The controller should be able to deal with all situations in an appropriate way before one can implement the controller on the real factory, which is step ④ in the framework of Figure 1.4. With this final implementation on the real industrial manufacturing system , Figure 1.3 is obtained, the ultimate objective of the research. Step ④ is not treated in this thesis.

In Figure 1.3, the controller receives measurements from the manufacturing system, which are used to determine new control actions, in this case production schedules. In this thesis, the controller receives *state* measurements. A state space representation for manufacturing systems is introduced, which can be measured instantaneously (as if taking a snapshot of the system) and is finite dimensional. This state space representation is used in Chapters 2–4 for modelling and control of manufacturing systems. For situations where it is not possible to measure the full state, observers might be used to reconstruct the state. Recent developments in observers for manufacturing systems are presented in [94] by Roset, which was partly inspired by Hardouin [51].

1.3 Objective and contributions

The objective of this Ph.D. research is twofold:

1. Model discrete event manufacturing flow lines in such a way that control techniques can be applied.
2. Based on the obtained models of manufacturing flow lines, develop controllers that make the flow line behave in a pre-determined desired (possibly optimal) manner.

Although the manufacturing systems that are to be controlled have a discrete event nature (which is explained in Chapter 2), the model types that are examined in this thesis are mostly based on ordinary differential equations and hybrid models. The resulting controllers can be validated on discrete event models, because the discrete event character can be regarded as a kind of disturbance with respect to the models. How do the controllers deal with these disturbances? It is expected that disturbances are handled well (without specifying ‘well’ here), since all controllers that are developed in this thesis are feedback controllers, that determine control actions based on measurements of the state of the manufacturing system or model under control.

It should be noted that although the focus in this thesis is on manufacturing *flow lines*, many concepts and insights can be applied to other manufacturing topologies as well.

1.3.1 Contributions of this research

The main contributions of the research presented in this thesis are:

- Introduction of a state space representation for manufacturing systems, which is finite dimensional, can be measured instantaneously and does not contain any information about the production and control policy.
- A coupling between different model types of manufacturing systems by means of maps between different state representations and signals, including the newly developed state space representation.
- Continuous time receding horizon control of manufacturing flow lines. Optimal production schedules are determined based on state information, with the feedback law computed off-line.
- Analysis and state feedback control of switching servers (workstations that process multiple lot types with switchover times). Results are obtained for single switching servers and classes of switching server flow lines. Optimal system behavior is defined with respect to mean wip levels and controllers are proposed that steer a switching server (flow line) to this optimal behavior.

1.3.2 Systems science and engineering

Systems science and *systems engineering* are complementary: the one does not exist without the other. It is science that discovers new theoretical methods and techniques, while engineering tries to make the science applicable for industrial purposes. On the other hand, problems or desires that are encountered in the industrial practice may inspire science to develop theory that is able to solve industrial problems or meets these desires.

In addition to stimulation of science and engineering by each other, also fields of tension exist. In science, one often tries to achieve *optimal* solutions to problems, while manufacturers generally want a ‘good enough’ or ‘better than the current’ solution. In Chapters 5 and 6 this field of tension is also encountered. An optimal solution to a problem might not be the solution of best interest. Consider for example a machine which has to be fine-tuned manually. Based on this parameter setting, a net profit is obtained. However, due to market fluctuations, the optimal parameter setting may not be known and needs to be estimated by the manufacturer. Suppose that a choice exists between two machines, each with its own net profit curve, as indicated in Figure 1.5. The optimal choice would be the narrow (left hand side) curve: it yields the largest net profit. However, the best parameter setting is unknown and a little deviation from the optimal setting results in a dramatic decrease of the profit. The other machine however yields less profit, but small deviations from the optimal setting result in a far less decrease of the profit. This example shows that although *systems science* may look for theoretically superior solutions, one should always keep the practical implications in mind, which is more the *engineering* approach.

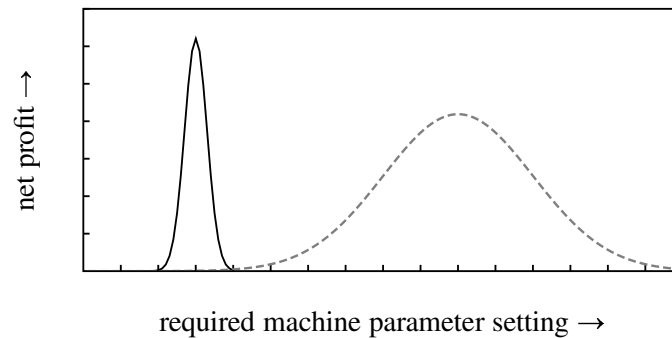


Figure 1.5: Which machine do manufacturers prefer?

1.4 Outline of this thesis

This thesis is organized as described below. After an introduction into modelling of manufacturing systems (Chapter 2), Chapters 3 and 4 form a part concerning modelling and control of manufacturing flow lines that process only one lot type. As mentioned, a state space representation is introduced, which is used in control of discrete event manufacturing flow lines. The models that are used in this part of the thesis allow for distinction of individual lots in the manufacturing system.

Chapters 5 and 6 form a part concerning modelling and control of switching servers, i.e. workstations that process multiple lot types with switchover times. The models that are used in this second part of the thesis are based on fluid approximations, meaning that individual lots cannot be distinguished anymore.

A quick overview of the chapters and their coherence is given below.

Chapter 2 gives a non-exhaustive overview of modelling techniques for manufacturing systems. Three classes of models are distinguished: discrete event models, continuous models and hybrid models. Model types in this chapter are used throughout the remainder of the thesis, or closely related to other models to indicate differences or similarities. A state space representation for a workstation is introduced, which involves both discrete and continuous variables.

In **Chapter 3** a coupling between different model types is established by means of mappings of the state. This facilitates the ‘use best of both worlds’ principle, e.g. measuring the state of one model type and performing analysis in an other model type. A coupling between the event domain and time domain is presented, and between two time domain model types. One of the models in this chapter uses the introduced state space representation of Chapter 2.

In **Chapter 4** a production schedule control method is developed. A feedback controller yields optimal production schedules of jobs in a flow line, based on measurements of the introduced state of the flow line and the due dates of lots. The underlying technique is multi-parametric linear programming. The developed control method is a continuous time receding horizon controller, in which the horizon is event based, i.e. a number of lots that has to be scheduled.

Chapter 5 treats switching servers, which are workstations that process multiple lot types. Switching from one lot type to the other takes time. For a switching server with two lot types and a constant arrival rate of new lots, an optimal process cycle is derived with respect to mean work in process levels. An important obtained insight is that in order to achieve optimal behavior, one should not always work as fast as possible. In addition to the derivation of an optimal process cycle, feedback controllers are proposed that steer the trajectory of a switching server to the desired (optimal) trajectory.

Chapter 6 elaborates on the results of its preceding chapter. Flow lines of switching servers are analyzed and for certain classes of flow lines, optimal process cycles are derived, with feedback controllers that settle down a trajectory of the system to the desired trajectory. The analyses are mostly based on the results of Chapter 5.

Finally, **Chapter 7** concludes this thesis with the general conclusions from all chapters, a summary of the main contributions of this thesis and suggestions for further research on the subjects.

In each chapter, examples are added to illustrate the basic concepts that are treated. In addition, most chapters end with an extensive case study about how the theory from that specific chapter can be applied to flow lines of manufacturing workstations. Although restricted here to flow lines, most ideas and concepts in this thesis can be applied to other manufacturing system topologies as well.

Some useful or insightful models and sources of examples or case studies are included in the appendices. These models and sources are intended to facilitate further research on the subjects.

Modelling manufacturing systems

In manufacturing systems a lot of phenomena occur. Incorporating all phenomena in analyzing manufacturing systems may on the one hand lead to too detailed analyses (which are time consuming and expensive), and on the other hand it can simply be too difficult a job to perform. The phenomena that occur in manufacturing systems can roughly be divided into these groups:

- The physical manufacturing process itself and the set of processes: product recipes.
- Physical capacity constraints: both process capacity (in products per time unit) and storage capacity (in products per storage location).
- The physical layout of the system and transportation between processes. In addition: capacity of transportation systems.
- Temporal influences: change of product mix, modifications of equipment, seasonal influences.
- Human aspects: operator availability (working hours, breaks, meetings), influence of working in shifts, operator dependent product and process quality.
- Scheduled and unscheduled maintenance and all other failure behavior.

To analyze a manufacturing system, abstractions of the real system are made, in which only a selection of the aforementioned groups of phenomena is taken into account. The obtained abstractions are referred to as *models*. Modelling is the first step in the framework that had been introduced in Chapter 1 and is schematically shown in Figure 2.1, where the thick arrow represents the development of models of physical manufacturing systems. Models can be used for analysis, optimization and control of the physical system. Because of the fact that only a limited number of phenomena is considered, one gets a better understanding of the basic dynamics of a physical system, without being distracted by noisy details. However, one should

always keep in mind that models remain abstractions of a real physical manufacturing system, although it is tempting to consider the models as the real world.

Different abstraction techniques exist for modelling manufacturing systems. These abstraction techniques, also called *modelling paradigms*, can be divided into several classes. In this thesis, the division is based on the classification of the signals and domains:

- In *time driven* systems, the state and signals change as time progresses. Time can be progressing continuously or at certain ticks of the clock. The former is called a *continuous time* system, the latter a *discrete time* system.
- In *event driven* systems, the state and signals change due to the occurrence of *events*. Events typically do not take time. These systems are called *discrete event* systems.

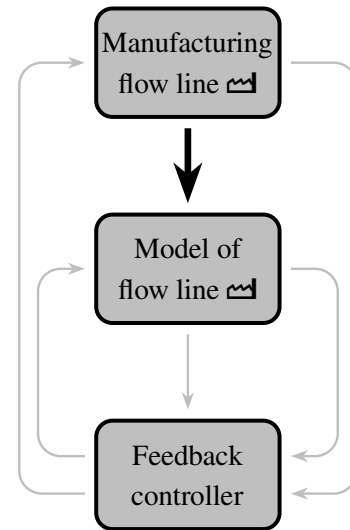


Figure 2.1: First step in the framework: modelling.

Time driven systems are also called systems in the *time domain*, whereas event driven systems are also referred to as systems in the *event domain*.

In addition to the distinction between continuous time, discrete time and discrete event systems, another distinction exists between *continuous variables* and *discrete variables*:

- Continuous variables are variables which can take on any value within an uncountably large allowed set. Typical examples of continuous variables are time, magnitudes of fluids, positions or lengths of objects, velocities and temperatures.
- Discrete variables are variables that can only take on a countable (possibly infinite) number of values (within an allowed set). Typical examples of discrete variables are: a binary signal (on/off), a number of products, an amount of money, a number of people or the primary colors (red, green, blue).

The descriptions of continuous and discrete *variables* also apply to continuous and discrete *states*.

This chapter gives an overview of modelling techniques for manufacturing systems that are used in the remainder of this thesis and is not an attempt to give a full overview of all modelling techniques. Section 2.1 starts with *discrete event models*. Modelling techniques are introduced which are event driven or in which all variables in the model are discrete. Next, Section 2.2 deals with *continuous models*, in which all variables in the models are continuous variables. Note that the domain of these models can be the *discrete time* domain. Finally, Section 2.3 treats hybrid modelling paradigms. In these *hybrid models*, a mix of continuous variables and discrete variables is used and the models can be either time driven or event driven. In all modelling

paradigms, some basic manufacturing entities, e.g. buffers and machines, are modelled. At the end of the chapter, all models are evaluated and compared for a manufacturing flow line example.

2.1 Discrete event models

A certain class of manufacturing systems can be regarded as *discrete event systems*. Loosely speaking, these are systems where individually distinguishable products, which are countable, are led through the facility, like (semiconductor) wafer industry or automotive industry. An example of industry where products are not individually distinguishable is an oil refinery: apart from the molecular level, oil is not countable. However, this type of industry often contains a discrete event part, for example after the oil has been put into barrels. If only the barrel-part is taken into account, the system can be modelled with a discrete event model. If only the oil fluid is taken into account, the system can be modelled with a continuous model, cf. Section 2.2. When both the continuous flow of oil and the barrelling are taken into account, a hybrid modelling paradigm might be considered for modelling the system, cf. Section 2.3.

A property of discrete event systems is that *events occur instantaneously*, so events do not take time. Examples of events are: arrival of a lot, start of processing a lot, finishing a lot, or machine breakdown. However, processes between events can take time, like waiting, processing or transportation.

This section elaborates on modelling pure events. The modelling techniques treated are *max-plus* models, *min-plus* models and *process algebra* models, using formalism χ .

2.1.1 Max-plus models

A valid question in manufacturing systems modelling that may arise is:

“When can a machine start processing a lot?”

Suppose now that the answer is:

“As soon as a lot is available and the previous lot has been finished on the machine.”

This informal description of the machine’s behavior can be formalized. In order to do so, first define $x(k)$ as the time instant the machine starts processing a lot for the k -th time ($k \in \mathbb{N}$) and $u(k)$ as the time instant a lot arrives for the k -th time. The *as soon as* part of the answer can be formalized with a maximum operator. The answer to the same question can now be modelled as:

$$x(k) = \max(u(k), x(k-1) + d) \quad (2.1)$$

where $d \in \mathbb{R}_+ \equiv [0, \infty)$ represents the process time of the machine. In words, (2.1) reads: The k -th time the machine starts processing a lot is the latest of two time instants: the arrival time of that particular lot and the start time of the previous ($k - 1$ -th) lot plus its process time d . The two operators in (2.1) are maximization and addition, and form the operators of the max-plus algebra.

Max-plus algebra

The max-plus algebra consists of the structure $\mathbb{R}_{\max} = (\mathbb{R}_\varepsilon, \oplus, \otimes)$ with $\mathbb{R}_\varepsilon \equiv \mathbb{R} \cup \{\varepsilon\}$, where $\varepsilon = -\infty$. Operations \oplus and \otimes are called max-plus-algebraic addition and max-plus-algebraic multiplication, defined as follows:

$$a \oplus b = \max(a, b) \quad (2.2a)$$

$$a \otimes b = a + b \quad (2.2b)$$

with $a, b \in \mathbb{R}_\varepsilon$. The ‘0’ element in conventional algebra can be compared with ε in max-plus algebra. In conventional algebra, the following property exists:

$$a + 0 = 0 + a = a \quad (2.3a)$$

whereas in max-plus algebra this property exists:

$$a \oplus \varepsilon = \varepsilon \oplus a = a. \quad (2.3b)$$

The ‘1’ in conventional algebra is similar to ‘0’ in max-plus algebra:

$$a \times 1 = 1 \times a = a \quad (2.4a)$$

$$a \otimes 0 = 0 \otimes a = a. \quad (2.4b)$$

Max-plus algebraic matrix additions and multiplications can be defined in a similar way as in conventional linear algebra. If $\mathbf{A}, \mathbf{B} \in \mathbb{R}_\varepsilon^{m \times n}$ and $\mathbf{C} \in \mathbb{R}_\varepsilon^{n \times p}$:

$$(\mathbf{A} \oplus \mathbf{B})_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}) \quad (2.5a)$$

$$(\mathbf{A} \otimes \mathbf{C})_{ij} = \bigoplus_{k=1}^n a_{ik} \otimes c_{kj} = \max_{k=1 \dots n} a_{ik} + c_{kj} \quad (2.5b)$$

for all i, j . For a detailed introduction into max-plus algebra, the reader is referred to [6, 53].

Modelling a workstation in max-plus algebra

In the previous section the basics of max-plus algebra have been explained. Manufacturing entities (building blocks) can be modelled using this algebra. Consider the workstation as shown in Figure 2.2. The workstation consists of buffer B and machine M with constant process time d . A number of ways of modelling the workstation in max-plus algebra exist, depending on the entities one wants to model and the characteristics of the buffer (e.g. finite or infinite storage capacity) and the machine (e.g. single-lot machine or batch machine).

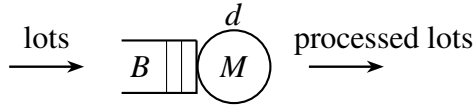


Figure 2.2: Workstation consisting of buffer B and machine M .

Suppose that the items of interest are the time instants that lots enter the buffer, start on the machine and leave the workstation. Define $u(k)$, $x_1(k)$, $x_2(k)$ and $y(k) \in \mathbb{R}_\varepsilon$ as the time instants the k -th lot arrives at the workstation, enters the buffer, is started on the machine and leaves the workstation respectively. Index $k \in \mathbb{N}$ is called the *event counter* of the system. A max-plus algebraic description of this workstation expressed in $u(k)$, $x_1(k)$, $x_2(k)$ and $y(k)$ is:

$$\begin{aligned} x_1(k) &= u(k) \\ x_2(k) &= \max(x_1(k), x_2(k-1) + d) = x_1(k) \oplus x_2(k-1) \otimes d \\ y(k) &= x_2(k) + d = x_2(k) \otimes d. \end{aligned} \quad (2.6)$$

If the buffer has a finite storage capacity, lots arriving at the workstation cannot always be accepted by the buffer, because the storage capacity cannot be exceeded. In the max-plus algebraic description of the workstation, the buffer capacity of $N \in \mathbb{N}$ lots must be incorporated:

$$\begin{aligned} x_1(k) &= \max(x_2(k-N), u(k)) = x_2(k-N) \oplus u(k) \\ x_2(k) &= \max(x_1(k), x_2(k-1) + d) = x_1(k) \oplus x_2(k-1) \otimes d \\ y(k) &= x_2(k) + d = x_2(k) \otimes d. \end{aligned} \quad (2.7)$$

Modelling a batch machine with fixed batch size is more difficult in max-plus algebra. Since the number of batches that are produced does not equal the number of lots anymore (if the batch size is bigger than one), the event counter k cannot be used straightforwardly anymore. Suppose that the fixed batch size equals two lots. There always exists a ‘first lot’ and a ‘second lot’ in the batch. These are modelled as different lot types. The buffer receives the artificial two lot types and both lots must have arrived before the machine can start processing the batch. Let $u_1(k)$ and $u_2(k)$ be the times instants lots of type 1 and 2 arrive for the k -th time. Variables $x_1(k)$ and $x_2(k)$ are the time instants the lots enter the buffer for the k -th time and $x_3(k)$ is the time instant the batch machine starts processing a batch for the k -th time. Variable $y(k)$ is the time instant the k -th batch leaves the system. The max-algebraic description of the workstation with a batch machine can now be written as in (2.8). An additional requirement on the input signals for the batch forming is that $u_1(k) \leq u_2(k) \leq u_1(k+1)$.

$$\begin{aligned} x_1(k) &= u_1(k) & &= u_1(k) \\ x_2(k) &= u_2(k) & &= u_2(k) \\ x_3(k) &= \max(x_1(k), x_2(k), x_3(k-1) + d) = x_1(k) \oplus x_2(k) \oplus x_3(k-1) \otimes d \\ y(k) &= x_3(k) + d & &= x_3(k) \otimes d. \end{aligned} \quad (2.8)$$

Max-plus algebraic models can be written in max-plus linear state space form, similar to conventional linear state space models. Max-plus linear state space models are of the form:

$$\begin{aligned} \mathbf{x}(k) &= \mathbf{A} \otimes \mathbf{x}(k-1) \oplus \mathbf{B} \otimes \mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C} \otimes \mathbf{x}(k). \end{aligned} \quad (2.9)$$

In cases where no confusion is possible, the \otimes sign can be omitted, similar to omitting the multiplication symbol in conventional algebra. A max-plus linear state space model for the workstation with infinite buffer capacity and batch machine (2.8) is:

$$\begin{aligned} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} &= \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d \end{bmatrix} \begin{bmatrix} x_1(k-1) \\ x_2(k-1) \\ x_3(k-1) \end{bmatrix} \oplus \begin{bmatrix} 0 & \varepsilon \\ \varepsilon & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \\ y(k) &= \begin{bmatrix} \varepsilon & \varepsilon & d \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}. \end{aligned} \quad (2.10)$$

Note that if one is only interested in $y(k)$, variables $x_1(k)$ and $x_2(k)$ are redundant, since $y(k)$ only depends on $x_3(k)$ and input signals $\mathbf{u}(k)$, which (in turn) are not dependent on $x_1(k)$ and $x_2(k)$. If the workstation has a finite buffer capacity, variables $x_1(k)$ and $x_2(k)$ are not redundant.

Example 2.1. In (2.7) the max-plus algebraic equations for a workstation consisting of a buffer with finite storage capacity and single-lot machine have been presented. Now an example of a realization is given. Suppose that the buffer capacity $N = 2$ lots and the process time of a lot on the machine takes one time unit, i.e. $d = 1$.

The arrival times of lots at the workstation are stacked in vector $\tilde{\mathbf{u}}$ where $u(k) = \tilde{\mathbf{u}}_k$, the k -th element of $\tilde{\mathbf{u}}$. Vector $\tilde{\mathbf{u}} = [0 \ 1 \ 1.5 \ 2 \ 2.5 \ 2.5 \ 7 \ 7 \ 7.5 \ 7.5]^\top$. The initial condition for the max-plus model is set to $\mathbf{x}(0) = [\varepsilon \ \varepsilon]^\top$, informally meaning that no lots have ever entered the workstation before. Figure 2.3 shows a lot-time diagram of the lots as they flow through the workstation. On the horizontal axis, running time is shown, while the vertical axis shows the lot number. A boxed B and M mean that a lot is in the buffer or on the machine respectively, whereas a boxed q means that the lot is queued in front of the workstation, since the buffer is full then. In Figure 2.4 the values of $x_1(k)$ and $y(k)$ are shown for the given $u(k) = \tilde{\mathbf{u}}_k$. Values of $x_2(k)$ have been omitted for clarity reasons. In this figure, it can be seen that $x_1(6) = 3 \neq u(6) = 2.5$, also showing that the sixth lot cannot be accepted by the buffer upon arrival.

In this section, the max-plus algebra has been used to model basic manufacturing systems. The max-plus model is an event domain model, where the variables take values from a continuous domain: time instants at which the respective event occurs. Note that this is not exclusive for the max-plus model. It is possible to use max-plus algebra in time domain with discrete signal values, but this is usually not the case in modelling manufacturing systems. Closely related to max-plus algebra is the min-plus algebra, which is discussed in the next section. The relation between max-plus and min-plus models of manufacturing systems and how these models can be translated into each other is discussed in Chapter 3.

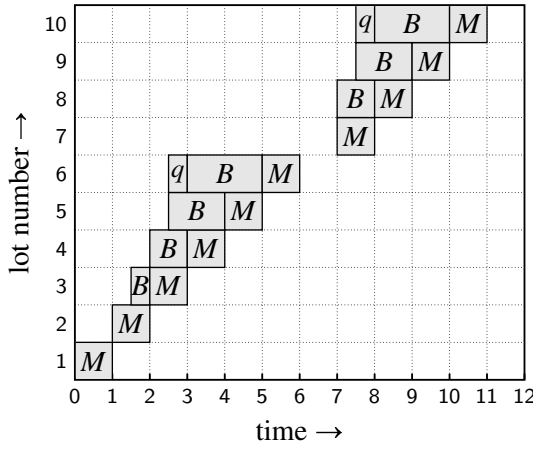


Figure 2.3: Lot-time diagram example 2.1.

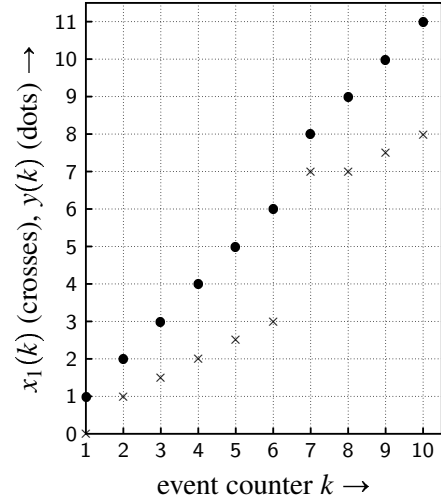


Figure 2.4: Values of max-plus realization.

2.1.2 Min-plus models

Min-plus algebra

Similar to max-plus algebra (explained in the previous section) another algebra exists in which (manufacturing) systems can be modelled: min-plus algebra. The min-plus algebra consists of the structure $\mathbb{R}_{\min} = (\mathbb{R}_{\infty}, \ominus, \otimes)$ with $\mathbb{R}_{\infty} \equiv \mathbb{R} \cup \{\infty\}$. Operations \ominus and \otimes are called min-plus-algebraic addition and min-plus-algebraic multiplication, defined as follows:

$$a \ominus b = \min(a, b) \quad (2.11a)$$

$$a \otimes b = a + b \quad (2.11b)$$

with $a, b \in \mathbb{R}_{\infty}$. Min-plus algebraic matrix subtraction and multiplications can again be defined in a similar way as in conventional linear algebra. If $\mathbf{A}, \mathbf{B} \in \mathbb{R}_{\infty}^{m \times n}$ and $\mathbf{C} \in \mathbb{R}_{\infty}^{n \times p}$:

$$(\mathbf{A} \ominus \mathbf{B})_{ij} = a_{ij} \ominus b_{ij} = \min(a_{ij}, b_{ij}) \quad (2.12a)$$

$$(\mathbf{A} \otimes \mathbf{C})_{ij} = \bigoplus_{k=1}^n a_{ik} \otimes c_{kj} = \min_{k=1 \dots n} a_{ik} + c_{kj} \quad (2.12b)$$

for all i, j . A more detailed introduction into min-plus algebra can be found in [6].

Modelling a workstation in min-plus algebra

Consider again the workstation consisting of a buffer and a machine of Figure 2.2. Another way of modelling the workstation is by means of min-plus algebra, where time instants at which events occur are not to be modelled (like in the max-plus algebra), but the number of events that have occurred at a certain time instant. Note however that similar to the max-plus algebra, the min-plus algebra is not exclusively used in time domain with real valued signals.

A workstation with infinite buffer capacity and a single-lot machine is modelled in min-plus

algebra. Define $u(t) \in \mathbb{Z}_\infty$ (integer subset of \mathbb{R}_∞) as the number of lots that have arrived at the workstation until (and including) time t . Variables $x_1(t)$ and $x_2(t) \in \mathbb{Z}_\infty$ denote the number of lots that have entered the buffer and the number of lots that have started on the machine at time t . The variable $y(t) \in \mathbb{Z}_\infty$ is the number of lots that have left the workstation at time t . The min-plus algebraic relations for this workstation now become:

$$\begin{aligned} x_1(t) &= u(t) \\ x_2(t) &= \min(x_1(t), x_2(t-d) + 1) = x_1(t) \ominus x_2(t-d) \otimes 1 \\ y(t) &= x_2(t-d) = x_2(t-d). \end{aligned} \quad (2.13)$$

The workstation can also be modelled in min-plus algebra when a buffer with finite capacity is involved. Lots that arrive at the workstation might not be accepted immediately. The capacity of the buffer is N lots. The min-plus algebraic model now becomes (with the same definitions for $u(t)$, $x_1(t)$, $x_2(t)$ and $y(t)$ as in the situation with infinite buffer capacity):

$$\begin{aligned} x_1(t) &= \min(x_2(t) + N, u(t)) = x_2(t) \otimes N \ominus u(t) \\ x_2(t) &= \min(x_1(t), x_2(t-d) + 1) = x_1(t) \ominus x_2(t-d) \otimes 1 \\ y(t) &= x_2(t-d) = x_2(t-d). \end{aligned} \quad (2.14)$$

Similar to the max-plus algebraic model of a workstation that contains a batch machine, the min-plus model needs a trick to model the batching behavior. Again, difference between ‘first’ and ‘second’ lots of a batch is made (for a batchsize of two lots). Therefore, $x_1(t)$, $x_2(t)$ and $x_3(t)$ are defined as the number of ‘first’ lots, the number of ‘second’ lots and the number of batches that have entered the buffer or started at the machine respectively. The min-plus algebraic model for the workstation with batch machine now becomes (with again the additional batch forming requirement $u_1(t) \geq u_2(t) \geq u_1(t) - 1$):

$$\begin{aligned} x_1(t) &= u_1(t) &= u_1(t) \\ x_2(t) &= u_2(t) &= u_2(t) \\ x_3(t) &= \min(x_1(t), x_2(t), x_3(t-d) + 1) &= x_1(t) \ominus x_2(t) \ominus x_3(t-d) \otimes 1 \\ y(t) &= x_3(t-d) &= x_3(t-d). \end{aligned} \quad (2.15)$$

Example 2.2. The min-plus algebraic model of the workstation consisting of a finite buffer and a single-lot machine (2.14) has been used to make a realization when applying a certain input signal $u(t)$. The same parameters as in the max-plus algebraic example (Section 2.1.1): process time $d = 1$ and buffer capacity $N = 2$. In Figure 2.5 an input signal $u(t)$ is shown. It corresponds to the same input series as in the max-plus example. Note that the signal is defined over the whole time interval: $t \in \mathbb{R}$ and that the signal is right-continuous, i.e. it is uniquely defined at each time instant and at signal jumps the value is determined as if coming ‘from the right’. In Figure 2.6 signals $x_1(t)$ and $y(t)$ have been plotted. Again, $x_2(t)$ has been omitted for visibility reasons. At two places in the graphs, $u(t)$ and $x_1(t)$ differ. These are the time instants that arriving lots cannot enter the buffer immediately due to lack of capacity.

Using min-plus models for manufacturing systems is discussed further in Chapter 3, where the relation with other models is investigated. These other models include max-plus models (which

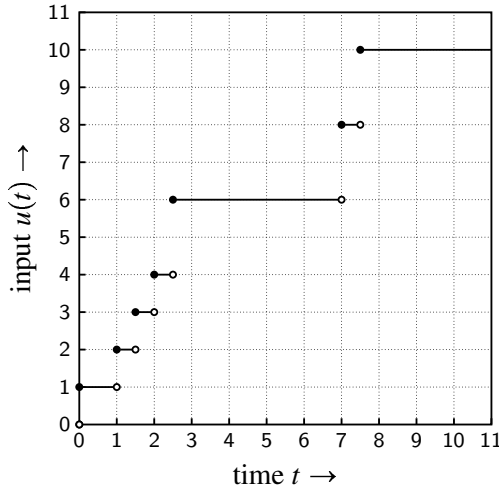
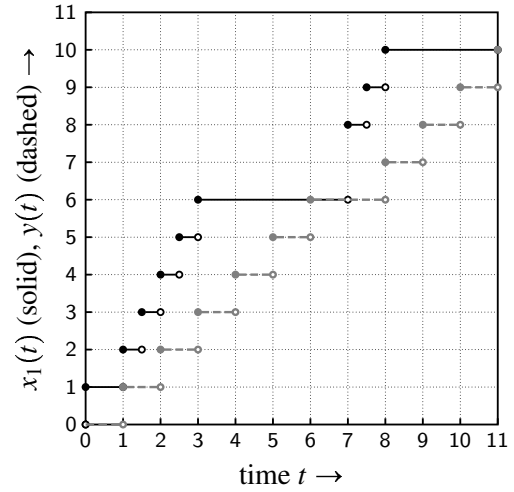
Figure 2.5: Input signal $u(t)$ evolution in time.

Figure 2.6: Values of min-plus realization.

have been discussed previously in Section 2.1.1) and hybrid process algebra models. Before hybrid process algebra models are discussed in Section 2.3.3, first *timed process algebra* models are discussed in the next section as part of the discrete event models.

2.1.3 Timed discrete event process algebra

Manufacturing systems often consist of multiple workstations in series or parallel. Distributed or parallel machines/systems can be described by *process algebra*, where *process* stands for the behavior of a system, as explained in Baeten et al. [8]. The behavior is the total of actions that takes place in the system. Since process algebra is rooted in computer science, the actions are usually discrete: they occur at a certain moment in time. The process is then called a discrete event system. A discrete event system is called *timed* (and can be described by a *timed process algebra*) if time may pass between successive events. Timed systems have been subject of many studies (e.g. see [7, 85]). Timed process algebras can be characterized by a few intuitive properties with respect to time (Hennessy and Regan [58]):

- time determinism: meaning that passage of time is deterministic. A process can reach only *one* state by performing a delay;
- instantaneous actions: time is not directly associated with communication actions. Time occurs independently. Events occur at certain moments in time;
- patience: processes wait indefinitely until they can communicate;
- maximal progress: processes communicate as soon as a possibility for a communication arises.

An example of a process algebra is formalism χ . In this section, first an introduction to formalism χ is given, then some examples of modelling a workstation are presented.

Timed process algebra: formalism χ

The χ formalism has been developed in the Systems Engineering Group of the Eindhoven University of Technology and dates back approximately ten years. The original (discrete event) simulator of Naumoski and Alberts [82] has been applied to several industrial cases, like semiconductor manufacturing, food processing industry, breweries and automobile industry. Later on, a hybrid language has been developed by Fábíán [42]. Recently, a new formal redesign of the χ formalism has taken place, which is hybrid of nature. A subset of this language is the timed process algebra as discussed in this section. The hybrid χ language is discussed in Section 2.3.3.

An introduction to process algebra χ is given in [8, 9] while the complete formal semantics is presented in Man and Schiffelers [78]. The language is explained here informally guided by the examples of modelling several different workstations in χ .

Modelling a workstation in χ

Several ways exist to model a workstation in χ , in both discrete event and hybrid form. Discrete event examples are given here, whereas hybrid models are presented in Section 2.3. The general workstation is shown (again) in Figure 2.7, in which buffer B and machine M can be distinguished. Buffer B and machine M can be modelled separately in χ using two (parallel) processes, which communicate with each other using a *channel*.

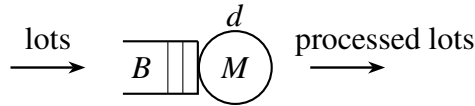


Figure 2.7: Workstation consisting of buffer B and machine M .

A χ model of workstation W consisting of buffer B with infinite storage capacity and single-lot machine M can schematically be represented by Figure 2.8. Lots arrive at the workstation via

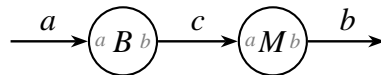


Figure 2.8: Schematic representation of χ model of workstation W .

channel a and leave the workstation via channel b , whereas communication/transport of lots between the buffer and the machine goes via channel c . The χ specification of this workstation is given in (2.16).

```

type lot = nat
proc B(chan a? : lot, b! : lot) =
  [ [ var x : lot, xs : [lot] = [ ]
  :: * ( a?; xs := xs ++ [x]
        [ len(xs) > 0 → b!hd(xs); xs := tl(xs)
        )
  ] ]
proc M(chan a?, b! : lot, val d : real) =
  [ [ var x : lot
  :: * ( a?x; Δ d; b!x )
  ] ]
model W(chan a, b : lot, val d : real) =
  [ [ chan c : lot
  :: B(a, c)
  || M(c, b, d)
  ] ]

```

(2.16)

In the model part of the specification, a , b and c can be recognized. These parameters are the *formal parameters* of the system. In (2.16), processes B and M (proc B and proc M) can be recognized. Within these processes, lots are received via channel a and sent via channel b . These are the *actual parameters* of the processes, how the channels are referred to *within* processes.

Process B represents the buffer of the workstation where incoming lots are accepted via channel a . When lots arrive at the workstation over channel a , they are stored in variable x which in turn is placed at the end of list xs . In the alternative option, when the list contains elements (i.e. the length of the list is greater than zero), lots can be sent to the machine. Condition is that the machine is willing to accept lots. Communication only takes place when the sender and receiver are able to communicate at the same time. The first lot of the list (head: hd) is sent to the machine, leaving behind the remainder (tail: tl) of the list. By placing new lots at the end of the list and sending away the front elements of the list, a first-in-first-out (FIFO) buffer has been created. The two alternatives (sending and receiving lots) are separated by the \parallel symbol.

Process M is the machine of the workstation. It accepts lots via channel a . After having received a lot (stored in variable x), it waits for the process time $d \in \mathbb{R}_+$ (in the process denoted as Δd). After the process time has been completed, the machine tries to send the lot away via channel b . Upon completion, the sequence starts over again. This looping behavior is denoted by the symbols $*$ (and). These were also present in buffer process B .

The model W connects the buffer and the machine. The \parallel symbol means that the processes in front of it and after it are executed in parallel. Process time d of the machine can be provided from an external source, for example the command line or by means of interconnection with another process.

In (2.16) lots are represented as natural numbers (in the variable type declaration). Instead of a

natural number, different, free to choose, data elements/structures can be used as lot.

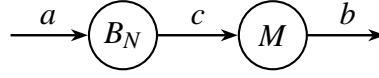


Figure 2.9: Schematic representation of χ model of workstation W with finite buffer B_N . The actual parameters inside the processes have been omitted.

In (2.17) a χ model for a workstation consisting of a buffer with finite storage capacity and a single-lot machine is presented (cf. Figure 2.9). The capacity to store lots is $N \in \mathbb{N}$ places. The major difference between (2.17) and (2.16) is the extra condition in buffer B_N for the acceptance of incoming lots. Lots can only be accepted as long as the buffer capacity is not exceeded: $\text{len}(xs) < N$.

```

type lot = nat
proc BN(chan a? : lot, b! : lot, val N : nat) =
  [[ var x : lot, xs : [lot] = [ ]
  :: * ( len(xs) < N → a?; xs := xs ++ [x]
        || len(xs) > 0 → b!hd(xs); xs := tl(xs)
        )
  ]]
proc M(chan a?, b! : lot, val d : real) =
  [[ var x : lot
  :: * ( a?x; Δ d; b!x )
  ]]
model W(chan a, b : lot, val N : nat, d : real) =
  [[ chan c : lot
  :: BN(a, c, N)
  || M(c, b, d)
  ]]

```

(2.17)

Specifications (2.16) and (2.17) represented workstations with a single-lot machine. However, if the machine is a batch machine, the specification needs to be adjusted, resulting in specification (2.18).

```

type lot = nat
, batch = [lot]
proc B(chan a? : lot, b! : batch, val N, k : nat) =
  [[ var x : lot, xs : [lot] = [ ]
  :: * ( len(xs) < N → a?; xs := xs ++ [x]
        || len(xs) ≥ k → b!take(xs, k); xs := drop(xs, k)
        )
  ]]

```



```

proc  $M(\text{chan } a?, b! : \text{batch}, \text{val } d : \text{real}) =$ 
  [[ var  $x : \text{batch}$ 
  :: * (  $a?x; \Delta d; b!x$  )
  ]]
model  $W(\text{chan } a : \text{lot}, b : \text{batch}, \text{val } N, k : \text{nat}, d : \text{real}) =$ 
  [[ chan  $c : \text{batch}$ 
  ::  $B(a, c, N, k)$ 
  ||  $M(c, b, d)$ 
  ]]

```

(2.18)

An additional data type is specified: *batch*, which is a list of lots. The machine accepts complete batches (in this case with fixed batch size k). The list in buffer B containing all lots is split up by the function *drop*, which literally ‘drops’ a natural number of lots from the buffer list.

Three different discrete event modelling methods (max-plus, min-plus and χ) have been presented so far. The classification of modelling techniques resulted in another two groups of models: continuous models of manufacturing systems and hybrid models. The former is discussed in the next section, while the latter is discussed in Section 2.3.

2.2 Continuous models

The previous section dealt with purely discrete event models of manufacturing systems. A different way to approach the modelling problem is by looking at the system from a higher level of abstraction. Assume that individual lots cannot be distinguished. This occurs when fluids are processed, e.g. in an oil refinery. From a different point of view: it might not be necessary to distinguish all individual lots, for example in mass production facilities of discrete items. If a very large number of products is in the system (e.g. in semiconductor industry, food processing, pharmaceuticals fabrication or other mass consumables), the stream of lots through the system can be regarded as a fluid, when looking at it from a distance. A workstation can then be interpreted as a fluid tank which is filled at a certain *input rate* and a valve which is able to empty the fluid tank at a certain *process rate*. Summarizing, when dealing with fluids or discrete items that can be approximated as a fluid, models of the system with *continuous variables* can be developed, cf. the introductory part of this chapter. This section gives a short (not exhaustive) overview of models based on ordinary differential equations (ODEs), also called *fluid* models. First, a continuous time standard fluid model is introduced, which displays a problem with the time delay due to the process time of machines. A few different solutions to this problem are then presented: sampling the continuous time system resulting in a discrete time system, using approximation methods for the time delay, or adding discrete event dynamics. The first two solutions are treated in this section, whereas the third solution is treated in Section 2.3, where hybrid model types are introduced.

2.2.1 Fluid models

Fluid models are continuous models based on ordinary differential or difference equations. An often used form of a fluid model in continuous time is:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned}\tag{2.19}$$

where $\mathbf{u}(t)$ is an $m \times 1$ vector containing the input variables, $\mathbf{x}(t)$ is an $n \times 1$ vector containing *state* elements and $\mathbf{y}(t)$ is a $p \times 1$ vector containing output (measured) variables. The matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times m}$, $\mathbf{C} \in \mathbb{R}^{p \times n}$ and $\mathbf{D} \in \mathbb{R}^{p \times m}$, are the *system matrices* of the fluid model.

Modelling a workstation with a standard fluid model

Consider the workstation plus extra buffer as shown in Figure 2.10. The (manipulative) infeed rate is $u_0(t) \in \mathbb{R}_+$ (lots per time unit), while the process rate is $u_1(t) \in \mathbb{R}_+$ (lots per time unit). The process rate has a lower bound ($u_1 \geq 0$) to prevent lots flowing backwards and an upper bound ($u_1 \leq \mu$). Parameter $\mu \in \mathbb{R}_+$ represents the maximum process rate. The amount of lots in buffer B_1 at time t is denoted as $x_1(t)$. Further downstream, finished products are stored in buffer B_2 , which has buffer level $x_2(t)$. The buffer levels are subject to constraints: $x_1(t) \geq 0$ and $x_2(t) \geq 0$, meaning that buffer contents cannot have a negative value.

The general idea of modelling manufacturing systems with fluid models is that the change of a buffer level equals the inflow rate minus the outflow rate. This idea was introduced by Kimemia and Gershwin [65]. Some of the hybrid models that are presented in Section 2.3 use these dynamics for the continuous part of the hybrid dynamics.

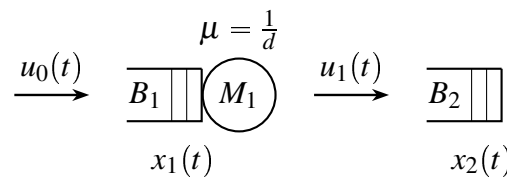


Figure 2.10: Schematic representation of fluid model with input rate $u_0(t)$, process rate $u_1(t)$ and buffer levels $x_1(t)$ and $x_2(t)$.

Remark 2.3. Instead of *buffer levels*, it is better to regard x_i as *work in process levels* of a workstation. The machines typically do not hold lots when processing, they just transfer lots from buffer to buffer. In fact, lots are on a machine for a while. Therefore, in continuous models without space to store lots on machines, x_i should be interpreted as a work in process (wip) level.

Work in process levels x_1 and x_2 are measured as outputs while infeed rate u_0 and process

rate u_1 are the inputs. The fluid model (2.19) now becomes:

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} \\ \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \end{aligned} \quad (2.20)$$

in which the general idea of Kimemia and Gershwin [65] can be recognized in matrix **B**: change of wip level equals inflow rate minus outflow rate. Note that multiple workstations can easily be connected in a flow line by making the outflow of the one workstation the inflow of the next workstation.

Example 2.4. Linear system (2.20) is evaluated in a simulation with initial condition $\mathbf{x}(0) = [x_1(0) \ x_2(0)]^T = [0 \ 0]^T$, constant input rate $u_0(t) = 1$ [lot/hour] and constant maximal process rate $\mu = 1$ [lot/hour]. The machine processes in a greedy manner, i.e. always at the highest possible rate. If the buffer is empty, then lots are processed at their arrival rate at the workstation, provided that this rate is lower than the maximum rate μ :

$$u_1(t) = \begin{cases} \mu & \text{if } y_1(t) > 0 \\ \min(u_0(t), \mu) & \text{if } y_1(t) = 0. \end{cases} \quad (2.21)$$

The wip levels evolve in time as shown in Figure 2.11. Buffer 1 remains empty: lots that arrive can immediately be processed at rate u_0 (which is equal to μ here). Another evaluation is done with constant $u_0(t) = \frac{1}{2}$, $\mu = 1\frac{1}{2}$ and initial wip levels $\mathbf{x}(0) = [x_1(0) \ x_2(0)]^T = [5 \ 0]^T$. The results of this evaluation are shown in Figure 2.12. The graph shows that first buffer 1 is emptied at full rate μ after which the machine processes the arriving lots at their arrival rate u_0 . An observation that can be made looking at the graphs is that immediately at time $t = 0$, lots come out of the machine and are stored in buffer B_2 . In reality, it takes time before the first lot arrives at this buffer: the process time of the machine $1/\mu$.

In the previous example, it was shown that the time delay caused by the process time of a machine is not modelled explicitly in the standard fluid model. Several ways exist to model this time delay in a linear system form as (2.19). Two possibilities are described briefly by Lefebvre [70]. One way is to model the time delay explicitly in the following form:

$$\dot{x}_1(t) = u_0(t) - u_1(t) \quad (2.22a)$$

$$\dot{x}_2(t) = u_1(t - \frac{1}{\mu}). \quad (2.22b)$$

This is a linear model which cannot be dealt with easily by standard linear control theory. The time delay makes this system infinitely dimensional, which does not make a control engineer's life easier. Infinitely dimensional linear systems are discussed extensively in Curtain and Zwart [29]. This dimensionality problem also reveals itself in Chapter 3 of this thesis, where it is discussed in more detail.

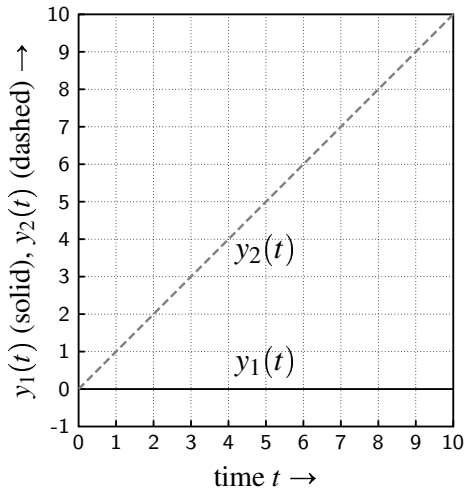


Figure 2.11: Evolution of wip levels $y_1(t)$ and $y_2(t)$ with $u_0(t) = 1$ and $\mu = 1$.

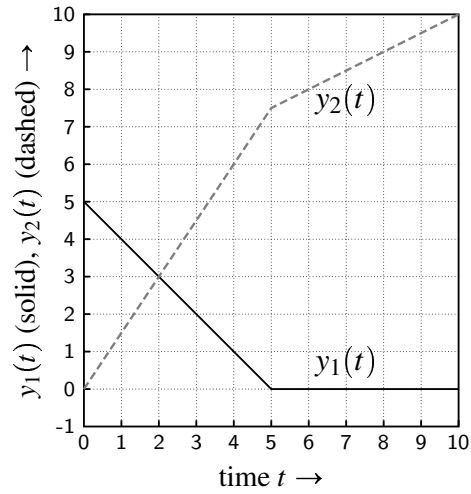


Figure 2.12: Evolution of wip levels $y_1(t)$ and $y_2(t)$ with $u_0(t) = \frac{1}{2}$ and $\mu = 1\frac{1}{2}$.

A consequence of the delay problem in linear system (2.20) is that the time span lots reside in the system (the so-called *flow time*, *sojourn time*, *cycle time* or *throughput time*) cannot be measured or computed anymore. With the presented model and constraints, it is possible to achieve the desired throughput with zero inventory. The model is only able to measure throughputs. With zero work in process, Little's law (as explained in Chapter 1) is useless to determine the flow time accurately: it would result in flow times of zero.

A way to overcome the dimensionality problem in the continuous time domain is to use approximations for the time delay. A well-known method is the use of Padé approximations [109]. This method implies expanding a function as a ratio of two power series. The higher the order of the power series, the better the approximation. However, raising the order of the Padé approximation results in a higher order linear system and therefore more complicated controllers. Another disadvantage of using Padé approximations is that the trajectory fluctuates slightly around a certain value for the duration of the time delay. For the manufacturing example this means that starting with empty buffers results in fluctuating trajectories around zero. The constraints on the system are that wip levels are not allowed to become negative. The Padé approximation therefore causes some numerical problems in the constrained environment. The Padé approximation for modelling the time delay has been elaborated in [35, 70, 71].

Another way to deal with the dimensionality problem present in (2.22) is by means of sampling the system, see Åström and Wittenmark [5]. At fixed timesteps the model is sampled, meaning that it is assumed that between successive samples, the signals are constant. A continuous time system without time delays

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (2.23)$$

can be sampled as:

$$\mathbf{x}(kh + h) = \Phi\mathbf{x}(kh) + \Gamma\mathbf{u}(kh) \quad (2.24)$$

in which k is an integer counter and h denotes the sampling period. Note that (2.22a) belongs

to the category of systems that can be sampled in this way. Matrices Φ and Γ are given by:

$$\Phi = e^{Ah} \quad (2.25)$$

$$\Gamma = \int_0^h e^{As} ds \mathbf{B}. \quad (2.26)$$

Next consider a system which consists of continuous time dynamics and time delayed inputs:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t - \tau) \quad (2.27)$$

in which τ represents the time delay. Note that (2.22b) belongs to this category. When the time delay is smaller than sampling period h , the sampled delayed model is given by:

$$\mathbf{x}(kh + h) = \Phi \mathbf{x}(kh) + \Gamma_0 \mathbf{u}(kh) + \Gamma_1 \mathbf{u}(kh - h) \quad (2.28a)$$

in which

$$\Phi = e^{Ah} \quad (2.28b)$$

$$\Gamma_0 = \int_0^{h-\tau} e^{As} ds \mathbf{B} \quad (2.28c)$$

$$\Gamma_1 = e^{A(h-\tau)} \int_0^{\tau} e^{As} ds \mathbf{B}. \quad (2.28d)$$

For systems where the time delay τ is greater than the sampling period, a different solution is obtained. Suppose that the time delay consists of two parts:

$$\tau = (d - 1)h + \tau' \quad 0 < \tau' \leq h \quad (2.29)$$

where d is an integer variable. The following sampled system equation is obtained:

$$\mathbf{x}(kh + h) = \Phi \mathbf{x}(kh) + \Gamma_0 \mathbf{u}(kh - dh + h) + \Gamma_1 \mathbf{u}(kh - dh) \quad (2.30)$$

where Φ , Γ_0 and Γ_1 are given by (2.28b)–(2.28d) in which τ is replaced by τ' . The state space description of this system contains $d \cdot r$ extra elements (with r denoting the number of input signals) to serve as an internal memory for the time delayed signals over the d sampling periods. The state space representation then looks like:

$$\begin{bmatrix} \mathbf{x}(kh + h) \\ \mathbf{u}(kh - dh + h) \\ \vdots \\ \vdots \\ \mathbf{u}(kh - h) \\ \mathbf{u}(kh) \end{bmatrix} = \begin{bmatrix} \Phi & \Gamma_1 & \Gamma_0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & I & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(kh) \\ \mathbf{u}(kh - dh) \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{u}(kh - 2h) \\ \mathbf{u}(kh - h) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ I \end{bmatrix} \mathbf{u}(kh). \quad (2.31)$$

The results for non-delayed inputs and delayed inputs can be combined in one state space description.

Remark 2.5. For the standard fluid model in continuous time with time delays as presented in (2.22), matrix \mathbf{A} is zero, so $e^{\mathbf{A}h}$ is the identity matrix, resulting in fairly simple expressions for Φ , Γ_0 and Γ_1 .

The sampling method, resulting in a discrete time fluid model, takes away the dimensionality problem. An intuitive interpretation of this is as follows: in the continuous time case, the delayed signal must be incorporated in the model for the duration of the time delay, serving as an internal memory. As the evolution of the delayed signal(s) is unknown, infinitely many realizations are possible, yielding an infinitely dimensional system. In a sampled system, it is assumed that between successive samples, signals are constant. The time delay consists of a finite number of sampling periods. Only for this number of samples, the signal values need to be stored in the internal memory, resulting in a finite dimensional model of the system. Another way of understanding the dimensionality issue is that, loosely speaking, a continuous model is the limit of a discrete model where the sampling period goes to zero. The number of samples within a fixed time delay duration grows to infinity then, which makes (2.31) infinitely dimensional.

A disadvantage of sampling is that for a manufacturing flow line, a wide variety of time delays can occur within the flow line. To keep realistic outcomes, the sampling period should be chosen small enough, resulting in a rather large state space, which makes controller synthesis for these typically MIMO (multi-input, multi-output) systems more difficult.

The fluid models are all based on ordinary differential equations (ODEs). Another way of continuous modelling is by means of partial differential equations (PDEs). These PDE models, also called *flow models* are treated in the next section. A completely different way of overcoming the time delay problem present in the standard continuous time fluid model is to add discrete event dynamics, resulting in a hybrid model. These models are treated in Section 2.3.

2.2.2 Flow models

The model presented in the previous section has a few disadvantages: first, cycle times are not computable and second, time delays (e.g. transportation or process times) are not easy to implement in a convenient way. The fluid models essentially were ODE (ordinary differential equations) models. Another stream of literature is based on partial differential equations (PDEs). As mentioned in the introductory chapter, traffic flow modelling shows similarities to manufacturing systems modelling. In the 1950s, Lighthill and Whitham [74] and Richards [91] proposed a first order fluid model (known as the LWR model) to describe the dynamics of traffic flow. The *first order* characterization of this model means (loosely speaking) that only first order partial derivatives of a variable are present in the (set of) equations.

Traffic flow theory makes use of basic principles, like mass conservation. Consider the density of cars at a point x in space and at time t . This density is denoted by $\rho(x, t)$ [cars/meter]. The

speed of the cars is $v(x, t)$ [meters/second] and the flow of cars is denoted by $q(x, t)$ [cars/second]. A basic relationship exists between these three variables:

$$q(x, t) = \rho(x, t) \cdot v(x, t). \quad (2.32)$$

This equation embodies the PDE version of Little's law [75], (see Chapter 1).

The number of cars needs to be conserved on a road (without on-ramps and off-ramps). This mass conservation principle can be expressed as: the change in the number of cars over a distance Δx in time equals the difference between inflow and outflow (cf. the fluid model of Kimemia and Gershwin). In other words:

$$\frac{\partial}{\partial t} \int_0^{\Delta x} \rho(x, t) dx = q(t)_{(x=0)} - q(t)_{(x=\Delta x)}. \quad (2.33)$$

In differential form, this equation becomes:

$$\frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial x} = 0. \quad (2.34)$$

These two relations (2.32) and (2.34) are the basic relations for traffic flow. However, three variables are used: q , v and ρ . Therefore, a third relation is needed. The LWR model uses a fixed relation between speed v and density ρ . This is an assumption next to the two fundamental relations (2.32) and (2.34).

Higher order PDE models have been proposed in the 1970s. The static relation between speed and density was replaced by an other PDE. This is often a momentum conservation law (resulting in a Payne-type model [86]), with the following form:

$$\frac{\partial v}{\partial t} + v \frac{\partial v}{\partial x} = \frac{\text{sum of internal and external forces}}{\rho(x, t)}. \quad (2.35)$$

The left hand side of this equation is the total time derivative of v , consisting of the the local acceleration noticed by a standing observer and the convection acceleration, describing the change in the mean velocity due to in- and outflowing cars with different speeds. The right hand side of (2.35) is composed of all factors that influence the change of velocity in time and space. Payne developed a model himself, but in [30] Daganzo pointed out some serious problems with that model type, e.g. unallowed backward travel of cars. Most literature after [30] do not suffer from this problem anymore. For a quick overview of these models, the reader is referred to [88]. A third category of traffic flow models are the Helbing-type models, see [55, 57, 84]. These models have a third PDE describing the dynamics and velocity variance. The latter is a possible term in the right hand side of (2.35). A hierarchy of PDE models for supply chains and possibly re-entrant networks is studied by Armbruster et al. in [2–4], where ideas from both gas dynamics and traffic flow theory are incorporated into the PDEs. A very thorough overview of modelling traffic flow is presented in Helbing [56].

As stated in the introductory chapter, insights, notions as well as modelling techniques from traffic flow theory might be used for manufacturing control. The analogy is that cars can be

substituted by products or lots, and road intersections can be regarded as multi-product servers (with or without setup times). Even without a great imagination, the analogy between traffic and manufacturing phenomena is quite obvious. In Chapter 5 of this thesis, an example of a road crossing is given that relates closely to a workstation that serves two product streams. The model type in that chapter is a hybrid model. This type of models is treated in the next section.

2.3 Hybrid models

Hybrid models contain both continuous (analog) and discrete (logic/events) dynamics. Events can make continuous variables jump from one value to the other. It is also possible that events do not cause a jump in the continuous variable, but a change of a discrete variable. This in turn may have a subsequent effect on the evolution of a continuous variable. Examples of hybrid systems are traffic control systems (continuous car flows with discrete ramp metering), beer breweries (continuous beer flow which is to be bottled in discrete bottles), temperature control systems (continuous temperature and discrete on/off switches), airplane coordination control (continuous speed, yaw, roll and pitch with discrete steering actions) and chemical plants (continuous chemical reaction kinetics and discrete event recipes). Hybrid systems are encountered in almost every branch of industry. Even when the hybrid nature of a system is not obvious at first sight, when the discrete material flow through a manufacturing system is modelled with a continuous approximation (as in the previous section) and the control actions remain discrete, a hybrid model has been obtained.

Several forms of hybrid system modelling paradigms/frameworks exist. The choice between the modelling paradigms is often a trade-off between modelling power on the one hand and decision power and tractability on the other hand. A fine introduction of a philosophical nature to this trade-off and different modelling techniques is presented by Boel et al. in [17]. During the last two decades, an enormous amount of literature on hybrid systems has appeared and currently it is still a hot topic in not only the systems and control community, but also in operations research, economics and management science.

This section first introduces a hybrid system modelling framework of discrete hybrid automata. Then a hybrid fluid model is presented in Section 2.3.2. Hybrid process algebra models using the hybrid χ formalism are studied in Section 2.3.3.

2.3.1 Discrete hybrid automata

A large class of hybrid systems can be represented as *discrete hybrid automata* (DHA). A discrete hybrid automaton is the interconnection of a finite state machine and a switched affine system through a mode selector and an event generator. An overview of discrete hybrid automata is given by Geyer et al. [49], in which the aforementioned components are described in more detail. DHAs are formulated in the discrete-time domain. They generalize many com-

putation oriented models for hybrid systems and therefore represent a universal starting point for solving complex analysis and synthesis problems for hybrid systems. Examples of modelling paradigms for DHAs for which analysis and control techniques have been developed are piecewise affine systems (PWA) and mixed logical dynamics (MLD). Piecewise affine systems are described by Sontag [102]. Mixed logical dynamics systems are introduced in Bemporad and Morari [14]. The two representations (PWA and MLD) are equivalent for a certain class of systems, as shown by Heemels et al. in [52]. Other model representations within the DHA framework are linear complementarity systems (LC), extended linear complementarity systems (ELC) and max-min-plus-scaling systems (MMPS). Each system representation has its own advantages and disadvantages, as described in Bemporad [11]. In the next subsection, a workstation is modelled in the MLD and PWA representations.

Remark 2.6. The max-plus and min-plus algebraic models as described in Sections 2.1.1 and 2.1.2 are subsets of the max-min-plus-scaling modelling framework. In these subsets, the hybrid nature has been lost.

A continuous model of a manufacturing system has been presented in Section 2.2.1, based on the work of Kimemia and Gershwin [65]. In Figures 2.11 and 2.12 it was shown that the delay due to the process time of a machine is not incorporated in this model type. A possible way to overcome this is to use approximation techniques, like the Padé approximation. A different way of overcoming the delay issue is presented here in a hybrid model. The idea behind this model was first presented in [35] and was also used in [70] by Lefeber.

Modelling a workstation as a discrete hybrid automaton

Consider again the workstation, consisting of a buffer and a single-lot machine (Figure 2.13) with the linear dynamics of (2.20) and the constraints on the wip levels: $x_1 \geq 0$ and $x_2 \geq 0$. The delay problem of the continuous fluid model was that lots immediately come out of a machine as soon as they are available in the buffer. However, in reality the machine can only start processing a lot when a complete lot has entered the workstation. In other words, the machine can only start processing a lot when the workstation contains at least one lot. In addition to this, the machine is allowed to process lots if the wip level is smaller than one, provided that the inflow is zero. Otherwise the buffer would never be emptied completely. The constraint on the process rate u_1 now becomes as in (2.36) (note that k represents the discrete time).

$$u_1(k) = \begin{cases} \mu & \text{if } x_1(k) \geq 1 \\ \mu & \text{if } 0 < x_1(k) \leq 1 \text{ and } u_0(k) = 0 \\ 0 & \text{if } 0 < x_1(k) \leq 1 \text{ and } u_0(k) > 0 \\ 0 & \text{if } x_1(k) = 0. \end{cases} \quad (2.36)$$

A discrete hybrid automaton has been obtained. It consists of the linear continuous dynamics of (2.20), wip level constraints and the logic relations given in (2.36). The model is now casted in MLD and PWA form.

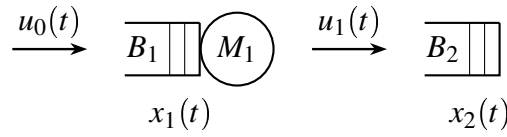


Figure 2.13: Schematic representation of a workstation with lot input rate $u_0(t)$, process rate $u_1(t)$ and buffer levels $x_1(t)$ and $x_2(t)$.

Mixed logical dynamical systems

In Bemporad and Morari [14] a class of systems has been introduced, containing logic, dynamics and constraints. The general form of this mixed logical dynamical system (MLD) is:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}_1\mathbf{u}(k) + \mathbf{B}_2\delta(k) + \mathbf{B}_3\mathbf{z}(k) \quad (2.37a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}_1\mathbf{u}(k) + \mathbf{D}_2\delta(k) + \mathbf{D}_3\mathbf{z}(k) \quad (2.37b)$$

$$\mathbf{E}_2\delta(k) + \mathbf{E}_3\mathbf{z}(k) \leq \mathbf{E}_1\mathbf{u}(k) + \mathbf{E}_4\mathbf{x}(k) + \mathbf{E}_5 \quad (2.37c)$$

where $\mathbf{x}(k) = [\mathbf{x}_r^T(k) \quad \mathbf{x}_b^T(k)]^T$ with $\mathbf{x}_r(k) \in \mathbb{R}^{n_r}$ and $\mathbf{x}_b(k) \in \{0, 1\}^{n_b}$ representing vectors of n_r real and n_b boolean variables respectively. Vectors $\mathbf{y}(k)$ and $\mathbf{u}(k)$ have a similar structure and vectors $\delta(k) \in \{0, 1\}^{r_b}$ and $\mathbf{z}(k) \in \mathbb{R}^{r_r}$ contain r_b and r_r auxiliary boolean and real variables. The inequalities in (2.37c) have to be interpreted componentwise. Evaluating an MLD model in general means that a mixed integer linear program needs to be solved in order to compute the values of the boolean and real auxiliary variables. For systems with a lot of boolean variables, this can be a time consuming procedure. Translating the model into a PWA model (see the next part of this section) might improve the evaluation speed then.

To model the workstation as an MLD system, boolean variables are introduced to deal with the logic relations in (2.36):

$$\delta_1(k) = (x_1(k) \geq 1) \quad (2.38a)$$

$$\delta_2(k) = (x_1(k) \leq 0) \quad (2.38b)$$

$$\delta_3(k) = (u_0(k) \leq 0). \quad (2.38c)$$

Remark 2.7. The expression for $\delta_2(k)$ and $\delta_3(k)$ look somewhat strange: input u_0 and wip level x_1 can never be negative (as a physical constraint). In (2.36) the test $u_0(k) = 0$ is a test for strict equality and $x_1 > 0$ is a strict inequality. Strict (in)equalities cannot be incorporated in an MLD system, so the ≤ 0 test is used to check if input u_0 is zero. Requirements $u_0 \geq 0$ and $x_1 \geq 0$ are added to (2.37c). To deal with strict inequalities, parameter $\varepsilon > 0$ is used (typically in the order of the machine computing precision). Inequality $x_1 > 0$ is then stated as $x_1 \geq \varepsilon$.

An additional boolean variable $\delta_4(k)$ is introduced together with real auxiliary variable $z_1(k)$ to determine the machine process rate, according to (2.36).

$$\text{if } \delta_1 \vee \underbrace{(\neg\delta_1 \wedge \neg\delta_2 \wedge \delta_3)}_{\delta_4} \text{ then } z_1 = \mu \text{ else } z_1 = 0. \quad (2.39)$$

The resulting MLD model for the workstation, obtained via HYSDEL [103] with manually added lower bounds on $x_1(k)$ and $u_0(k)$, then becomes (gray terms are zero):

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} T_s \\ 0 \end{bmatrix} u_0(k) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} + \begin{bmatrix} -T_s \\ T_s \end{bmatrix} z_1(k) \quad (2.40a)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u_0(k) + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} z_1(k) \quad (2.40b)$$

$$\begin{bmatrix} -999 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -\varepsilon & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & -\varepsilon & 0 \\ 0 & 0 & 1000 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & -1 \\ -1 & 1 & 0 & 1 \\ -1 & 0 & -1 & 1 \\ 0 & 0 & 0 & \mu \\ 0 & 0 & 0 & -\mu \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} z_1(k) \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u_0(k) + \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1000 \\ 0 \\ 1000 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.40c)$$

in which T_s denotes the sample time of the discrete-time MLD system. Note that the input u_0 and wip level x_1 are bounded by constraints. They are not only non-negative, but also have an upper bound, here set to 1000. The upper bound is needed to be able to determine the values of $\delta(k)$.

Piecewise affine systems

The same workstation can be modelled as a piecewise affine system (PWA). PWA systems are defined by partitioning the input space and state space into polyhedral regions. With each region, a different linear state-update equation is associated. The general form of a PWA model is:

$$\mathbf{x}(k+1) = \mathbf{A}_{i(k)}\mathbf{x}(k) + \mathbf{B}_{i(k)}\mathbf{u}(k) + \mathbf{f}_{i(k)} \quad (2.41a)$$

$$\mathbf{y}(k) = \mathbf{C}_{i(k)}\mathbf{x}(k) + \mathbf{D}_{i(k)}\mathbf{u}(k) + \mathbf{g}_{i(k)} \quad (2.41b)$$

$$i(k) \text{ such that } \mathbf{H}_{i(k)}\mathbf{x}(k) + \mathbf{J}_{i(k)}\mathbf{u}(k) \leq \mathbf{K}_{i(k)}. \quad (2.41c)$$

Matrices \mathbf{A}_i , \mathbf{B}_i , \mathbf{C}_i , and \mathbf{D}_i are all of proper dimensions and the inequalities of (2.41c) should be interpreted componentwise. To evaluate a PWA system, first the current region $i(k)$ needs to be determined. In practice, this means that for all regions the inequalities are evaluated to check for feasibility. Once the current region (or regions) has been found, the state elements $\mathbf{x}(k)$ and output elements $\mathbf{y}(k)$ can be computed using the corresponding system matrices. Checking all regions for feasibility can be quicker than solving a mixed integer linear program for the (equivalent) MLD case, especially when a lot of boolean variables are involved in the MLD representation.

Instead of defining boolean variables to incorporate the logic relations of (2.36) into inequalities, the logic relations define the different modes in which the system can be. The PWA model has been obtained by means of hybrid systems description language HYSDEL [103]. The system consists of six modes. The affine equations (2.41a) and (2.41b) are identical for all modes, except for the $\mathbf{f}_{i(k)}$ vector. For each mode, this vector is given below together with the inequalities that determine the current mode.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} T_s \\ 0 \end{bmatrix} u_0(k) + \mathbf{f}_{i(k)} \quad (2.42a)$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u_0(k) + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.42b)$$

$$\text{mode 1: } \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{f}_1 = \begin{bmatrix} -\mu T_s \\ \mu T_s \end{bmatrix} \quad (2.42c)$$

$$\text{mode 2: } \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{f}_2 = \begin{bmatrix} -\mu T_s \\ \mu T_s \end{bmatrix} \quad (2.42d)$$

$$\text{mode 3: } \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{f}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.42e)$$

$$\text{mode 4: } \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{f}_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.42f)$$

$$\text{mode 5: } \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{f}_5 = \begin{bmatrix} -\mu T_s \\ \mu T_s \end{bmatrix} \quad (2.42g)$$

$$\text{mode 6: } \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{f}_6 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (2.42h)$$

Remark 2.8. The bounds on variables $x_1(k)$ and $x_2(k)$ and input $u_0(k)$ are not present in the PWA expressions, while they were necessary in the MLD representation. In order to add the bounds on the wip levels and the input rate if desired, the following inequalities should be added

to each mode:

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} u_0(k) \leq \begin{bmatrix} 1000 \\ 0 \\ 1000 \\ 0 \\ 1000 \\ 0 \end{bmatrix}. \quad (2.43)$$

If the initial buffer levels x_1 and x_2 are chosen within these bounds and the input signal u_0 that is applied lies within its bounds, these inequalities are always obeyed. These bounds, especially the lower bound on the input rate, can be necessary when the input rate becomes a manipulative input, determined by a controller. Without the lower bound, a controller might yield negative input rates, which are physically impossible.

Example 2.9. A simulation is performed for the workstation with the MLD model (2.40) (the PWA model gives the same result, because they are equivalent, cf. [52]). The input signal $u_0(t)$ is chosen piecewise linear. First lots arrive with $u_0(t) > \mu$, then the input rate is put to zero. After a while, lots start to arrive again with $u_0(t) < \mu$, and finally the input rate is put to zero again. The complete input signal $u_0(t)$ is shown in the top graph of Figure 2.14. The resulting wip levels are shown in the bottom graph of the figure. It can be seen that buffer B_2 is filled when the workstation contains at least one lot. The other phenomenon is also visible in the graph: the workstation is only emptied when the input rate is zero.

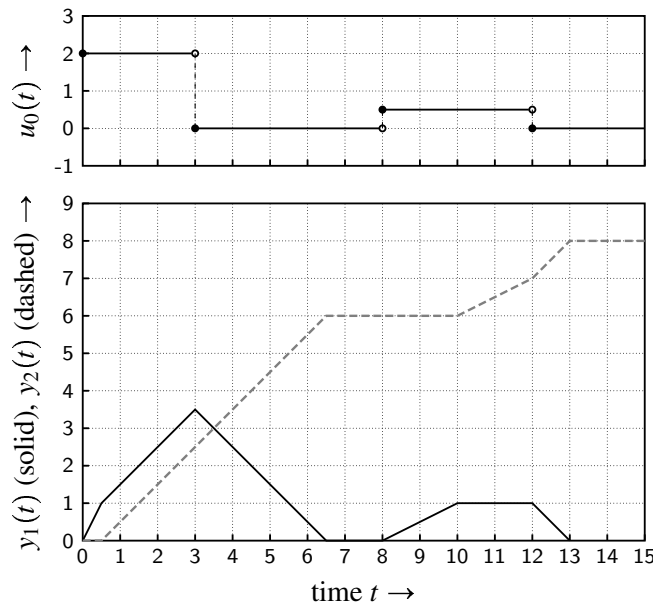


Figure 2.14: Evaluation of MLD model (2.40). Top graph: lot input rate. Bottom graph: wip level of the workstation $y_1(t)$ and cumulative output $y_2(t)$.

2.3.2 Hybrid fluid models

Consider again the linear fluid model as stated in (2.20). With a discrete event dynamics part, it is possible to incorporate dynamics that influence the nominal behavior of this linear system. One could think of scheduled and unscheduled maintenance, breakdowns, cleaning of the workstation, switching between different lot types or recipes. In the next subsection, an example is given of a *hybrid fluid model* in which a workstation is modelled that needs to be cleaned after a while. This model type is also used in Chapter 5 and Chapter 6.

Modelling a workstation with a hybrid fluid model

Consider a single machine workstation with a FIFO buffer (infinite capacity), for example the workstation as shown in Figure 2.2. The machine needs to be cleaned whenever it suits the machine operator. The wip level is denoted by x_1 and the machine maximum process rate is $\mu \in \mathbb{R}_+$ lots per hour. Lots arrive with fixed rate $\lambda < \mu$, $\lambda \in \mathbb{R}_+$ and the process rate of the machine at time t is denoted by $u_1(t) \leq \mu$, which is a manipulative input of the system. In addition to this input, another (discrete) input u_0 represents the action that needs to be performed by the workstation. The following actions are possible:

$$\begin{aligned} u_0 &= P : \text{process lots} \\ u_0 &= C : \text{perform a cleaning sequence.} \end{aligned}$$

A cleaning sequence takes a positive fixed amount of time: $c \in \mathbb{R}_+$ hours. Immediately after the machine has been cleaned, it can process lots again. State variable x_0 denotes the *remaining cleaning time* of the machine. The state vector of this system $\mathbf{x}(t)$ is therefore defined as:

$$\mathbf{x}(t) = \begin{bmatrix} x_0(t) & x_1(t) \end{bmatrix}^T \in [0, c] \times \mathbb{R}_+ \quad (2.44)$$

and the input vector of this system is defined as:

$$\mathbf{u}(t) = \begin{bmatrix} u_0(t) & u_1(t) \end{bmatrix}^T \in \{P, C\} \times [0, \mu]. \quad (2.45)$$

The inputs are bound to constraints at any moment in time:

$$\begin{aligned} u_0 &\in \{P, C\}, \quad 0 \leq u_1 \leq \mu && \text{for } x_0 = 0 \quad \text{and } x_1 > 0 \\ u_0 &\in \{P, C\}, \quad 0 \leq u_1 \leq \lambda && \text{for } x_0 = 0 \quad \text{and } x_1 = 0 \\ u_0 &= C, \quad u_1 = 0 && \text{for } x_0 > 0. \end{aligned}$$

The discrete event dynamics of this system are:

$$x_0 := c \text{ if } x_0 = 0 \text{ and } u_0 = C \quad (2.46)$$

which means that whenever the machine is processing ($x_0 = 0$) and the operator decides to perform a cleaning sequence, the remaining cleaning time is set to c . The continuous dynamics

of this system is:

$$\dot{x}_0(t) = \begin{cases} -1 & \text{if } x_0(t) > 0 \\ 0 & \text{if } x_0(t) = 0 \end{cases} \quad (2.47a)$$

$$\dot{x}_1(t) = \lambda - u_1(t). \quad (2.47b)$$

The remaining cleaning time decreases linearly over time when its value is greater than zero. Once it has reached zero, it remains zero (unless it is put to c again by the operator). The wip level x_1 follows the same dynamics as the linear fluid model in (2.20).

Remark 2.10. Although input u_0 is neither a real variable nor a boolean variable, the hybrid fluid model presented in this section might fit in the DHA framework as presented in Section 2.3.1. The possible realizations of u_0 should be translated to boolean variables then, in order to fit in a MLD structure for example. An example of such a translation is given in [14], in which qualitative outputs of a thermal system (COLD, COOL, NORMAL, WARM, ...) are translated into a set of natural variables, which in case are formed by the sum of boolean variables, fitting (2.37). The hybrid fluid model is treated separately here, since it is used extensively in other chapters in this thesis.

Example 2.11. Model (2.44)–(2.47) has been evaluated with initial state $\mathbf{x}(0) = [0 \ 2]^T$. The process policy is greedy: always process lots at the highest possible rate, which means:

$$u_1(t) = \begin{cases} \mu & \text{if } x_1(t) > 0 \\ \lambda & \text{if } x_1(t) = 0. \end{cases} \quad (2.48)$$

Furthermore, at $t = 4$ it is decided to perform a cleaning sequence, after which processing lots continues. The maximum process rate $\mu = 3$ lots/hour, while the constant arrival rate of jobs $\lambda = 2$ lots/hour. The evolution of the inputs and state elements is shown in Figure 2.15. First, the buffer is emptied at rate μ , while it is being filled at rate λ . The effective decrease rate of the wip level therefore is $\mu - \lambda$ lots/hour. At $t = 2$, the buffer is empty and the machine continues processing lots at the arrival rate (as specified in (2.48)). Then at $t = 4$ it is decided to perform the cleaning sequence. No jobs can be processed, so the process rate is zero. The wip level increases with rate λ . After the cleaning sequence, the machine starts processing the lots again with rate μ .

As shown above, the hybrid fluid model type is suitable to incorporate various (discrete event) phenomena explicitly in the dynamics of the system. For workstations with multiple lot types and switchover times between processing the lot types, a hybrid fluid model is used in Chapters 5 and 6.

2.3.3 Hybrid process algebra

In Section 2.1.3 models of a workstation have been presented which were specified in formalism χ , a process algebra. These models were completely based on events that occur in time,

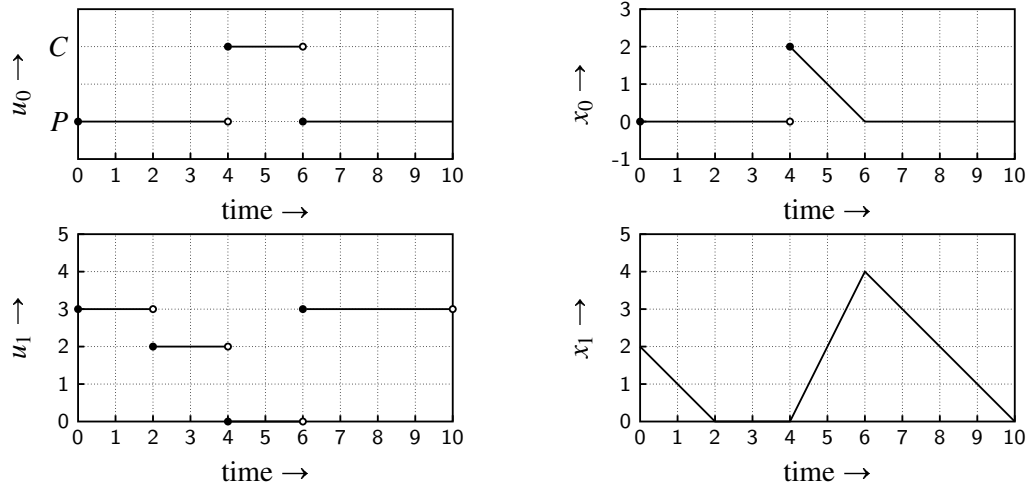


Figure 2.15: Evolution of input signals and state elements of hybrid fluid model (2.44)–(2.47).

so called *discrete event models*. It is also possible to use process algebra for modelling hybrid dynamics: systems where both events occur in time and continuous variables evolve in time. In this case, a hybrid process algebra is used. In fact, formalism χ is suited for hybrid systems modelling. In Section 2.1.3, only the discrete event part of the formalism had been used.

Hybrid process algebra: formalism χ

The hybrid χ language was originally developed by Fábíán [42] and it was redesigned in 2005/2006. This resulted in a hybrid process algebra with formal semantics, as defined in [9]. In this section, hybrid χ models of basic workstations are explained. The informal meaning of the different constructs is given along with the specifications. For the complete formal semantics of the language, the reader is referred to [9, 78].

Modelling a workstation in hybrid process algebra χ

For modelling a workstation in hybrid process algebra, a different approach is used than with discrete event process algebra. Next to the events that occur (arrival of a lot, start of a lot on the machine), time-consuming phenomena take place: processing lots. The *total* process time equals $d \in \mathbb{R}_+$ time units. The *remaining* process time of a lot that has started on the machine decreases over time until zero. If the remaining process time equals zero, the lot has been completed and the lot can be sent away. If a lot cannot be sent away immediately after processing, the remaining process time stays zero. When a new lot is started on the machine, the remaining process time is set to d again and decreases again over time. This piecewise continuous behavior can be caught in the continuous part of a hybrid model, together with events in the discrete part.

A workstation with infinite buffer capacity and a single-lot machine is modelled in hybrid pro-

cess algebra, see (2.49).

type lot = nat

```

proc W(chan a? : lot, b! : lot, val d : real) =
  |[ var x : lot, xs : [lot] = [ ], ml : lot, m : nat = 0, cont xc : real = 0
  :: *( a?x; xs := xs ++ [x]
      |[ len(xs) > 0 ∧ m = 0 → ml := hd(xs); m := 1; xs := tl(xs); xc := d
      |[ m = 1 ∧ xc = 0 → b!ml; m := 0
      )
  |[ xc > 0 ⇒  $\dot{x}_c = -1$ 
  |[ xc = 0 ⇒  $\dot{x}_c = 0$ 
  ]

```

(2.49)

The workstation is not split up anymore in a separate buffer and machine, but is modelled as a whole, as schematically depicted in Figure 2.16. Arriving lots are stored in list xs . Lots can be of any type, in this case of type natural: nat . If the machine is idle, $m = 0$. Variable ml contains the lot that is currently in process (one at a time for a single-lot machine). When a lot is put on the machine ($ml := \text{hd}(xs)$), the remaining process time $x_c \in \mathbb{R}_+$ is set to the total process time ($x_c := d$). The continuous dynamics of the workstation mean that if the remaining process time is greater than zero, it declines with slope -1 in time: $x_c > 0 \Rightarrow \dot{x}_c = -1$. When the remaining process time has reached zero, it stays zero: $x_c = 0 \Rightarrow \dot{x}_c = 0$. The arrow \Rightarrow means *implication*. In classical logic $A \Rightarrow B$ is equivalent to $\neg A \vee B$ (i.e. NOT A OR B).

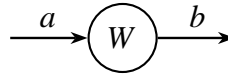


Figure 2.16: Schematic representation of hybrid χ model of workstation W .

If the buffer has limited storage capacity and the machine processes batches with a fixed batch size $k \in \mathbb{N}$, the hybrid process algebra model in χ could be as in (2.50).

type lot = nat

, batch = [lot]

```

proc W(chan a? : lot, b! : batch, val d : real, N, k : nat) =
  |[ var x : lot, xs : [lot] = [ ], m : batch = [ ], cont xc : real = 0
  :: *( len(xs) < N → a?x; xs := xs ++ [x]
      |[ len(xs) ≥ k ∧ m = [ ] → m := take(xs, k); xs := drop(xs, k); xc := d
      |[ m ≠ [ ] ∧ xc = 0 → b!m; m := [ ]
      )
  |[ xc > 0 ⇒  $\dot{x}_c = -1$ 
  |[ xc = 0 ⇒  $\dot{x}_c = 0$ 
  ]

```

(2.50)

With respect to (2.49), a constraint has been added on the acceptance of new lots in the buffer. Moreover, variable m gets k lots at a time via the take and drop command, as used earlier in the discrete event χ models in Section 2.1.3.

Remark 2.12. In cases where individual lots are not distinguishable or not to be distinguished, lists xs and m can be replaced by two counters, which represents the number of lots in the buffer and on the machine respectively. The specification of (2.50) is transformed into a specification with counters x and n . The communication between processes can now be restricted to synchronization (communication with type void). This results in the following specification:

$$\begin{aligned}
 &\text{proc } W(\text{chan } a? : \text{void}, b! : \text{void}, \text{val } d : \text{real}, N, k : \text{nat}) = \\
 &\quad \llbracket \text{var } x : \text{nat} = 0, n : \text{nat} = 0, \text{cont } x_c : \text{real} = 0 \\
 &\quad :: * (\begin{aligned} &x < N \quad \rightarrow a?; x := x + 1 \\ &\parallel x \geq k \wedge n = 0 \rightarrow n := k; x := x - k; x_c := d \\ &\parallel n > 0 \wedge x_c = 0 \rightarrow b!; n := 0 \end{aligned} \\
 &\quad) \\
 &\quad \parallel x_c > 0 \Rightarrow \dot{x}_c = -1 \\
 &\quad \parallel x_c = 0 \Rightarrow \dot{x}_c = 0 \\
 &\quad \rrbracket
 \end{aligned} \tag{2.51}$$

In this section hybrid process algebra models have been used to model a workstation. Discrete variables involved the number of lots in a buffer and the presence of lots on the machine, while a continuous variable represented the remaining processing time of the lot that was currently being processed on the machine. Intuitively, a state space representation has been introduced for a manufacturing system that is finite dimensional and incorporates the delay due to the process time. This state space representation therefore overcomes the dimensionality and delay issues that have been mentioned in Section 2.2.1. In the next chapters, this state space representation for manufacturing systems is elaborated on. Maps to other representations are investigated in Chapter 3, and feedback control methods for scheduling lots in manufacturing flow lines are developed using this state space representation in Chapter 4.

2.4 Case study: modelling a flow line

In this chapter, a variety of modelling techniques has been presented. Some models are continuous or hybrid models in which individual lots cannot be distinguished, whereas in other models individual lots can be distinguished. In this section, all models are evaluated for a flow line example. The length of the flow line does not influence the phenomena that occur in a manufacturing system, so the smallest possible flow line is investigated here: a flow line consisting of two workstations.

Consider the manufacturing flow line as shown in Figure 2.17. Two workstations exist, each consisting of a FIFO buffer with infinite capacity and a single-lot machine. At the end of the flow line, finished lots are stored in buffer B_3 . The buffer levels are denoted by $x_1(t)$,

$x_2(t)$ and $x_3(t)$, as indicated in the figure. For the continuous model and the hybrid discrete automata, the buffer levels should be interpreted as wip levels, as explained before. The constant process times of the machines are d_1 and d_2 respectively, so the maximum process rates (for the continuous and hybrid models) of the machines are $\mu_1 = \frac{1}{d_1}$ and $\mu_2 = \frac{1}{d_2}$ lots per time unit respectively. Without loss of generality, the time unit is ‘hours’. For this example, $\mu_1 = 2$ [lots/hour] and $\mu_2 = 1\frac{1}{2}$ [lots/hour]. It is assumed that the evaluation starts with an empty factory.

The rate at which lots arrive at the workstation is denoted by $u_0(t)$. To make several phenomena visible, a ‘rich’ input signal is chosen, shown in Figure 2.18. During the first hour, no lots arrive at the first workstation, so the flow line should stay in rest for one hour. Then lots start to arrive and the interarrival time decreases (rate $u_0(t)$ increases) after which a constant arrival rate is kept until $t = 7$. No lots arrive then until $t = 12\frac{1}{2}$. For four hours, one lot arrives at each hour then. At the end of the evaluation, from $t = 19$ lots arrive with an increasing interarrival time, until $t = 23$. From that time, lots do not arrive anymore. The total number of lots that arrive is $\int_0^{23} u_0(t) dt = 28$. The arrival rate profile is chosen in this way because it contains rates lower and greater than the process rates, it contains periods of no arrivals, and it contains increasing and decreasing rates.

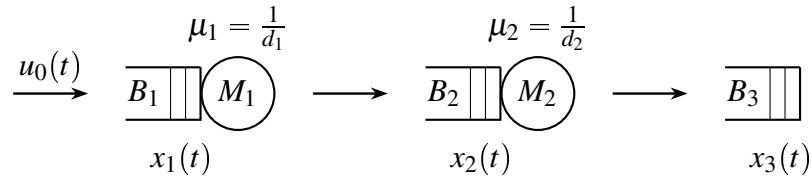


Figure 2.17: Manufacturing flow line for comparison of modelling techniques.

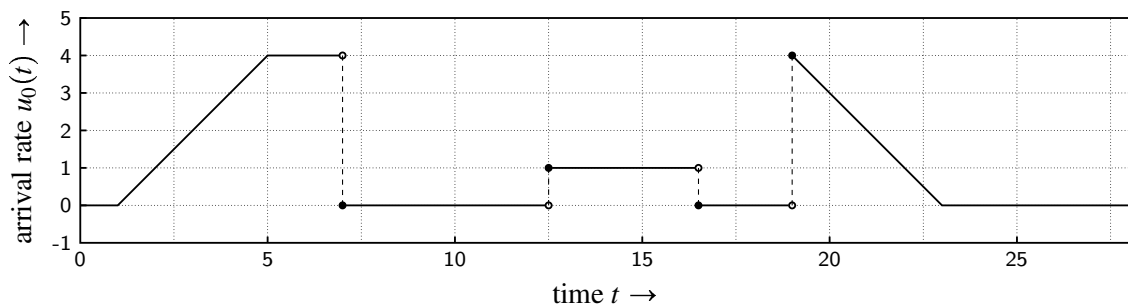


Figure 2.18: Arrival rate $u_0(t)$ of lots at the flow line.

For the models in which individual lots can be distinguished, their arrival times are determined from the $u_0(t)$ graph. The reason for this procedure is that it is easier to determine arrival times based on an arrival rate profile than constructing an arrival rate profile based on arrival times. The arrival times of lots are the time instants at which the cumulative inflow rate has increased

by one. The arrival times of the 28 lots are:

$$\begin{aligned} \tilde{\mathbf{u}}_0 = [& \sqrt{2}+1 \quad 3 \quad \sqrt{6}+1 \quad \sqrt{8}+1 \quad \sqrt{10}+1 \quad \sqrt{12}+1 \quad \sqrt{14}+1 \quad 5 \quad \dots \\ & 5\frac{1}{4} \quad 5\frac{1}{2} \quad 5\frac{3}{4} \quad 6 \quad 6\frac{1}{4} \quad 6\frac{1}{2} \quad 6\frac{3}{4} \quad 7 \quad 13\frac{1}{2} \quad 14\frac{1}{2} \quad 15\frac{1}{2} \quad 16\frac{1}{2} \quad \dots \\ & 23 - \sqrt{14} \quad 23 - \sqrt{12} \quad 23 - \sqrt{10} \quad 23 - \sqrt{8} \quad 23 - \sqrt{6} \quad 21 \quad 23 - \sqrt{2} \quad 23]^T. \end{aligned} \quad (2.52)$$

All model types (max-plus, min-plus, timed discrete event χ , standard fluid, discrete hybrid automaton, hybrid fluid and hybrid χ) are evaluated for this example. For this example, the hybrid fluid model of Section 2.3.2 reduces to the standard fluid model of Section 2.2.1, since no additional discrete event dynamics are present in the flow line. For the max-plus and min-plus algebraic models, characters w are used instead of x , to prevent confusion with the buffer levels x_i , $i \in \{1, 2, 3\}$. Signals w_1 , w_3 and w_5 relate to the entrance of a lot in buffers B_1 , B_2 and B_3 respectively. Signals w_2 and w_4 relate to the start of a lot on machine M_1 and M_2 respectively. In the max-plus model, the signals denote the time instants at which the events occur, while in the min-plus model the signals count the number of specific events until (and including) time t . The max-plus and min-plus model for the flow line are:

max-plus model:	min-plus model:
$w_1(k) = u(k)$	$w_1(t) = u(t)$
$w_2(k) = \max(w_1(k), w_2(k-1) + d_1)$	$w_2(t) = \min(w_1(t), w_2(t-d_1) + 1)$
$w_3(k) = w_2(k) + d_1$	$w_3(t) = w_2(t-d_1)$
$w_4(k) = \max(w_3(k), w_4(k-1) + d_2)$	$w_4(t) = \min(w_3(t), w_4(t-d_2) + 1)$
$w_5(k) = w_4(k) + d_2$	$w_5(t) = w_4(t-d_2).$

For the standard fluid model, machine actual process rates $u_1(t)$ and $u_2(t)$ are introduced. The continuous-time fluid model for the flow line is:

$$\begin{aligned} \dot{x}_1 &= u_0(t) - u_1(t) & u_1(t) &= \begin{cases} \mu_1 & \text{if } x_1(t) > 0 \\ \min(u_0(t), \mu_1) & \text{if } x_1(t) = 0 \end{cases} \\ \dot{x}_2 &= u_1(t) - u_2(t) & u_2(t) &= \begin{cases} \mu_2 & \text{if } x_2(t) > 0 \\ \min(u_1(t), \mu_2) & \text{if } x_2(t) = 0 \end{cases} \\ \dot{x}_3 &= u_2(t). \end{aligned}$$

The discrete hybrid automaton for this flow line is constructed using HYSDEL. The resulting MLD model resembles (2.37) and is not included here because of its size: 29 inequalities specify the logic relations and determine the actual machine process rates.

The χ models are simple and elegant models, very similar to the models specified in Section 2.1.3 and Section 2.3.3. A lot generating process G is added that sends lots to the first workstation at the time instants specified in (2.52).

For all model types, it is expected that during the first simulated hour, the signals in the models do not change. This is due to the fact that the factory starts empty and during the first hour, no

lots arrive at the flow line. For the model types in which individual lots can be distinguished, it is expected that the results of the evaluation are similar. The resulting signals may have to be translated/interpreted to other signals to make the comparison possible (e.g. the translation from time instants in event domain for max-plus models to counters in the min-plus model and a lot-time diagram for the χ models). For the models that use continuous variables for wip levels, it is expected that the wip level signals in the standard fluid model evolve earlier than the wip level in the DHA model, because the latter takes the time delay due to processing lots into account, while the former does not.

The max-plus and min-plus models are evaluated with MATLAB. The standard fluid model is evaluated with SIMULINK and the discrete hybrid automaton is evaluated using HYSDEL and MATLAB. The χ models are evaluated using χ version 1.0. Both χ models give similar results. The results from the χ models are shown in Figure 2.19 as a lot-time diagram. The lightgray boxes indicate that a lot is in a buffer. Buffer 3 has not been plotted here, it is the stock of finished lots. The black boxes represent the processing of a lot on machine M_1 , whereas the darkgray boxes represent residence of a lot on machine M_2 . The max-plus signals w_1 and w_5 have also been plotted in the figure by means of crosses and dots respectively. Notice that iteration counter k is the vertical axis, while the values of $w_1(k)$ and $w_5(k)$ are on the horizontal time axis. Signals w_2 , w_3 and w_4 have not been plotted. They coincide completely with the left hand sides of the boxes in Figure 2.19. The results of the simulations satisfy the expectations: the models stay in rest until the first lot comes in and the model types give similar results.

The results of the standard fluid model and MLD model are shown in Figures 2.20, 2.21 and 2.22. In Figure 2.22 signal $w_5(t)$ of the min-plus model is also plotted. It should be noticed that the min-plus realization fits the lot-time diagram of Figure 2.19 exactly, so this realization can be used to compare the discrete event models with the continuous and hybrid models. The expectations are fulfilled quite well. The first hour, all models remain in rest (no inflow of lots). From $t = 1$, the standard fluid model starts to evolve, although it seems that buffer B_1 and B_2 remain in rest for a longer time. Looking at Figure 2.22 reveals that from $t = 1$, lots start to arrive at buffer B_3 , which indicates that they have passed through two workstations. The maximal process rate of the machines was higher than the arrival rate during the first hours, so buffers B_1 and B_2 could be kept empty. As expected, the standard fluid model finished lots earlier than the hybrid MLD model. In this MLD model, buffer B_2 could be emptied further at $t = 13$, since the process rate of machine M_1 is zero then (its wip level is below 1. However, the graph shows that buffer B_2 is not emptied here. The reason for this is that the process rate of machine M_1 is set to $\varepsilon > 0$, as explained in the Section 2.3.1. The second workstation cannot empty the buffer now, according to the logic relations in (2.36). It should also be noted that the MLD model suffers from small numerical differences with respect to the values that one should expect. If the discrete time step size is chosen small enough, the differences are also kept small. For all model evaluations, the 28 lots all come out of the workstations. This is an expected result.

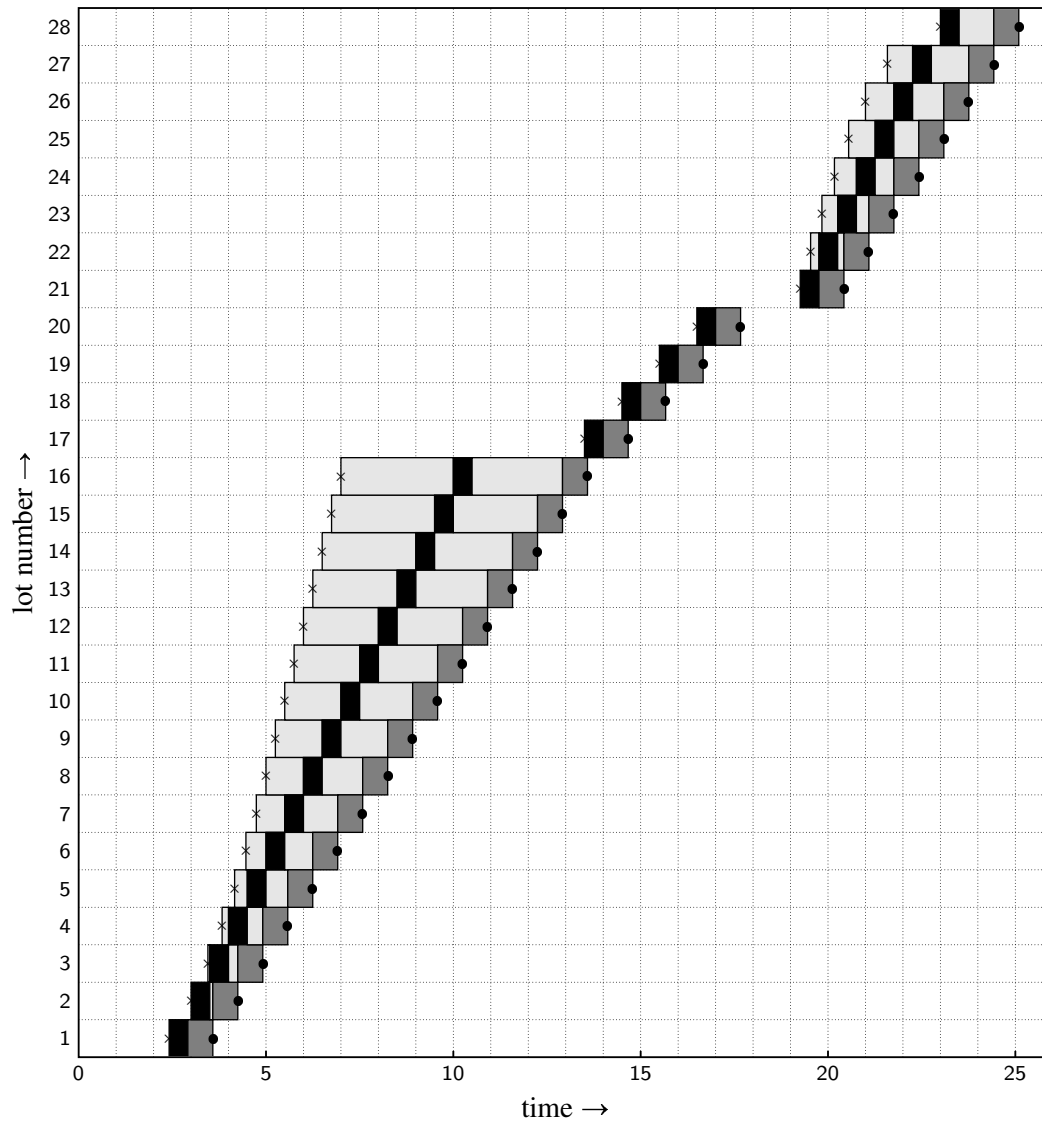


Figure 2.19: Lot-time diagram for χ simulation (both timed χ and hybrid χ) of the flow line. Light gray box: lot in buffer. Black box: lot on machine M_1 . Dark gray box: lot on machine M_2 . Also max-plus results: crosses denote signal w_1 , dots denote signal w_5 .

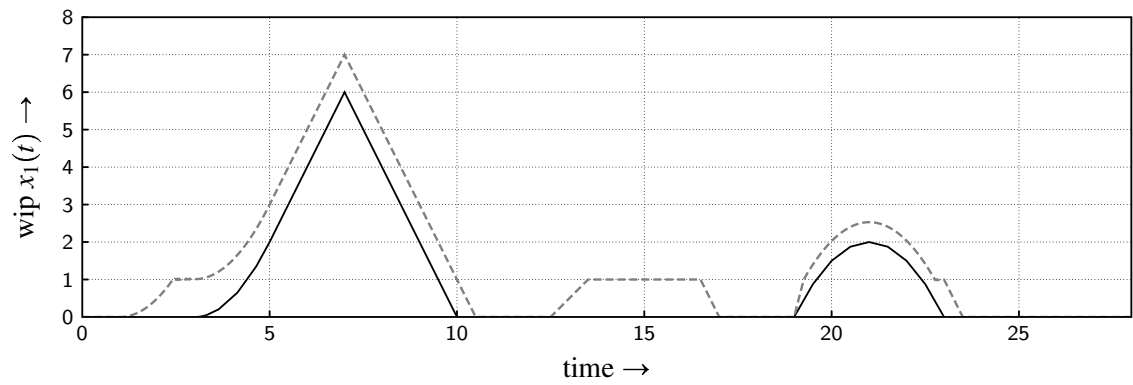


Figure 2.20: Work in process levels of workstation $B_1 - M_1$ in the standard fluid model (solid) and MLD model (dashed).

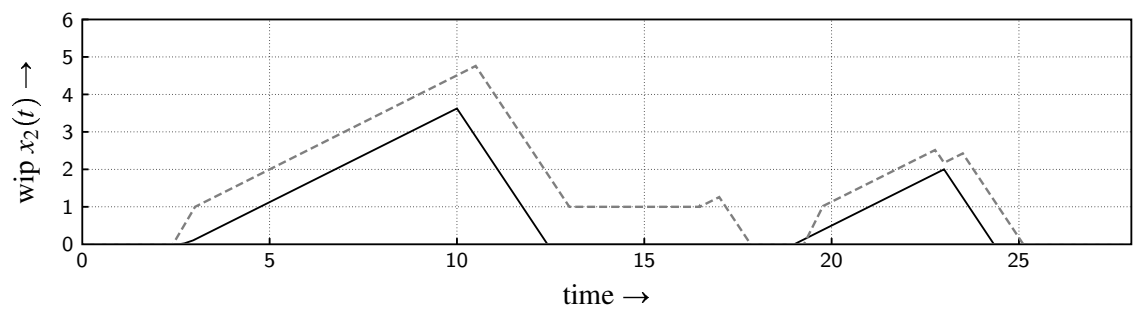


Figure 2.21: Work in process levels of workstation $B_2 - M_2$ in the standard fluid model (solid) and MLD model (dashed).

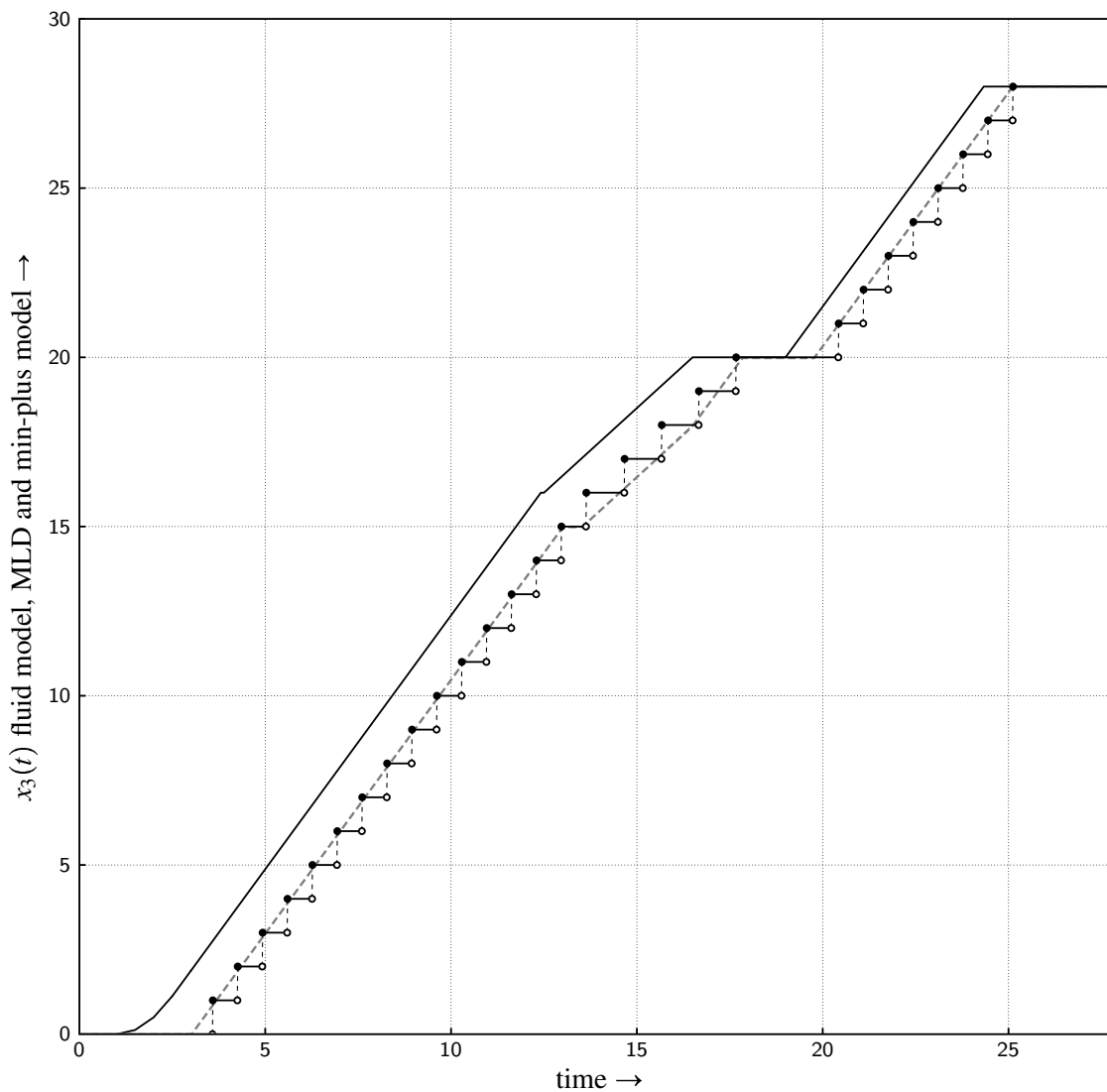


Figure 2.22: Buffer level of B_3 -stock in the standard fluid model (solid) and MLD model (dashed). Additionally, signal $w_5(t)$ has been plotted (stairs-shaped).

2.5 Summary

In this chapter, an overview of different modelling techniques for manufacturing systems has been given, which is by no means intended to be exhaustive. This section summarizes the models that have been treated.

Manufacturing systems can be modelled in many different ways. This chapter provides an overview of a few different modelling techniques and is not an attempt to cover the whole area of modelling paradigms and frameworks. The rationale for including these modelling techniques is that they are either elaborated on in the remainder of this thesis or closely related to the model types that are treated, to indicate similarities and differences.

A distinction between several classes of model types has been made: discrete event models, continuous models and hybrid models. The distinction has been made as follows: discrete event models are models that ‘live’ in the event domain (the dynamics is driven by events) or where all variables are of discrete nature, i.e. they can take on a countable number of values. Continuous models are models in which the variables can take on an uncountable number of values. Continuous models generally ‘live’ in the time domain (the discrete time domain is possible). Hybrid models are models in which the variables are a mixture of discrete and continuous variables. Both continuous dynamics and discrete event dynamics are part of hybrid models.

The different modelling techniques that have been treated in this chapter are:

- Max-plus models, for manufacturing systems often used in the event domain with real-valued variables.
- Min-plus models, for manufacturing systems often used in the time domain. In this chapter, the variables were counters that count the number of a specific event that has taken place until and including a certain time instant.
- Timed process algebra models, using formalism χ . A powerful and formal way to model manufacturing systems. Timed (discrete event) process algebra is a subclass of the hybrid process algebra χ .
- Fluid models, to approximate the discrete nature of buffer levels and counters. Disadvantage of the fluid model is that the flow time of lots cannot be determined, since the time delay due to processing of lots is not incorporated in the models. Important is the insight that instead of buffer levels, one should regard the variables as work in process levels.
- Flow models, often used in traffic flow models. Lately, partial differential equation (PDE) models have been used in manufacturing systems modelling, because great similarities exist between traffic flow and products through a manufacturing network. Difficulty however is to catch the dynamic phenomena of a manufacturing system in a PDE.
- Discrete hybrid automata (DHA), a framework for a class of hybrid systems that are computationally tractable. Mixed logical dynamic (MLD) models and piecewise affine (PWA) models have been explained. These models are able to incorporate the time delay due to the process time of a machine. In this model type it is also better to speak of work

in process levels rather than buffer levels.

- Hybrid fluid models, which can possibly be written as a DHA. Hybrid fluid models combine the standard continuous fluid models with discrete events that represent all kinds of phenomena that may occur in manufacturing systems. Hybrid fluid models are used in the analyses in Chapter 5 and Chapter 6.
- Hybrid process algebra, using formalism χ . A formal method to describe the dynamical behavior of manufacturing systems, in which discrete events are combined with differential equations.

While developing the hybrid process algebra model, a state space representation was introduced. The state includes the number of lots on each machine and in each buffer and the remaining process or transport time of the lots that are being processed or transported. This finite dimensional and instantaneously measurable state does not contain any information about the production or control policy. The introduced state space representation is used in Chapters 3 and 4.

As mentioned before, the process time of machines was not incorporated in the standard continuous fluid model. The problem of this property is that flow times cannot be determined anymore, since eventually the required throughput can be reached without any work in process. In Section 2.2 a way to overcome this problem has been mentioned: Padé approximations. Another way to overcome the time delay problem is to model a manufacturing flow line with a *flow* model. These models are based on *partial* differential equations. Flow models are not elaborated further in the remainder of this thesis, but a short idea of modelling a manufacturing system as a PDE has been given in this chapter. The ideas behind this modelling technique have been adopted from traffic flow modelling and control.

Outlook

The next chapters elaborate on the different modelling techniques. Chapter 3 investigates the coupling between models by means of maps between the state trajectories. Max-plus, min-plus and hybrid χ models are coupled, facilitating the use of the different analysis and control techniques that are available for the modelling paradigms. Once a model of a manufacturing system has been made, it can be used to develop and/or derive feedback controllers. These controllers aim to steer a manufacturing system to its desired behavior and keep it there as good as possible, even if disruptions occur. Due to the feedback mechanism, the controlled system is aware of deviations from the desired behavior and new actions can be computed. In Chapter 4, state feedback controllers are developed, based on multi-parametric programming techniques. The measured state in that chapter is part of the state space representation that has been introduced in this chapter.

The hybrid fluid model technique is used in Chapter 5 and Chapter 6 where it is used in the analysis of switching servers, i.e. workstations that process multiple product types. For these hybrid fluid models, controllers are proposed in these chapters.

Coupling event domain and time domain models of manufacturing systems

Manufacturing systems are often characterized as discrete event systems and therefore modelled as discrete event systems. These models are event driven, i.e. events occur and time labels are assigned to events. Different modelling techniques exist for discrete event systems. A detailed overview of modelling techniques is presented by Cassandras and Lafortune in [24]. Control methods have been developed in the event domain. A modelling and control framework for timed event graphs in dioids is presented in [28, 51]. For max-plus models in a model predictive control framework, a lot of work has been done by De Schutter and Van den Boom, for example in [33]. An advantage of some discrete event modelling paradigms is that models are simple and elegant, and scale up linearly when enlarging the system to be modelled. However, a disadvantage of discrete event models is that a mathematical background for analysis in the time domain is hard to establish. Moreover, especially in the timed event graph and dioid paradigm, dealing with initial conditions of a manufacturing system is difficult, which makes practical relevance doubtful. In event domain, the state usually is not a function of time, which makes real-time control difficult, if not impossible. In Chapter 2 a state space representation as a function of time for manufacturing systems has been introduced. This characterization is well suited for incorporating initial conditions of a manufacturing system. In Chapter 4 also a control method is introduced for a certain class of manufacturing systems, using this system representation.

In real industrial manufacturing systems, although often event driven, events occur while time elapses and many control and performance notions are specified in time domain, as explained earlier in the introductory chapter. One could think of stability, transient behavior, throughput

measurements and flow times of lots. In addition, for time domain systems (not necessarily manufacturing systems) many control methods have been developed. This is the reason why a framework for real-time control of discrete event systems using time domain notions is developed. In [94, 95] a coupling has been established between max-plus and min-plus models of a manufacturing system (which is used in this thesis in Section 3.3.1) and an MPC setup in the time domain has been formulated for a class of manufacturing systems.

In this chapter event domain and time domain models of manufacturing systems are coupled. This facilitates the use of the advantages of both domains and maybe loosen the disadvantages of some modelling techniques. Maps are presented between states and signals of models in different modelling paradigms. The maps are generic in a way that when manufacturing systems grow, the models and maps between states grow proportionally. In Section 3.1 definitions and notations are presented for dynamical systems and state space dynamical systems. In Section 3.2 models are developed for the most basic manufacturing system: one workstation. Three different modelling techniques are used, for both event domain and time domain. The modelling techniques are max-plus, min-plus and hybrid χ , as introduced in the previous chapter. In Section 3.3 maps are presented to couple state and signal vectors from different model representations. In this way, time domain and event domain models can be interconnected for analysis and real-time control, combining all advantages of the two domains.

3.1 (State space) Dynamical systems

3.1.1 Class of manufacturing systems under consideration

In this chapter manufacturing systems are considered in which only synchronization occurs. All product recipes, orders and routes are predetermined and all system parameters are deterministic. Moreover, only timed manufacturing systems are considered: processing a lot takes a non-negligible amount of time. Examples of manufacturing systems under consideration are: buffers, single-lot machines, batch machines and assembly stations. These examples are building blocks from which larger manufacturing systems can be constructed by means of interconnection. In this chapter, the coupling framework is restricted initially to an elementary building block: a workstation, consisting of a first-in-first-out buffer with finite storage capacity and a single-lot machine. However, all concepts presented here are suitable for the manufacturing building blocks mentioned above.

3.1.2 Definitions and notational aspects

The manufacturing systems are considered to be *dynamical systems*, as described by Willems in [111]. Definitions for time domain dynamical systems are:

Definition 3.1. A *dynamical system* Σ is a triple $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ with $\mathbb{T} \subseteq \mathbb{R}$ the *time axis*, \mathbb{W} the *signal space* and $\mathcal{B} \subseteq \mathbb{W}^{\mathbb{T}}$ the *behavior*.

A dynamical system is thus defined by \mathbb{T} (the time instants of interest), \mathbb{W} (the space in which the time signals take on their values) and \mathcal{B} (a family of \mathbb{W} -valued time trajectories).

Definition 3.2. A dynamical system with latent variables is defined as $\Sigma_f = (\mathbb{T}, \mathbb{W}, \mathbb{L}, \mathcal{B}_f)$ with $\mathbb{T} \subseteq \mathbb{R}$ the *time axis*, \mathbb{W} the *signal space* of manifest variables, \mathbb{L} the space of latent variables and $\mathcal{B}_f \subseteq (\mathbb{W} \times \mathbb{L})^{\mathbb{T}}$ the *full behavior*. It defines a *latent variable representation* of the *manifest dynamical system* $\Sigma = (\mathbb{T}, \mathbb{W}, \mathcal{B})$ with *manifest behavior* $\mathcal{B} := \{w : \mathbb{T} \rightarrow \mathbb{W} \mid \exists l : \mathbb{T} \rightarrow \mathbb{L} \text{ such that } (w, l) \in \mathcal{B}_f\}$.

In the framework of Willems, the signal variables are those variables which the model aims at describing, and are called *manifest* variables, whereas the *latent* variables are considered auxiliary variables or internal variables. A special case of a dynamical system with latent variables is a state space dynamical system:

Definition 3.3. A *state space dynamical system* is defined as a dynamical system with latent variables $\Sigma_s = (\mathbb{T}, \mathbb{W}, \mathbb{X}, \mathcal{B}_s)$ in which the full behavior \mathcal{B}_s satisfies the axiom of state. This axiom requires that:

$$\{(\mathbf{w}_1, \mathbf{x}_1), (\mathbf{w}_2, \mathbf{x}_2) \in \mathcal{B}_s, t \in \mathbb{T} \text{ and } \mathbf{x}_1(t) = \mathbf{x}_2(t)\} \Rightarrow \{(\mathbf{w}, \mathbf{x}) \in \mathcal{B}_s\}$$

with (\mathbf{w}, \mathbf{x}) defined as

$$(\mathbf{w}(t'), \mathbf{x}(t')) = \begin{cases} (\mathbf{w}_1(t'), \mathbf{x}_1(t')) & \text{for } t' < t \\ (\mathbf{w}_2(t'), \mathbf{x}_2(t')) & \text{for } t' \geq t. \end{cases}$$

Informally, the axiom of state means that the *state should contain sufficient information about the past so as to determine future behavior* [90, 111]. The state makes the future independent from the past. The axiom of state is illustrated in Figure 3.1, in which the three dimensions represent time t , state x and (manifest) signal w . The lightgray plane is the intersection plane at $t' = t$. The equality $x_1(t) = x_2(t)$ has been indicated in the figure.

In Chapter 2 several modelling techniques for manufacturing systems have been presented. In this chapter, the max-plus algebraic model, min-plus algebraic model and hybrid χ model of manufacturing workstations are elaborated on.

In the max-plus model of a manufacturing system, the *time domain* has been replaced with the *event domain*, which represents a counter for the events. The signals in the max-plus algebraic models of a workstation represent time instants at which the events occur. Thus for a max-plus algebraic dynamical system: $\mathbb{T} = \mathbb{Z}$ and $\mathbb{W} = \mathbb{R}_\varepsilon^p$, with p a proper dimension. Because events occur chronologically in time (e.g. the second event does not occur earlier than the first event), the signals in a max-plus algebraic model are non-decreasing. Let $\mathbf{w}_\mathcal{K}$ denote the manifest variables in the max-plus model (with \mathcal{K} indicating that the signals ‘live’ in the event domain), behavior $\mathcal{B}_\mathcal{K}$ can then be defined as:

$$\mathcal{B}_\mathcal{K} = \left\{ \mathbf{w}_\mathcal{K} \left| \begin{array}{l} \gamma^k \mathbf{w}_\mathcal{K} \leq \mathbf{w}_\mathcal{K}, \forall k > 0 \\ \text{“Physical laws of system are satisfied”} \end{array} \right. \right\} \quad (3.1)$$

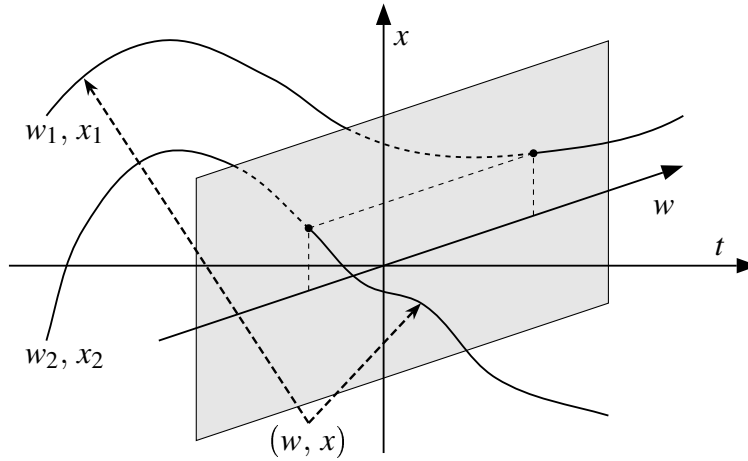


Figure 3.1: Graphical representation of axiom of state (taken from [111]).

where γ is the event shift operator: $\gamma^n w_{\mathcal{X}}(k) = w_{\mathcal{X}}(k - n)$. The phrase “*Physical laws of system are satisfied*” contains constraints on product recipes, routes, capacities and production policies. In Section 3.2, this phrase is made explicit.

In the min-plus model of a manufacturing system, the *time domain* is a real time axis, representing time instants. The signals in the min-plus algebraic models of a workstation represent counters. The counters denote the number of times a specific event has occurred at a certain time instant. Thus for a min-plus algebraic dynamical system: $\mathbb{T} = \mathbb{R}$ and $\mathbb{W} = \mathbb{Z}^q$, with q a proper dimension. Since the signals in the min-plus models *count* how often an event has occurred, it is clear that these signals are also non-decreasing signals. Let $\mathbf{w}_{\mathcal{T}}$ denote the manifest variables in the min-plus model (with \mathcal{T} indicating that the signals ‘live’ in time domain), behavior $\mathcal{B}_{\mathcal{T}}$ can then be defined as:

$$\mathcal{B}_{\mathcal{T}} = \left\{ \mathbf{w}_{\mathcal{T}} \mid \begin{array}{l} \sigma^{\tau} \mathbf{w}_{\mathcal{T}} \leq \mathbf{w}_{\mathcal{T}}, \forall \tau > 0 \\ \text{“Physical laws of system are satisfied”} \end{array} \right\} \quad (3.2)$$

where σ is the time shift operator: $\sigma^{\tau} w_{\mathcal{T}}(t) = w_{\mathcal{T}}(t - \tau)$ and again the phrase “*Physical laws of system are satisfied*” contains constraints on the same issues as described above. The non-decreasing signals property of both the max-plus and min-plus signals is important in coupling the models, as is explained in Section 3.3.1.

The hybrid χ model ‘lives’ in the time domain and the signals represent the numbers of lots in buffers and machines and the remaining process times of lots on machines (see Section 2.3.3). For the hybrid χ model, $\mathbb{T} = \mathbb{R}$ and $\mathbb{W} = \mathbb{N}^r \times \mathbb{R}^s$, with r and s of proper dimensions. Let \mathbf{w}_{χ} denote the manifest variables in hybrid χ , then behavior $\mathcal{B}_{\mathcal{T}}^{\chi}$ can be defined as:

$$\mathcal{B}_{\mathcal{T}}^{\chi} = \{ \mathbf{w}_{\chi} \mid \text{“Physical laws of system are satisfied”} \}. \quad (3.3)$$

Although the same sentence “*Physical laws of system are satisfied*” appears three times in the defined behaviors, they represent different constraints in each case. However, they embody the same physical phenomena and constraints that occur in a manufacturing system.

Now that the manufacturing systems under consideration and the type of signals that are allowed have been defined, different modelling techniques can be used to describe the evolution of the signals in both time and event domain, i.e. in the three model paradigms: max-plus, min-plus and hybrid χ .

3.2 Modelling a workstation

In this section, the basic building block of a manufacturing system, a workstation, is presented. First the dynamics are specified in an informal way and then the three modelling techniques are used to explicitly model the dynamics of this workstation. The deliberately left vague sentence “Physical laws of system are satisfied” of Section 3.1 is made explicit in this section.

3.2.1 Informal description

Consider the workstation consisting of buffer B and single-lot machine M as shown in Figure 3.2. Buffer B has a capacity of N lots. Machine M has a constant process time of d time units and can process only one lot at a time. Lots are pushed through the workstation, i.e. the machine never stays idle when lots reside in the buffer. In the next sections, this informal description of the dynamics is elaborated in different time domain and event domain models.

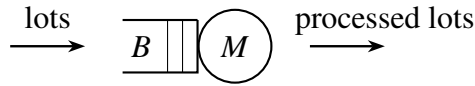


Figure 3.2: Workstation with buffer B (finite capacity) and single-lot machine M .

3.2.2 Event domain: max-plus model

Max-plus algebra is suitable to obtain a representation of the workstation in event domain. In Chapter 2 an introduction into this algebra has been given. For reasons of readability, the conventional $+$ and max operators are used in modelling the workstation. Let $w_{\mathcal{X}_u}$ be the signal denoting the time instants at which lots arrive at the workstation. Signal $w_{\mathcal{X}_1}$ denotes the time instants lots enter the buffer, while $w_{\mathcal{X}_2}$ is the signal containing the time instants lots leave the workstation after being processed. Furthermore, $w_{\mathcal{X}_u}(k)$ denotes the time instant a lot arrives at the workstation for the k^{th} time, and similarly for $w_{\mathcal{X}_1}(k)$ and $w_{\mathcal{X}_2}(k)$.

A lot enters the buffer as soon as it has arrived and an empty space is available in the buffer. A lot leaves the workstation as soon as it has been processed. Only one lot can be processed at a time. The max-plus representation is then given by:

$$w_{\mathcal{X}_1}(k) = \max [w_{\mathcal{X}_u}(k), w_{\mathcal{X}_2}(k - (N + 1))] \quad (3.4a)$$

$$w_{\mathcal{X}_2}(k) = \max [w_{\mathcal{X}_1}(k) + d, w_{\mathcal{X}_2}(k - 1) + d]. \quad (3.4b)$$

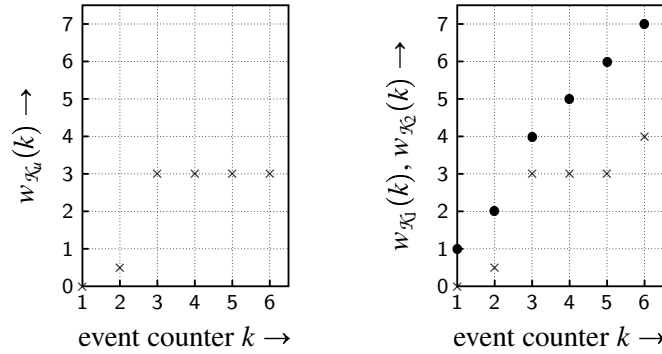


Figure 3.3: Example of signals $w_{\mathcal{K}_u}(k)$ (left), $w_{\mathcal{K}_1}(k)$ (crosses) and $w_{\mathcal{K}_2}(k)$ (dots) for one workstation with $N = 2$ and $d = 1$.

The event driven max-plus model of the workstation then becomes (the \oplus superscript indicates the modelling technique):

$$\mathcal{B}_{\mathcal{K}}^{\oplus} = \left\{ \mathbf{w}_{\mathcal{K}} : \mathbb{Z} \rightarrow \mathbb{R}^3 \mid \gamma^k \mathbf{w}_{\mathcal{K}} \leq \mathbf{w}_{\mathcal{K}}, \forall k > 0; (3.4) \right\} \quad (3.5)$$

with $\mathbf{w}_{\mathcal{K}} = [w_{\mathcal{K}_u} \ w_{\mathcal{K}_1} \ w_{\mathcal{K}_2}]^T$. Note that modelling in max-plus algebra is generic in a sense that adding a second workstation to the line would be a matter of adding one max-algebraic equation to (3.4). In Figure 3.3 an example is given of the signals in this behavior with $N = 2$ and $d = 1$. The horizontal axis is the event counter, while the vertical axis is a time axis. The graphs thus show the *time instants* at which *events* occur. The figure shows that 4 lots arrive at the workstation at the same time (time = 3), but they cannot enter the buffer all at once, because of the limited capacity: $w_{\mathcal{K}_1}(6) = 4$ while $w_{\mathcal{K}_u}(6) = 3$. The sixth lot must remain at its source until time = 4. This source can be a warehouse or a preceding workstation, which is blocked because it cannot send the lot away when possible. Signal $w_{\mathcal{K}_u}$ can be regarded as input signal.

Remark 3.4. Vector $\mathbf{w}_{\mathcal{K}}(k)$ in this model is *not* the state of the system. The state decouples the past from the future. For the workstation, (3.4a) contains the term $w_{\mathcal{K}_2}(k - (N + 1))$. In further iterations, $w_{\mathcal{K}_1}(k + 1)$ contains the term $w_{\mathcal{K}_2}(k - N)$ and so on. To make the future independent from the past, at least all terms $w_{\mathcal{K}_2}(k - 1) \dots w_{\mathcal{K}_2}(k - (N + 1))$ need to be included in state vector $\mathbf{x}_{\mathcal{K}}(k)$. The value of $N + 1$ is called the *event memory span* of the system.

3.2.3 Time domain: min-plus model

The workstation described in Section 3.2 can also be modelled in the time domain, for example by means of min-plus algebra. A short introduction into the min-plus algebra has been given in Chapter 2. Again, only the conventional $+$ and min operators are used in the expressions here. Let $w_{\mathcal{T}_u}$ be the signal denoting the number of lots that arrived at the workstation. Let $w_{\mathcal{T}_1}$ denote the signal containing the number of lots that entered the buffer, while $w_{\mathcal{T}_2}$ is the signal containing the number of lots that have left the workstation after being processed. Furthermore, $w_{\mathcal{T}_u}(t)$

denotes the number of lots that have arrived at the workstation at time t , and similarly for $w_{T_1}(t)$ and $w_{T_2}(t)$. Only right-continuous signals are considered.

The dynamics in time domain perspective can now be described in words as: The number of lots that have entered the buffer at time t equals the minimum of the number of available lots and the number of lots that have left the workstation minus the complete capacity of the workstation. In addition, the number of lots that have left the workstation equals the minimum of lots that has entered the buffer and of the number of lots that had left d time units ago. In other words:

$$w_{T_1}(t) = \min[w_{T_u}(t), w_{T_2}(t) + N + 1] \quad (3.6a)$$

$$w_{T_2}(t) = \min[w_{T_1}(t - d), w_{T_2}(t - d) + 1]. \quad (3.6b)$$

The time driven min-plus model of the workstation then becomes (the \ominus superscript indicates the modelling technique):

$$\mathcal{B}_T^\ominus = \{\mathbf{w}_T: \mathbb{R} \rightarrow \mathbb{Z}^3 \mid \sigma^\tau \mathbf{w}_T \leq \mathbf{w}_T, \forall \tau > 0; (3.6)\} \quad (3.7)$$

with $\mathbf{w}_T = [w_{T_u} \ w_{T_1} \ w_{T_2}]^\top$. Similar to the max-plus model (3.5), min-plus model (3.7) is also scalable: adding more workstations makes the model grow proportionally.

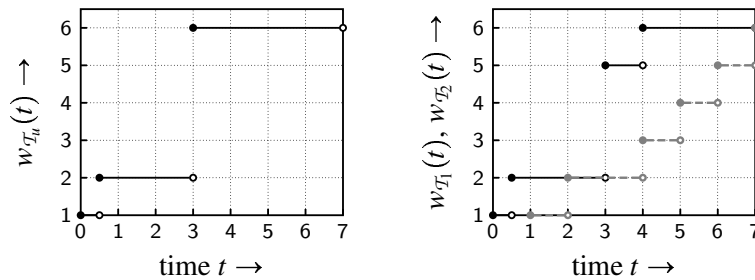


Figure 3.4: Example of signals $w_{T_u}(t)$ (left), $w_{T_1}(t)$ (solid) and $w_{T_2}(t)$ (dashed) for one workstation with $N = 2$ and $d = 1$.

The signals in \mathbf{w}_T for the same example as Figure 3.3 are shown in Figure 3.4. The jump in $w_{T_u}(t)$ at $t = 3$ means that at this time instant, four new lots are offered at the buffer. The right hand side graph in Figure 3.4 shows that not all four lots can enter the buffer immediately upon arrival: the fourth lot can only enter at $t = 4$ since the buffer's maximum capacity cannot be exceeded.

Remark 3.5. Note that vector $\mathbf{w}_T(t)$ in this model is *not* the state of the system. In (3.6b), the expression for $w_{T_2}(t)$ contains the terms $w_{T_1}(t - d)$ and $w_{T_2}(t - d)$. In the evolution of the signals after a small time step $\varepsilon \ll d$, $w_{T_2}(t + \varepsilon)$ contains $w_{T_1}(t - d + \varepsilon)$ and $w_{T_2}(t - d + \varepsilon)$. So information of these signals over at least the interval $[-d, 0]$ is necessary in the state to make the future independent from the past. The interval over which the signals are stored in state $\mathbf{x}_T(t)$ is called the *memory span* $\Delta \geq d$ of the system. More details on the memory span are provided in [111].

If signal $w_{\mathcal{T}_i}$ is regarded as the input of a state space model, the relation between state variables $x_{\mathcal{T}_i}(t)$ and signals $w_{\mathcal{T}_i}(t)$ can be defined as:

$$x_{\mathcal{T}_i}(t) : [-\Delta, 0] \rightarrow \mathbb{Z} \text{ with } x_{\mathcal{T}_i}(t)(\tau) = w_{\mathcal{T}_i}(t + \tau) \quad (3.8a)$$

$$w_{\mathcal{T}_i}(t) : \mathbb{R} \rightarrow \mathbb{Z} \text{ with } w_{\mathcal{T}_i}(t) = x_{\mathcal{T}_i}(t)(0) \quad (3.8b)$$

with $i \in \{1, 2\}$ and the full behavior of the workstation is:

$$\mathcal{B}_{Ts}^\Theta = \left\{ \begin{array}{l} \mathbf{w}_T : \mathbb{R} \rightarrow \mathbb{Z}^3 \\ \mathbf{x}_T : \mathbb{R} \rightarrow ([-\Delta, 0] \rightarrow \mathbb{Z})^2 \end{array} \middle| \begin{array}{l} \sigma^\tau \mathbf{w}_T \leq \mathbf{w}_T, \forall \tau > 0 \\ (3.6), (3.8) \\ \Delta \geq d \end{array} \right\} \quad (3.9)$$

with $\mathbf{x}_T(t) = [x_{\mathcal{T}_1}(t) \ x_{\mathcal{T}_2}(t)]^\top$. Note that this state is infinitely dimensional, since it consists of piece-wise constant signals over time interval $[-\Delta, 0]$ (as defined in (3.8)).

3.2.4 A hybrid model in the χ formalism

Another way of modelling the dynamics of the workstation is by means of the χ formalism. This formalism is suited for modelling, simulation and analysis of hybrid systems, i.e. with discrete event dynamics and continuous dynamics, in a formal and mathematically unambiguous way. A detailed introduction into the χ formalism has been presented in Chapter 2, formal semantics is described in [9]. The χ models in this chapter are given explicitly, but are discussed in an informal way.

The idea for modelling the workstation in χ is that the concept of using a *memory span*, which is needed to determine a state in min-plus modelling, can be omitted by introducing a continuous state variable in time, representing the remaining process time of the machine, $\bar{x}_{\chi_3}(t) \in [0, d]$, as already presented in Chapter 2. As discrete state variables, the number of lots in the buffer, $\bar{x}_{\chi_1}(t) \in \{0, 1, \dots, N\}$, the number of lots on the machine, $\bar{x}_{\chi_2}(t) \in \{0, 1\}$ and the number of finished lots, $\bar{x}_{\chi_4}(t) \in \mathbb{Z}$ are used. The state of the workstation is defined as:

$$\bar{\mathbf{x}}_\chi(t) = \left[\begin{array}{l} \bar{x}_{\chi_1}(t) \in \{0, 1, \dots, N\} \\ \bar{x}_{\chi_2}(t) \in \{0, 1\} \\ \bar{x}_{\chi_3}(t) \in [0, d] \\ \bar{x}_{\chi_4}(t) \in \mathbb{Z} \end{array} \right] = \begin{array}{l} \text{number of lots in buffer } B; \\ \text{number of lots on machine } M; \\ \text{remaining process time of lot residing on } M; \\ \text{number of finished lots.} \end{array} \quad (3.10)$$

Note that for the number of finished lots the integer type \mathbb{Z} is used, since the event counters in max-plus models have been defined in \mathbb{Z} .

A χ model of the workstation is presented in (3.11). A schematic representation of this χ model is given in Figure 3.5. The χ model consists of three processes: generator G , workstation W and exit process E . Model S connects the processes to each other and communicates the externally provided initial conditions (with subscript 0) to the processes. Process G sends lots to the workstation and has an input signal $w_{\mathcal{T}_u}(t)$, which is similar to the input in the min-plus

model. The number of lots that cannot be sent immediately to the workstation (due to a fully loaded buffer) is stored in variable n . As mentioned before, this is the amount of lots waiting in a warehouse or on previous workstations. Process E receives lots from workstation W after they have been finished and the process updates state element $\bar{x}_{\chi_4}(t)$ and variable $\bar{w}_{\mathcal{T}_2}(t)$. Workstation W combines buffer B and machine M from the informal workstation description in Section 3.2.1. Workstation W works as follows: as long as the maximum buffer capacity is not exceeded, accept lots and store them in counter \bar{x}_{χ_1} . If no lots reside on the machine ($\bar{x}_{\chi_2} = 0$) and the buffer contains lots, start a lot on the machine. The remaining process time decreases linearly according to the differential equations. In processes W and E , variables $\bar{w}_{\mathcal{T}_1}$ and $\bar{w}_{\mathcal{T}_2}$ are the signals in time domain, as if they were the min-plus signals, together with $w_{\mathcal{T}_u}$.

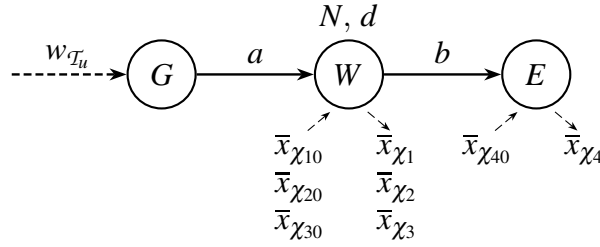


Figure 3.5: Schematic representation of χ model in (3.11).

```

proc G(chan a! : void, alg w_{\mathcal{T}_u} : int) =
  [ [ var n : nat = 0, pu : int = w_{\mathcal{T}_u}
    :: * ( w_{\mathcal{T}_u} \neq pu \rightarrow n := n + w_{\mathcal{T}_u} - pu; pu := w_{\mathcal{T}_u}
          [ n > 0 \rightarrow a!; n := n - 1
          )
    ] ]
]

proc W(chan a?, b! : void, val d : real, N : nat, \bar{w}_{\mathcal{T}_1} : int, \bar{x}_{\chi_1}, \bar{x}_{\chi_2} : nat, \tilde{x}_3 : real) =
  [ [ cont \bar{x}_{\chi_3} : real = \tilde{x}_3
    :: * ( \bar{x}_{\chi_1} < N \rightarrow a?; \bar{w}_{\mathcal{T}_1} := \bar{w}_{\mathcal{T}_1} + 1; \bar{x}_{\chi_1} := \bar{x}_{\chi_1} + 1
          [ \bar{x}_{\chi_1} > 0 \wedge \bar{x}_{\chi_2} = 0 \rightarrow \bar{x}_{\chi_1} := \bar{x}_{\chi_1} - 1; \bar{x}_{\chi_2} := 1; \bar{x}_{\chi_3} := d
          [ \bar{x}_{\chi_2} > 0 \wedge \bar{x}_{\chi_3} = 0 \rightarrow b!; \bar{x}_{\chi_2} := 0
          )
          ] ]
    [ \bar{x}_{\chi_3} > 0 \Rightarrow \dot{\bar{x}}_{\chi_3} = -1
    [ \bar{x}_{\chi_3} = 0 \Rightarrow \dot{\bar{x}}_{\chi_3} = 0
    ] ]
  ]

proc E(chan b? : void, val \bar{w}_{\mathcal{T}_2}, \bar{x}_{\chi_4} : int) =
  [ [ * ( b?; \bar{x}_{\chi_4} := \bar{x}_{\chi_4} + 1; \bar{w}_{\mathcal{T}_2} := \bar{x}_{\chi_4} ) ] ]

```

```

model S(alg  $w_{\mathcal{T}_u}$  : int, val  $d$  : real,  $N$  : nat) =
|| chan  $a, b$  : void
::  $G(a, w_{\mathcal{T}_u})$ 
||  $W(a, b, d, N, \overline{w}_{\mathcal{T}_{10}}, \overline{x}_{\chi_{10}}, \overline{x}_{\chi_{20}}, \overline{x}_{\chi_{30}})$ 
||  $E(b, \overline{w}_{\mathcal{T}_{20}}, \overline{x}_{\chi_{40}})$ 
||

```

Since elementary building blocks proc are specified separately from the model, hybrid χ models are highly scalable. Adding an extra workstation (with different parameters) is only a matter of adding one process term (which is one line) to model S . In this way, very large systems can be modelled using a relatively small specification.

The state space dynamical model $\overline{\mathcal{B}}_{\mathcal{T}_s}^\chi$ of the workstation is now given by:

$$\overline{\mathcal{B}}_{\mathcal{T}_s}^\chi = \left\{ \begin{array}{l} \overline{\mathbf{w}}_{\mathcal{T}} : \mathbb{R} \rightarrow \mathbb{Z}^3 \\ \overline{\mathbf{x}}_\chi : \mathbb{R} \rightarrow \mathbb{N}^2 \times \mathbb{R} \times \mathbb{Z} \end{array} \middle| \chi \text{ model (3.11)} \right\} \quad (3.12)$$

with $\overline{\mathbf{w}}_{\mathcal{T}}(t) = [w_{\mathcal{T}_u}(t) \quad \overline{w}_{\mathcal{T}_1}(t) \quad \overline{w}_{\mathcal{T}_2}(t)]^\top$ and $\overline{\mathbf{x}}_\chi(t)$ as in (3.10).

A major difference with the max-plus or min-plus models is that due to the discrete event nature, signals are piecewise linear over *closed intervals*, i.e. at time instants where events take place, they can have multiple values. In order to obtain single-valued functions of time, the signals from the χ model are mapped onto right continuous signals. The physical meaning of right-continuous signals in this case is that the state at a certain time instant is measured only if all events that can happen at that time instant have taken place. The resulting state space dynamical model is then:

$$\mathcal{B}_{\mathcal{T}_s}^\chi = \left\{ \begin{array}{l} \mathbf{w}_{\mathcal{T}} : \mathbb{R} \rightarrow \mathbb{Z}^3 \\ \mathbf{x}_\chi : \mathbb{R} \rightarrow \mathbb{N}^2 \times \mathbb{R} \times \mathbb{Z} \end{array} \middle| \exists \left(\begin{array}{l} \overline{\mathbf{w}}_{\mathcal{T}}(t) \\ \overline{\mathbf{x}}_\chi(t) \end{array} \right) \in \overline{\mathcal{B}}_{\mathcal{T}_s}^\chi \text{ s.t. } \forall t \in \mathbb{R} : \left(\begin{array}{l} \mathbf{w}_{\mathcal{T}}(t) \\ \mathbf{x}_\chi(t) \end{array} \right) = \lim_{\tilde{t} \downarrow t} \left(\begin{array}{l} \overline{\mathbf{w}}_{\mathcal{T}}(\tilde{t}) \\ \overline{\mathbf{x}}_\chi(\tilde{t}) \end{array} \right) \right\} \quad (3.13)$$

with $\mathbf{w}_{\mathcal{T}}(t) = [w_{\mathcal{T}_u}(t) \quad w_{\mathcal{T}_1}(t) \quad w_{\mathcal{T}_2}(t)]^\top$ and $\mathbf{x}_\chi(t)$ the right-continuous variant of $\overline{\mathbf{x}}_\chi(t)$.

The state evolution from the right-continuous χ model for the same situation as in Figures 3.3 and 3.4 is shown in Figure 3.6. Properties (not exhaustive) of the right-continuous state in (3.13) are:

$$x_{\chi_1}(t) \in \{0, 1, \dots, N\} \quad (3.14a)$$

$$x_{\chi_2}(t) \in \{0, 1\} \quad (3.14b)$$

$$x_{\chi_3}(t) \in [0, d] \quad (3.14c)$$

$$x_{\chi_2}(t) = 0 \Rightarrow x_{\chi_1}(t) = 0; x_{\chi_3}(t) = 0 \quad (3.14d)$$

$$x_{\chi_1}(t) \neq 0 \Rightarrow x_{\chi_2}(t) \neq 0 \quad (3.14e)$$

$$x_{\chi_2}(t) = 1 \Rightarrow x_{\chi_3}(t) > 0 \quad (3.14f)$$

or in words:

- The buffer can only contain a non-negative integer number of lots with the maximum buffer capacity as upper bound.
- The machine processes one lot at a time, so either one or zero lots reside on the machine.
- If the machine is idle, then the buffer must be empty and the remaining process time equals zero (3.14d).
- If the buffer is not empty, a lot must reside on the machine (3.14e).
- If the machine contains a lot, its remaining processing time is strictly greater than zero, because when it becomes zero, it is immediately sent away (3.14e), increasing $x_{\chi_4}(t)$ by one.

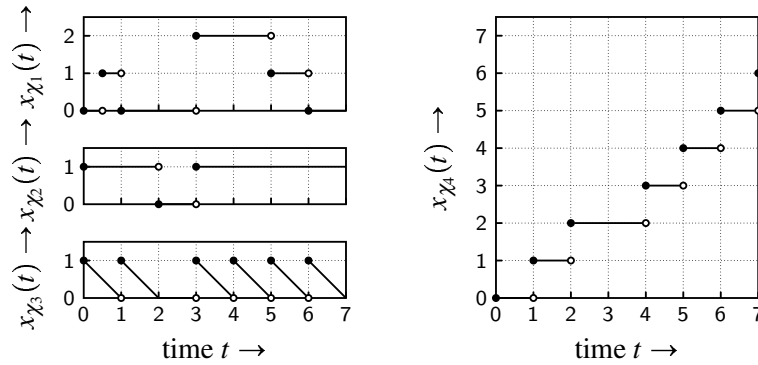


Figure 3.6: Example of state elements $x_{\chi_1}(t)$, $x_{\chi_2}(t)$, $x_{\chi_3}(t)$ and $x_{\chi_4}(t)$ for one workstation with $N = 2$ and $d = 1$, modelled in χ .

These properties do also appear in Figure 3.6. Note that although the state elements itself do not contain any information about production policy, these properties do, since they are only valid within full behavior (3.13), in which a production policy (push) has been enclosed.

Now that the workstation has been specified in a pure event domain model (max-plus), pure time domain model (min-plus) and a hybrid form, in which events take place and the state can be measured continuously over time, the coupling between these system representations is investigated by means of mapping the state trajectories.

3.3 Maps between states and signals

A coupling between the presented models facilitates the use of analysis and control methods from both domains. In order to establish a coupling between the system's representations, a map between the state or signal trajectories of the different models is looked for. The state of the system contains sufficient information about the past to determine future behavior and is therefore suitable as an 'intermediary' between different model types. The three models all represent the same physical manufacturing system, so having information about the past in one model form should be sufficient to construct the state in a different model form. The maps that are treated in this section are schematically shown in Figure 3.7. Between the max-plus and

min-plus signals, a bijection π is established. A state in the min-plus model has been defined in (3.8). Between the min-plus and right continuous hybrid χ state trajectories, maps $\mathcal{M}_{\Theta \rightarrow \chi}$ and $\mathcal{M}_{\chi \rightarrow \Theta}$ are developed. With these maps, one can switch from one system representation to another, using current state or signal information and the maps.

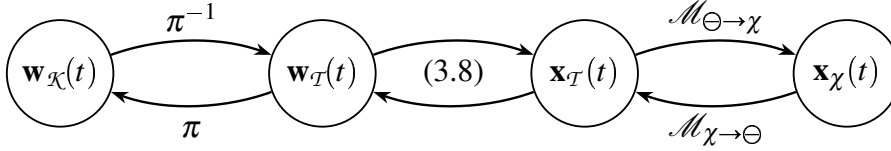


Figure 3.7: Overview of maps between the state trajectories of different system representations.

3.3.1 Coupling the max-plus and min-plus model

Max-plus models (in event domain) and min-plus models (in time domain) of a workstation are closely related if the signals in the models correspond to the same physical events that occur in the workstation. For these max-plus and min-plus models, a bijection π has been developed in [95], which is a map (both surjective and injective). The following properties must hold for the bijection:

- Surjective (map ‘onto’): $\forall w_K \in \mathcal{B}_K, \exists w_T \in \mathcal{B}_T$ such that $\pi(w_T(t)) = w_K(k)$
- Injective (map ‘one-to-one’): $\forall w_{T_1}, w_{T_2} \in \mathcal{B}_T, \pi(w_{T_1}(t)) = \pi(w_{T_2}(t)) \Leftrightarrow w_{T_1}(t) = w_{T_2}(t)$

Proposition 3.6. *If the signals $w_K \in \mathcal{B}_K$ and $w_T \in \mathcal{B}_T$ are non-decreasing, the following maps π and π^{-1} map signals from the max-plus model and the min-plus model onto each other. The resulting signals correspond to the same physical phenomena in the manufacturing system.*

$$w_{K_i}(k) = \pi(w_{T_i}(t)) = \inf_{w_{T_i}(t) \geq k, t \in \mathbb{R}} t, i \in \{1, \dots, n\}, \forall k \in \mathbb{Z} \quad (3.15a)$$

$$w_{T_i}(t) = \pi^{-1}(w_{K_i}(k)) = \sup_{w_{K_i}(k) \leq t, k \in \mathbb{Z}} k, i \in \{1, \dots, n\}, \forall t \in \mathbb{R}. \quad (3.15b)$$

Proof. See [94, 95] for the proof, including conditions and assumptions. □

Loosely speaking, these maps swap the axes of the counter-time graph in min-plus modelling to obtain the time-event graph in max-plus modelling (cf. Figures 3.4 and 3.3). The relatively simple maps π and its inverse map π^{-1} require the signals to be non-decreasing to make this axes-swap possible: otherwise the signals would become multi-valued, which does not fit in the max-plus and min-plus models.

3.3.2 Coupling the min-plus model and hybrid χ model

The state in the min-plus model (as defined in (3.8)) is (in general) infinitely dimensional (signals over a time interval $[-\Delta, 0]$ with $\Delta \geq d$), while the state of the hybrid χ model consists of 4 scalars. To establish the coupling between the two representations, a map between the state trajectories is looked for. So the map from min-plus state to hybrid χ state constructs scalars from signals over a time interval, while the map from hybrid χ state to min-plus state constructs a non-empty set of signals over time interval $[-\Delta, 0]$ from scalars. The maps are presented below, each with an informal explanation afterwards.

For any state $\mathbf{x}_T(t)$ in the min-plus model, it is possible to find a corresponding hybrid χ state $\mathbf{x}_\chi(t) \in \mathcal{B}_{T_s}^\chi$ using the map $\mathcal{M}_{\Theta \rightarrow \chi} : ([-\Delta, 0] \rightarrow \mathbb{Z})^2 \rightarrow \mathbb{N}^2 \times \mathbb{R} \times \mathbb{Z}$:

$$\mathcal{M}_{\Theta \rightarrow \chi} : \begin{cases} x_{\chi_1}(t) = \max(0, x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1) \\ x_{\chi_2}(t) = \min(x_{T_1}(t)(0) - x_{T_2}(t)(0), 1) \\ x_{\chi_3}(t) = \begin{cases} 0 & \text{if } x_{T_1}(t)(0) = x_{T_2}(t)(0) \\ \dots\dots\dots \\ \max(\inf\{\tau \in [-\Delta, 0] | x_{T_2}(t)(\tau) = x_{T_2}(t)(0)\} + d, 0) & \text{if } x_{T_1}(t)(-\Delta) \geq x_{T_2}(t)(0) + 1 \\ \dots\dots\dots \\ \max \left(\begin{array}{l} \inf\{\tau \in [-\Delta, 0] | x_{T_2}(t)(\tau) = x_{T_2}(t)(0)\} + d \\ \inf\{\tau \in [-\Delta, 0] | x_{T_1}(t)(\tau) = x_{T_2}(t)(0) + 1\} + d \\ 0 \end{array} \right) & \text{if } x_{T_1}(t)(-\Delta) < x_{T_2}(t)(0) + 1 \\ & \text{and } x_{T_1}(t)(0) > x_{T_2}(t)(0) \end{cases} \\ x_{\chi_4}(t) = x_{T_2}(t)(0). \end{cases} \quad (3.16)$$

The expressions for $x_{\chi_1}(t) \dots x_{\chi_4}(t)$ can be explained informally:

- $x_{\chi_1}(t)$: This max-expression has been built up from its two parts:

$$x_{\chi_1}(t) = \begin{cases} x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1 & \text{if } x_{T_1}(t)(0) - x_{T_2}(t)(0) > 1 \\ 0 & \text{if } x_{T_1}(t)(0) - x_{T_2}(t)(0) \leq 1. \end{cases} \quad (3.17)$$

The first alternative means that if the number of arrivals at the workstation at time t minus the number of lots that have left the workstation is more than one, then the number of lots in the buffer equals this difference minus one, since one lot must reside on the machine (cf. (3.14e)). The second alternative means that if the aforementioned difference is at most one, then no lots are in the buffer, since the only lot that can be in the workstation is on the machine then.

- $x_{\chi_2}(t)$: This min-expression has also been built up from two parts:

$$x_{\chi_2}(t) = \begin{cases} 1 & \text{if } x_{T_1}(t)(0) > x_{T_2}(t)(0) \\ 0 & \text{if } x_{T_1}(t)(0) = x_{T_2}(t)(0). \end{cases} \quad (3.18)$$

If the number of arrivals at the workstation differs from the number of lots that have left the workstation, a lot must reside on the machine. If not, no lots reside on the machine.

- $x_{\chi_3}(t)$: Three situations are distinguished:
 - Currently, no lots reside in the workstation. The remaining process time is zero then.
 - Enough lots reside in the workstation to ensure that during the interval $[-\Delta, 0]$, the machine has constantly been busy. It is known then that after the last jump of $x_{\mathcal{T}_2}(t)$, a new lot was started. The remaining process times is the time instant of that last jump plus the nominal process time d .
 - A lot resides in the workstation, but it may have arrived after the most recent departure of a lot from the workstation. The additional infimum in the maximization checks this option.
- $x_{\chi_4}(t)$: The number of lots that have left the workstation at time t is equal to the value of $w_{\mathcal{T}_2}(t)$, but since this signal is not a state element, it cannot be used in the expression for $x_{\chi_4}(t)$. The evolution of $w_{\mathcal{T}_2}(t)$ over time interval $[-\Delta, 0]$ however is part of the state, so taking the value of this state element at its right boundary ($\tau = 0$) gives the correct number of lots that have left.

Map $\mathcal{M}_{\Theta \rightarrow \chi}$ has a simple and elegant structure (the cases in $x_{\chi_3}(t)$ only make sure that all infima exist), which makes it suitable for larger manufacturing systems (proportional scaling).

For any state $\mathbf{x}_{\chi}(t) \in \mathcal{B}_{\mathcal{T}s}^{\chi}$, one can find a corresponding non-empty set of min-plus states $\mathbf{x}_{\mathcal{T}}(t) \in \mathcal{B}_{\mathcal{T}s}^{\Theta}$ using the map $\mathcal{M}_{\chi \rightarrow \Theta} : \mathbb{N}^2 \times \mathbb{R} \times \mathbb{Z} \rightarrow ([-\Delta, 0] \rightarrow \mathbb{Z})^2$:

$$\mathcal{M}_{\chi \rightarrow \Theta} : \left\{ \begin{array}{l} \{\mathbf{x}_{\mathcal{T}}(t) \mid \exists \tilde{\mathbf{x}}_{\mathcal{T}}(t) : [-\Delta - d, d] \rightarrow \mathbb{Z}^2 \text{ with } \Delta \geq d \text{ such that:} \\ \dots\dots\dots \\ \tilde{x}_{\mathcal{T}_1}(t)(\tau) \leq \tilde{x}_{\mathcal{T}_2}(t)(\tau) + N_1 + 1 \\ \tilde{x}_{\mathcal{T}_2}(t)(\tau) = \min(\tilde{x}_{\mathcal{T}_1}(t)(\tau - d), \tilde{x}_{\mathcal{T}_2}(t)(\tau - d) + 1) \text{ for } \tau \in [-\Delta, d] \\ \dots\dots\dots \\ \tilde{x}_{\mathcal{T}_1}(t)(0) = x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) \\ \tilde{x}_{\mathcal{T}_2}(t)(\tau) = \begin{cases} x_{\chi_4}(t) & \text{for } \tau \in [x_{\chi_3}(t) - d, 0] \text{ if } x_{\chi_2}(t) > 0 \\ x_{\chi_4}(t) & \text{for } \tau = 0 \text{ if } x_{\chi_2}(t) = 0 \end{cases} \\ \tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d) \geq x_{\chi_4}(t) + 1 \text{ if } x_{\chi_2}(t) > 0 \\ \tilde{x}_{\mathcal{T}_2}(t)(\tau) = x_{\chi_4}(t) \text{ for } 0 \leq \tau < x_{\chi_3}(t) \text{ if } x_{\chi_2}(t) > 0 \\ \dots\dots\dots \\ \forall \varepsilon > 0, \forall \tau \in [-\Delta - d + \varepsilon, d] : \tilde{x}_{\mathcal{T}_i}(t)(\tau - \varepsilon) \leq \tilde{x}_{\mathcal{T}_i}(t)(\tau), i \in \{1, 2\} \\ \dots\dots\dots \\ \forall \tau \in [-\Delta, 0] \text{ and } i \in \{1, 2\} : x_{\mathcal{T}_i}(t)(\tau) = \tilde{x}_{\mathcal{T}_i}(t)(\tau) \}. \end{array} \right. \quad (3.19)$$

Map $\mathcal{M}_{\chi \rightarrow \Theta}$ can be explained informally as:

- The first and last line of the map concern the domain of the signals. Auxiliary signals $\tilde{\mathbf{x}}_{\mathcal{T}}(t)$ are used to make sure that time shifted signals, e.g. $\tau - d$ exist for the whole domain of $\mathbf{x}_{\mathcal{T}}(t)$. Signals $\mathbf{x}_{\mathcal{T}}(t)$ and $\tilde{\mathbf{x}}_{\mathcal{T}}(t)$ are equal on domain $[-\Delta, 0]$.

- The lines starting with $\tilde{x}_{\mathcal{T}_1}(t)(\tau)$ and $\tilde{x}_{\mathcal{T}_2}(t)(\tau)$ are similar to (3.6). In the expression for $\tilde{x}_{\mathcal{T}_1}(t)(\tau)$ the original minimization has been omitted, to make the map independent of an input signal.
- The next four lines set the boundaries of the signals ($\tilde{x}_{\mathcal{T}_i}(t)(0)$) at the correct values, based on the values of $\mathbf{x}_{\chi}(t)$.
- The remaining line of the map is a monotonicity requirement on all signals in $\tilde{\mathbf{x}}_{\mathcal{T}}(t)$.

Map $\mathcal{M}_{\chi \rightarrow \Theta}$ constructs the state elements in $\mathbf{x}_{\mathcal{T}}(t)$, which are a function over time interval (memory span) $[-\Delta, 0]$ with $\Delta \geq d$, as defined in (3.8). Since in general many states $\mathbf{x}_{\mathcal{T}}(t)$ in the min-plus model map to the same state $\mathbf{x}_{\chi}(t)$ in the hybrid χ model, a set of states $\mathbf{x}_{\mathcal{T}}(t)$ corresponds to a single state $\mathbf{x}_{\chi}(t)$. Therefore, map $\mathcal{M}_{\chi \rightarrow \Theta}$ returns a non-empty *set of states*. One can interpret this set as all possible states of the min-plus models that could have led to the current physical situation in the manufacturing system. Map $\mathcal{M}_{\chi \rightarrow \Theta}$ yields the complete set of feasible trajectories. In (3.19) the dynamics equations (3.6), conditions on the upper bound of interval $[-\Delta, 0]$, and the requirement for non-decreasing signals can be recognized. This map therefore can easily be extended for larger manufacturing systems, since it is scaling up proportionally. Furthermore, auxiliary state $\tilde{\mathbf{x}}_{\mathcal{T}}(t)$ was used to be able to construct feasible signals at the left boundary of interval $[-\Delta, 0]$, but this is of minor importance for understanding the map.

Both maps are complementary, i.e. if one starts with a right-continuous hybrid χ state $\mathbf{x}_{\chi}(t)$, maps it to a set of min-plus states $\mathbf{x}_{\mathcal{T}}(t)$ and then back again, the original state $\mathbf{x}_{\chi}(t)$ is returned. And the other way around: if one starts with a min-plus state $\mathbf{x}_{\mathcal{T}}(t)$, maps it to a hybrid χ state $\mathbf{x}_{\chi}(t)$ and then back to a set of min-plus states, the original state $\mathbf{x}_{\mathcal{T}}(t)$ lies within the resulting set. In other words:

Proposition 3.7.

$$\forall \mathbf{x}_{\chi}(t) \in \mathcal{B}_{\mathcal{T}_S}^{\chi} : \mathbf{x}_{\chi}(t) = \mathcal{M}_{\Theta \rightarrow \chi}(\mathcal{M}_{\chi \rightarrow \Theta}(\mathbf{x}_{\chi}(t))) \quad (3.20a)$$

and

$$\forall \mathbf{x}_{\mathcal{T}}(t) \in \mathcal{B}_{\mathcal{T}}^{\Theta} : \mathbf{x}_{\mathcal{T}}(t) \in \mathcal{M}_{\chi \rightarrow \Theta}(\mathcal{M}_{\Theta \rightarrow \chi}(\mathbf{x}_{\mathcal{T}}(t))). \quad (3.20b)$$

Proof. See Appendix A.1. □

3.3.3 Example

Consider the workstation as described in Section 3.2, with buffer capacity $N = 2$ and process time $d = 1$. Six lots are pushed through the system. Lots are available at time 0, 0.5, 3, 3, 3 and 3. The flow of lots through the system is shown in Figure 3.8. The horizontal axis is the time axis, the vertical axis shows the lot number. Blocks in the diagram indicate the presence of the lot in either the buffer or the machine. Note that although available at time = 3, the sixth

lot can only enter the buffer at time = 4, due to the buffer capacity. Until time = 4, this lot must remain at its source, e.g. a preceding workstation. Figures 3.3, 3.4 and 3.6 were taken from this example. At time = 4.5, one lot is on the machine with remaining process time 0.5 and two lots are in the buffer. The number of already finished lots equals 3, so $\mathbf{x}_\chi(4.5) = [2 \ 1 \ 0.5 \ 3]^T$. Assume that memory span $\Delta = 2 \geq d$. Applying map $\mathcal{M}_{\chi \rightarrow \Theta}$ from (3.19) gives a set of solutions for $x_{T_1}(4.5)$ and $x_{T_2}(4.5)$. One possible realization is given in Figure 3.9. This realization differs from the w_{T_1} and w_{T_2} graphs between time = 2.5 and 4.5 in Figure 3.4, but it is a feasible realization of the past which leads to the current situation in the manufacturing system. Applying map $\mathcal{M}_{\Theta \rightarrow \chi}$ (3.16) on the realization of Figure 3.9 yields the original state $\mathbf{x}_\chi(4.5) = [2 \ 1 \ 0.5 \ 3]^T$.

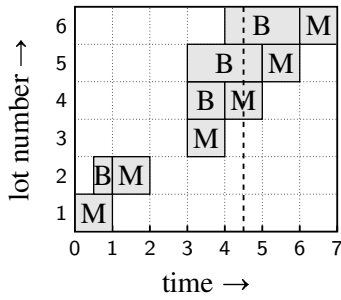


Figure 3.8: Lot-time diagram.

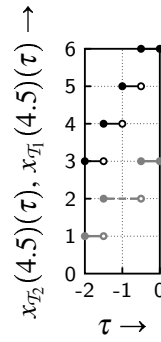


Figure 3.9: Possible realization for $x_{T_1}(4.5)$ (solid) and $x_{T_2}(4.5)$ (dashed).

3.4 Summary

In this chapter, a generic way of coupling different model types for manufacturing systems by means of maps between the states and signals of the models has been presented. Goal of this coupling method is to be able to use analysis techniques, real-time control methods, and performance measurement techniques which are either formulated in time domain or event domain. By coupling time domain and event domain models by state maps, the gap between those two domains is bridged. The models and maps presented in this chapter are generic and scalable in a sense that enlarging the manufacturing system under consideration results in proportional growth of the models and maps. For this reason, only a single workstation has been considered in this chapter. Specifying larger manufacturing systems in the presented model techniques (max-plus, min-plus and hybrid χ) is straightforward as shown in Chapter 2.

Receding horizon control using (multi-parametric) linear programming

In the previous chapters, different modelling techniques have been introduced and a coupling between discrete event and hybrid model types has been established. The ultimate goal of the research is to control manufacturing flow lines in a sense that the flow line should behave in a predetermined desired manner. In this chapter, feedback controllers are developed. Based on the state of manufacturing system under consideration, the feedback controllers determine new control actions. These control actions are intended to steer the system to the desired behavior/trajectories. When the control actions have been implemented, the resulting state is the basis for new control actions, yielding a control loop.

The high-level model and control framework has reached the point where controllers are to be designed. In the framework picture (Figure 4.1), the thick black arrow is elaborated on in the first part of this chapter. First, *model predictive control*, *linear programming* and *multi-parametric linear programming* are explained in preliminary Section 4.1. For a single workstation, a multi-parametric optimization problem is formulated in Section 4.2. The solution of this optimization problem is an optimal manufacturing schedule for a certain number of lots. Only the first step of this schedule is implemented and then a new schedule is determined only whenever the discrete state variables change. This is called

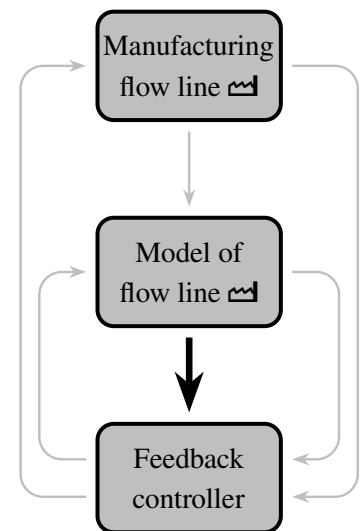


Figure 4.1: Next step in the framework: design of controller.

a *receding horizon strategy*. An important feature of this method is the off-line computation (optimization) of the control law. This facilitates large optimization problems that take quite some time to solve to be tractable in this situation, because optimization does not take place online. Moreover, off-line determination of a control law is safer than online optimization. In case of an emergency, a manufacturer does not want to await the results of an optimization solver. A second and even more important feature is that the method is carried out in the time domain whereas required re-determination of schedules is invoked by events that take place. In other words: if nothing happens, new schedules are not needed. On the other hand, during the time span between events the feedback controller keeps its optimal schedule. Since optimal schedules are available continuously in time (in the form of a look-up table or drawing a record from a database), a continuous (time) receding horizon controller has been obtained. Another manufacturing related work on continuous model predictive control is [96], in which a method to solve continuous linear programs is used which is described by Weiss [108].

Finally, the method is applied to a larger flow line. The controller is implemented in a discrete event simulation, which functions as the to be controlled manufacturing system. A generic methodology is developed to generate feedback controllers for flow lines of arbitrary length, with arbitrary buffer capacities and arbitrary control horizons.

4.1 Model predictive control and (multi-parametric) linear programming

This section provides preliminaries on model predictive control and the basics of linear programming and multi-parametric linear programming. Insight in the working principle of linear programming is necessary to understand multi-parametric linear programming.

4.1.1 Model predictive control

An interesting overview of model predictive control (MPC) techniques is given by Qin and Badgwell [89]. They describe MPC as a class of algorithms that compute a sequence of manipulated variable adjustments in order to optimize the future behavior of a plant. Originally developed in the 1970s to meet the specialized control needs of power plants and petroleum refineries, MPC technology can now be found in a wide variety of application areas including chemicals, food processing, automotive, aerospace, metallurgy, etc. Instead of going into technical details on MPC, the important features and principles are explained here.

A model predictive controller computes control actions for a plant in order to optimize future behavior. The length of this future is called a *horizon*. In most MPC techniques, two different horizons exist: a control horizon N_c and a prediction horizon N_p . The latter is always longer than (or at least equal to) the former. The control horizon is the time span over which the control actions may vary. The final control action is held constant during the remaining length

of the prediction horizon. The output variables are optimized over the prediction horizon, with the control actions over the control horizon as manipulative variables. For the predictions of the outputs an internal model is used. Figure 4.2 visualizes the concept of the two horizons, cf. [19]. After an optimal control action has been computed, only the first step is implemented. Then the state and/or outputs are measured again and new control actions are optimized over shifted horizons. This is called a receding horizon strategy. To keep the optimization problem finite dimensional, MPC is often used in a discrete time environment with fixed time steps. Continuous time receding horizon control over a finite and fixed time horizon is much more difficult. A paper related to this subject is by Weiss [108].

An important feature of MPC is that it is able to handle constraints on the manipulative variables, state and outputs. With the constraints it is possible to take physical limitations, safety regulations and performance requirements into account.

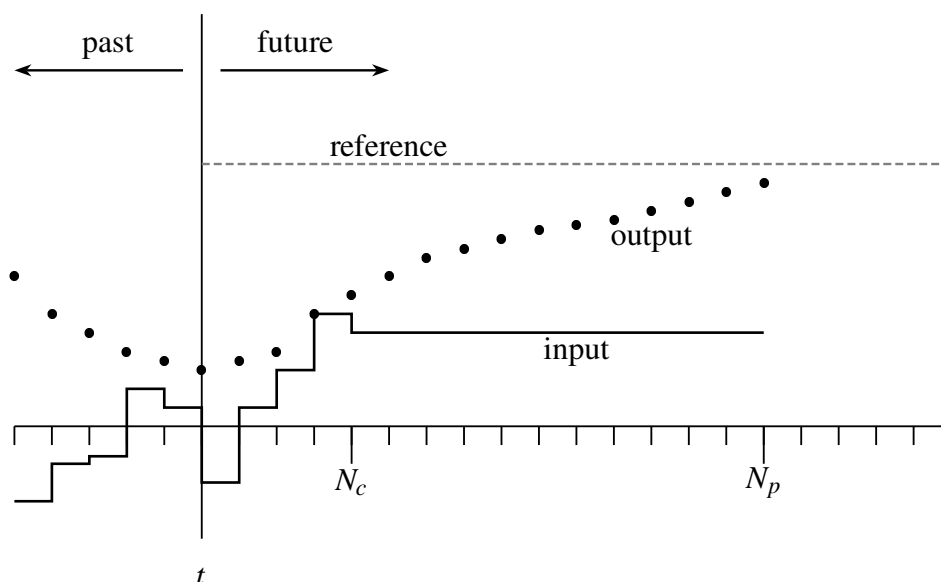


Figure 4.2: Model predictive control receding horizon principle.

An elegant model predictive control method for max-plus linear discrete event systems has been presented by De Schutter and Van den Boom [33]. Instead of discrete time MPC, they use an event horizon: a number of lots. They present an MPC control structure with a similar objective function as in this chapter. For practical industrial implementation, a causality issue emerges in [33]: in order to compute input time instants of a k th lot, the departure time of the $(k - 1)$ th lot is needed. In practice, this information is not yet available at the time instant it is needed. In [18] the same authors present a method to overcome this problem by estimating the unknown state components using forward iteration with the system matrices and combining estimated and actually measured time instants. Another difficulty in the max-plus linear setting (as mentioned in Chapter 3) is the fact that initial conditions of a factory (in time domain) are hard to catch in a max-plus linear initial condition. With the maps presented in Chapter 3, this issue can be resolved.

In this chapter, a slightly different than the conventional (discrete time) approach is used, based on the MPC principles: similar to [33], an optimal production schedule is computed for a control horizon (here equal to the prediction horizon), which is a fixed *number of lots* instead of time. In this chapter however, the evolution of signals and measurement of the state of the system take place in the time domain. With the multi-parametric linear programming technique, which is explained in Section 4.1.2, it is possible to compute optimal schedules for all possible states and for all possible times and due dates of the jobs. In this way, a continuous time predictive control method is obtained. The control actions are implemented at their scheduled times, and since the schedule updates itself continuously in time based on the current state, the receding horizon principle is embodied. Contrary to the method in [33], the feedback law is completely computed offline, yielding an explicit feedback structure.

4.1.2 Linear programming and multi-parametric linear programming

Invented in 1939 by Leonid Vitaliyevich Kantorovich, linear programming (LP) became widely known by the work of George Dantzig [31]. Kantorovich received the Nobel Prize in Economics in 1975, together with Tjalling C. Koopmans “for their contributions to the theory of optimum allocation of resources” [1]. Although the invention of the linear programming method had contributed greatly to this achievement, Dantzig is often mentioned as the founding father of linear programming, cf. [31, 32].

Linear programming involves the optimization of a linear objective function subject to linear (in)equality constraints. The variables that can be varied to reach an optimum are called the *design variables* of the optimization problem. Every LP problem can be written into the following form:

$$\min_{\mathbf{x}} \mathbf{c}^T \mathbf{x} \quad (4.1a)$$

$$\text{subject to } \mathbf{x} \geq 0 \quad (4.1b)$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}. \quad (4.1c)$$

Expression (4.1a) is called the *objective function*, in which vector \mathbf{c} weighs the elements of design variable vector \mathbf{x} . The inequalities in (4.1b) are the *non-negativity* constraints, while (4.1c) embodies all other linear constraints. Note that inequalities (4.1b) can be incorporated into inequality set (4.1c), but they are often mentioned separately. Any other form than (4.1) can be rewritten into this general form. One could think of maximization objectives, equality constraints and non-positivity constraints. Details on this rewriting procedure are given in [104].

Example 4.1. Consider the objective function

$$\min_{x_1, x_2} -(x_1 + x_2)$$

subject to the non-negativity constraints $x_1 \geq 0$ and $x_2 \geq 0$ and additionally the following constraints:

$$x_1 + 4x_2 \leq 16, \quad 8x_1 + 5x_2 \leq 56, \quad -x_1 + x_2 \leq 1.$$

The design variables are x_1 and x_2 . Graphically, the *feasible domain*, i.e. the domain of the design variables in which no constraints are violated, and the objective function are shown in Figure 4.3. The feasible domain has been shaded gray, from darkgray to lightgray. The arrow indicates the direction of decreasing objective function. The black dot is the solution of this LP problem:

$$x_1 = 5\frac{1}{3} \text{ and } x_2 = 2\frac{2}{3}.$$

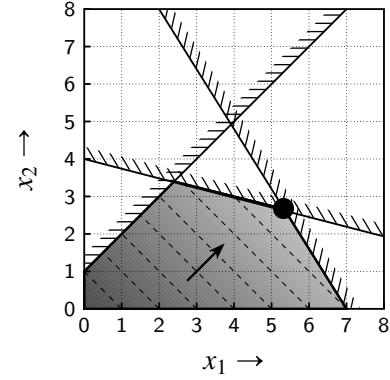


Figure 4.3: Linear programming example.

In practice the coefficients of all constraints and of the objective function are not always known exactly beforehand. The solution of the optimization problem may depend heavily on the exact values of these coefficients, for example if the particular constraints are *active*. A constraint is active in the optimal solution, if the constraint holds with equality. If one is interested for the optimal solution for all possible values of certain coefficients, the optimization problem turns into a *multi-parametric* optimization problem.

A few years after the invention of the simplex method (a quick and efficient way to solve linear programs), the term *parametric programming* appeared for the first time. In the early 1950s a few papers appeared on the subject and the multitude of articles on parametric programming grew over the years. A historical overview of this method is given by Gal in [44, 45].

Often used in economical decision problems and management sciences, multi-parametric programming is used in the systems and control domain mainly for optimal control of constrained linear and piecewise affine systems (PWA systems have been introduced in Section 2.3.1), see Bemporad [12], Bemporad et al. [13], Borrelli et al. [20]. Multi-parametric programming techniques provide an explicit control law for these systems by means of off-line computation of this law. From a system and control theoretic point of view, one could interpret this as follows: for all possible states $\mathbf{x}(t)$ an optimal control action $\mathbf{u}(t)$ is computed off-line. Different multi-parametric programming techniques exist, e.g. multi-parametric linear programming, multi-parametric quadratic programming and multi-parametric mixed integer linear and quadratic programming. An overview is given in the manual of the multi-parametric toolbox (MPT) for MATLAB [68]. In this thesis, only multi-parametric *linear* programming problems (MPLP problems) are considered.

Remark 4.2. It is important to notice that the design variables are stored in vector \mathbf{u} , rather than in vector \mathbf{x} , which is now used for the parameters that may vary. Notice also that the term *parameter vector* is used for \mathbf{x} , since it may contain more elements than the state elements of a system only.

The general form of a multi-parametric linear program with its constraints is:

$$V(\mathbf{x}) = \min_{\mathbf{u}} \left(\mathbf{H}^T \mathbf{u} + \mathbf{F}^T \mathbf{x} \right) \quad (4.2a)$$

$$\text{subject to } \mathbf{G}\mathbf{u} \leq \mathbf{W} + \mathbf{E}\mathbf{x} \quad (4.2b)$$

$$(\mathbf{A}\mathbf{x} \leq \mathbf{b}) \quad (4.2c)$$

in which (4.2a) is the objective function. The problem may be bound by linear constraints (4.2b) which involve the state \mathbf{x} and the input \mathbf{u} . Linear constraints that depend on \mathbf{x} only can be included in (4.2b), but can also be expressed in the familiar form of (4.2c). In numerical implementations, like in the multi-parametric toolbox for MATLAB [68], the form of (4.2c) is mandatory. The solution consists of N regions, which are defined by the polyhedral partition P_n with $n \in \{1, 2, \dots, N\}$:

$$P_n(i) = \{\mathbf{x} : \mathbf{H}_n \mathbf{x} \leq \mathbf{K}_n\} \quad (4.3)$$

with the optimal control law and corresponding cost function expression:

$$\mathbf{u} = \mathbf{F}_i \mathbf{x} + \mathbf{G}_i \quad (4.4a)$$

$$V(\mathbf{x}) = \mathbf{B}_i \mathbf{x} + \mathbf{C}_i. \quad (4.4b)$$

Example 4.3. Consider again the linear programming problem of Example 4.1. Suppose that the first constraint is modified into:

$$x_1 + 4x_2 \leq a$$

with as domain for a : $16 \leq a \leq 28$. In Figure 4.4 the optimization problem is shown again for $a = 28$. The original constraint has been plotted in lightgray. As can be seen, the optimum has shifted to $x_1 = 3\frac{12}{13}$ and $x_2 = 4\frac{12}{13}$.

The MPLP problem is formulated as:

$$\mathbf{u} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T, \quad \mathbf{x} = a, \quad \min_{\mathbf{u}} \begin{bmatrix} -1 & -1 \end{bmatrix} \mathbf{u}$$

$$\text{subject to } \begin{bmatrix} 1 & 4 \\ 8 & 5 \\ -1 & 1 \end{bmatrix} \mathbf{u} \leq \begin{bmatrix} 0 \\ 56 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mathbf{x}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} -16 \\ 28 \end{bmatrix}.$$

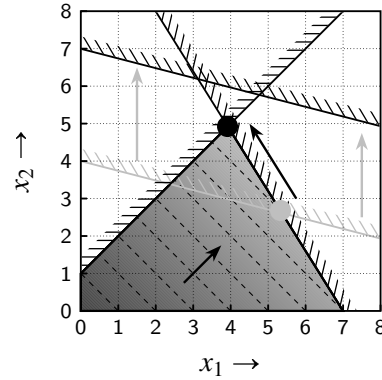


Figure 4.4: Multi-parametric linear programming example.

The solution consists of two regions, each with its own cost function:

$$\text{region 1: } \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} -16 \\ 23\frac{8}{13} \end{bmatrix}, \quad V(\mathbf{x}) = -\frac{1}{9}\mathbf{x} - 6\frac{2}{9}$$

$$\text{region 2: } \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mathbf{x} \leq \begin{bmatrix} -23\frac{8}{13} \\ 28 \end{bmatrix}, \quad V(\mathbf{x}) = -8\frac{11}{13}.$$

The transition from region 1 to region 2 is at $a = 23\frac{8}{13}$. For this value, the constraint line exactly crosses the optimum as indicated in Figure 4.4. If a increases further, the optimum remains the same, which can be seen in the corresponding cost function, which is constant.

The (hybrid) state space representation which was developed in Chapter 2 and elaborated on in Chapter 3 is used to develop a control strategy in time domain which optimizes over events, i.e. the start times of lots on machines. Developing a controller for the production planning of a single workstation using MPLP problems is treated in the next section.

4.2 Deriving a controller for a single workstation

Now that the basics of linear programming and multi-parametric linear programming (MPLP) have been explained, this section deals with the development of a controller for the production of lots in a single workstation. A receding horizon control method is used. The idea is as follows: the workstation's state is chosen to be as in the hybrid χ examples of Chapters 2 and 3. The state therefore is finite dimensional and consists of three elements. For all possible state vectors, an optimal control action is to be computed. Therefore, the state of the workstation is part of vector \mathbf{x} in the MPLP structure: the solution is parametrized over all state realizations. The optimal schedule that is computed by the MPLP problem is then implemented. The schedule is optimal at every moment in time. A continuous (time) receding horizon controller has been obtained then.

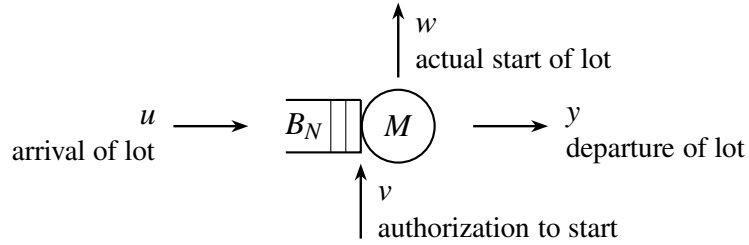


Figure 4.5: Workstation consisting of a buffer with finite storage capacity N and a single-lot machine. Variables u , v , w and y denote different input and output signals.

4.2.1 System characterization, state and inputs

Consider the workstation as shown in Figure 4.5. It consists of a buffer B_N with finite capacity N and a single-lot machine with process time d . The hybrid state as introduced in Section 2.3.3 is used:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \in \mathbb{N} \\ x_2(t) \in \mathbb{N} \\ x_3(t) \in \mathbb{R}_+ \end{bmatrix} = \begin{bmatrix} \text{number of lots in } B_N \text{ at time } t \\ \text{number of lots on machine } M \text{ at time } t \\ \text{remaining process time of current lot on } M \text{ at time } t \end{bmatrix}.$$

Two different types of input signals are used: $u_i(t)$ is the time instant at which the i -th next lot is fed into the workstation, computed at time t . Input signal $v_i(t)$ is the time instant at which machine M is scheduled to start for the i -th time. Inputs u_i and v_i not necessarily correspond to the same lot. When lots reside in the buffer, $v_1(t)$ denotes the time instant that the first lot from the buffer starts on the machine, scheduled at time t and $u_1(t)$ is the time instant the first new lot is to arrive at the buffer, scheduled at time t . All u_i and v_i are stored in vectors \mathbf{u} and \mathbf{v} respectively. In addition to input signals \mathbf{u} and \mathbf{v} , let vector \mathbf{w} denote the time instants lots actually start on the machine. This is a sort of auxiliary variable, since in the optimization, the authorization signal is computed exactly as the time instant the machine starts processing.

The control law for scheduling lots on the workstation is computed by means of an MPLP problem in this chapter. The design variables of the optimization problem are the elements of \mathbf{u} and \mathbf{v} . The parameters of the multi-parametric optimization problem are state \mathbf{x} , current time t and reference vector \mathbf{r} . The time t is included, since it is not possible to schedule the start of a job back in time. Note that a time dependent system has been obtained now. Reference vector \mathbf{r} contains the time instants at which lots should be finished, also called *due dates*. The length of \mathbf{r} is the horizon over which the schedule is optimized. This *control horizon* is denoted by N_c and equals the prediction horizon in this chapter: $N_c = N_p$. Due dates are assumed to be in chronological order, which translates into constraints on the optimization problem. Whenever a lot is finished on the machine and leaves the system, the corresponding due date is removed from \mathbf{r} (from the top) and a new due date is added to this vector (at the bottom).

To prevent confusion, the control action vector and parameter vector of the MPLP problem are denoted by capitals \mathbf{U} and \mathbf{X} respectively.

4.2.2 Control objective

The control objective is to minimize the difference between the due date and the actual completion time of a lot. Both earliness and tardiness of lots are to be penalized. Let $y_i(t)$ denote the time instant the i -th lot leaves the system, computed at time t . Note that y does not necessarily equal the start time on the machine, v , plus the process time d . Particularly when a flow line is modelled, blocking effects may force that y becomes strictly greater than $v + d$. Linear costs are put on the penalty. The main objective is thus to minimize the absolute value of the difference between y_i and r_i for all lots. Within this objective, lots are to be fed to the system and started on the machine as late as possible. The reason for this additional control objective is that *value* is to be added as late as possible to lots. Every production step costs money in practice, so value is added to lots as they are being processed. Manufacturers tend to keep the expenses as low as possible, so all production steps should be performed as late as possible. This additional objective is weighed lower than the main objective of minimizing the output error. The total control objective is:

$$\min_{\mathbf{u}, \mathbf{v}} -\lambda \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix}^T + \sum_{i=1}^{N_c} |y_i - r_i|. \quad (4.5)$$

Vector $\lambda \geq 0$ (element-wise) is the weighting vector that is used to prioritize the main control goal of minimizing the output error. Typically, $\lambda \ll 1$ contains small values. The secondary control objective (which implies just-in-time processing) also makes sure that the values of \mathbf{v} and \mathbf{w} are forced to be equal in the optimization: the machine starts only when it minimizes the output error and authorization signals are sent to the machine as late as possible. Thus ideally, v_i and w_i are equal in the solution of the MPLP problem.

The control objective as stated in (4.5) is not directly suitable to fit in a linear program, because the absolute value function is non-linear. Therefore, auxiliary variables z_i are introduced:

$$z_i = |y_i - r_i| = \max(y_i - r_i, r_i - y_i). \quad (4.6)$$

The control objective then translates into the following expression with additional constraints:

$$\min |y_i - r_i| = \min z_i \quad (4.7a)$$

$$\text{subject to } z_i \geq y_i - r_i \quad (4.7b)$$

$$z_i \geq r_i - y_i. \quad (4.7c)$$

Auxiliary variables z_i are stored in vector \mathbf{z} and are part of the multi-parametric program. They are not to be parametrized, so they are put in the control action vector \mathbf{U} . For the same reason variables y_i are also part of \mathbf{U} , which is composed as follows:

$$\mathbf{U} = [\mathbf{u} \quad \mathbf{v} \quad \mathbf{w} \quad \mathbf{y} \quad \mathbf{z}]^T. \quad (4.8)$$

So far, the workstation's state and inputs have been defined and the control objective has been stated. The constraints for the multi-parametric linear program can now be constructed. It appears that several MPLP problems need to be formulated, based on the different possible initial conditions of the workstation. This is elaborated in the next section.

4.2.3 Initial conditions, constraints and set of MPLP problems

Number of design variables

As mentioned earlier, optimization takes place over a certain horizon N_c , the number of lots that is to be scheduled. Although N_c is chosen beforehand and remains fixed, the number of design variables (in $\mathbf{U}(t)$) is not fixed, but depends on state $\mathbf{x}(t)$ of the workstation. This can be explained as follows: when a schedule is made for a workstation that contains lots in the buffer, fewer new lots should arrive at the workstation than when a schedule is made for an empty workstation. Thus the number of elements in \mathbf{u} is dependent on x_1 and x_2 . In addition, if a workstation is currently busy processing a lot, the number of lots that has to start on the machine decreases by one, so the number of elements in \mathbf{v} depends on x_2 . Straightforwardly solving one MPLP problem is therefore not sufficient to compute a scheduling controller. This problem is solved by formulating multiple MPLP problems. For each possible state realization that leads to a different number of design variables, a different MPLP problem is formulated. The

conditions of the regions of the solutions of the individual MPLP problems are then extended with the state condition that led to this MPLP problem. In this way, by proper bookkeeping, all solutions can be put in one overall solution, containing the feedback control law for the schedules. The number of jobs in the buffer x_1 and the number of jobs on the machine x_2 are not part of the individual MPLP problems anymore. Parameter vector \mathbf{X} therefore only consists of the following elements:

$$\mathbf{X} = \begin{bmatrix} \mathbf{r}(t) \\ t \\ x_3(t) \end{bmatrix}. \quad (4.9)$$

The number of elements in vector \mathbf{U} is given by:

$$\begin{aligned} \dim \mathbf{u} &= \max(N_c - x_1 - x_2, 0) \\ \dim \mathbf{v} &= \max(N_c - x_2, 0) \\ \dim \mathbf{w} &= \dim \mathbf{v} \\ \dim \mathbf{y} &= N_c \\ \dim \mathbf{z} &= \dim \mathbf{y} \end{aligned}$$

where the max operators prevent negative numbers of design variables, e.g. when the x_1 or x_2 outnumber control horizon N_c . An important issue is that the number of MPLP problems may grow to infinity now. This can happen when the buffer has infinite storage capacity. For all possible state realizations that lead to a different number of design variables, a separate MPLP problem is constructed. Actual buffer contents value x_1 is allowed to grow arbitrarily high, so does the number of different design variables and thus the number of MPLP problems. A way to overcome this problem is to exclude the set of MPLP problems which do not contribute to the number of different solutions. For example, if the buffer storage capacity $N = 5$ and control horizon $N_c = 3$, it is only useful to construct the MPLP problems for $x_1 \in \{1, 2, 3\}$ and even, if $x_2 = 1$, only $x_1 \in \{1, 2\}$ gives new MPLP problems. The max operators in determining the dimensions of \mathbf{u} and \mathbf{v} already take care of this issue, but in an implementation, one should take care not to construct multiple MPLP problems for all realizations of x_1 and x_2 that result in $\dim \mathbf{u} = 0$ or $\dim \mathbf{v} = 0$, since all those MPLP problems are identical.

Constraints and bounds on the exploration space

As explained in the previous section, the number of design variables depends on the initial state (i.e. the state at each moment of re-scheduling) and this results in a number of MPLP problems that need to be solved. The constraints in the MPLP problem as a consequence also vary with the number of design variables and the initial state. In this section, the *constraint classes* are introduced. When implementing the formulation of the MPLP problem set, bookkeeping should make sure that all constraints get the correct subscript indices.

The constraints in the MPLP problems look very similar to the terms in the max-plus algebraic expressions in Section 2.1.1. The difference however is that in the max-plus models, all expressions involve equalities, while in the MPLP problems all expressions are inequalities. In the max-plus model, lots are pushed through the workstation, while in this chapter lots are

pulled out of the workstation, by computing the desired arrival time and start time of a lot at the workstation based on the due date. The inequalities ensure that events can be delayed by the authorization signals v_i on the one hand, and that the optimization solver forces the solution to lie at certain constraint boundaries.

The constraint classes that can be distinguished are as follows. Note that indices i and j are determined by the initial state.

- Authorization to start a lot on a machine is only useful when the lot has arrived: $v_i \geq u_j$.
- The machine can only start a job when it is authorized: $w_i \geq v_i$.
- The machine can only start a job when that job is present: $w_i \geq u_j$.
- A lot can only leave a machine once the process time has been completed: $y_i \geq w_j + d$.
- A machine can only start a job when the previous lot has been sent away: $w_i \geq y_j$.
- Due to buffer capacity constraints, a job can only enter the buffer after a certain job was started on the machine: $u_i \geq w_j$. In case of infinite buffer storage capacity, these constraints can be lifted.
- The absolute function was re-casted into auxiliary variables \mathbf{z} with additional constraints, yielding: $z_i \geq y_i - r_i$ and $z_i \geq r_i - y_i$.
- If a lot resides on the machine ($x_2 = 1$), then it can only leave after its remaining process time: $y_1 \geq x_3 + t$.
- All time instants that are computed are in the future, since it is not possible to change the past: $u_i \geq t$, $v_i \geq t$, $w_i \geq t$, $y_i \geq t$.
- Lots do not overtake: $u_{i+1} \geq u_i$, $v_{i+1} \geq v_i$, $w_{i+1} \geq w_i$ and $y_{i+1} \geq y_i$.

In addition to these constraints on the design variables, some constraints set the bounds on the exploration space. Theoretically, these bounds are not necessary, as shown in the theory and example in a paper by Bemporad et al. [13]. However, for numerical implementation purposes, as in the MATLAB MPT toolbox, the value of the objective function needs to be bounded in each of the polyhedra. This is achieved by specifying bounds on the exploration space. These bounds have to be put in the $\mathbf{Ax} \leq \mathbf{b}$ inequalities of the MPLP problem formulation:

- It is assumed that the reference vector contains due dates in chronological order: $r_{i+1} \geq r_i$.
- The remaining process time $x_3 \in [0, d]$.
- All parameters need a lower and an upper bound: $0 \leq r_i \leq \Omega$ and $0 \leq t \leq \Omega$, in which Ω represents a very large number. This upper bound is never allowed to become an active constraint, since then it influences the schedule. The upper bounds only facilitate the use of the MPT toolbox.
- Time is assumed to be non-negative and is upper bounded for the same reasons as the upper bounds on the due dates: $0 \leq t \leq \Omega$.

As mentioned before, the values of subscripts i and j depend on the initial state of the system and have to be taken care of by means of proper bookkeeping. All ingredients of a multi-parametric linear program have been treated now. Parameter vector \mathbf{X} and control vector \mathbf{U} have been defined. The control objective has been stated as a minimization of the earliness and

tardiness of lots and a just-in-time production policy. The receding horizon method based on events caused the problem to be decomposed into several subproblems. Based on the state of the workstation, a specific MPLP problem needs to be solved. All possible MPLP problems can be formulated beforehand and solved. The set of solutions of all individual problems is the feedback control law for the production scheduling of the workstation.

4.2.4 Case study: development of a controller for a workstation

The receding horizon feedback controller that has been explained in the previous sections is developed for a specific numerical example in this section. A single workstation is considered, as shown in Figure 4.5. The process time of the machine, d , is 3 hours and the buffer storage capacity equals 5 lots. A control horizon N_c of two lots is used. This means that the manufacturing schedule for the first two lots is optimized. The receding horizon principle facilitates that once the first lot has left the system, optimization takes place over the next two lots.

Parameter vector \mathbf{X} is defined as follows:

$$\mathbf{X} = \begin{bmatrix} r_1 & r_2 & t & x_3 \end{bmatrix}^T$$

which contains two due dates, current time t and the remaining process time of a lot on the machine. State elements x_1 and x_2 are not explicit part of the MPLP problems, but all useful combinations within the control horizon specify separate MPLP problems, as explained in the previous section. The following combinations of (x_1, x_2) result in unique MPLP problems:

$$(x_1, x_2) \in \{(0, 0), (0, 1), (1, 0), (1, 1), (2, 0)\}.$$

In case $x_1 = 2$ and $x_2 = 1$ and a control action is needed (e.g. in an initial situation or in a stochastic environment), one should compute the control action for the $x_1 = 1$ and $x_2 = 1$ situation, because it yields the same MPLP problem. On the next pages, the five MPLP problems are built up in detail. First, the design variables vector is stated, which differs for each situation. Vectors \mathbf{H} and \mathbf{F} that define the objective function are then presented. They prioritize the minimization of the output error $|y_i - r_i|$ and force just-in-time manufacturing within this primary objective. Then the constraints are formulated for that set of design variables. The order of constraints is the same as in the explanation given in Section 4.2.3, where the physical interpretation of each constraint class was given. In addition to the constraints that involve design variables, the bounds on the exploration space are set, in which upper bound Ω is set to 1000. In an implementation, one should keep in mind that these upper bound constraints are never to be active. At the right hand side of each situation, the solution of the MPLP problem is presented. Each solution consists of four regions. The regions are bounded by constraints. Only the constraints that are additive to the bounds on the exploration space are shown. Finally, for each region the control actions \mathbf{u} and \mathbf{v} are presented, expressed as a function of the parameter vector \mathbf{X} . The other design variables \mathbf{w} , \mathbf{y} and \mathbf{z} are not presented here, although they are also expressions in \mathbf{X} . However, these design variables involve non-manipulative variables or auxiliary variables. It is possible that the system's state is part of multiple regions. It is on the

edge of the regions then, where the inequalities hold with strict equality. This is not a problem, because of the continuity of the control action at the edges, see [13]. In the region characterizations, control actions and constraints, the process time $d = 3$ of the machine is indicated by variable d . Note that this variable is not part of the parameter vector \mathbf{X} .

Situation: MPLP problem structure:

$$\begin{aligned} x_1 = 0 & \quad \mathbf{U} = [u_1 \ u_2 \ v_1 \ v_2 \ w_1 \ w_2 \ y_1 \ y_2 \ z_1 \ z_2]^T \\ x_2 = 0 & \quad \mathbf{H} = [-\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ 0 \ 0 \ 1 \ 1]^T \\ & \quad \mathbf{F} = [0 \ 0 \ 0 \ 0]^T \text{ and } \varepsilon = \frac{1}{100} \ll 1 \end{aligned}$$



Constraints:

$$\begin{aligned} v_1 &\geq u_1, v_2 \geq u_2 \\ w_1 &\geq v_1, w_2 \geq v_2 \\ w_1 &\geq u_1, w_2 \geq u_2 \\ y_1 &\geq w_1 + d, y_2 \geq w_2 + d \\ w_2 &\geq y_1 \\ z_1 &\geq y_1 - r_1, z_1 \geq r_1 - y_1, z_2 \geq y_2 - r_2, z_2 \geq r_2 - y_2 \\ u_1 &\geq t, u_2 \geq t, v_1 \geq t, v_2 \geq t, w_1 \geq t, w_2 \geq t, y_1 \geq t, y_2 \geq t \\ u_2 &\geq u_1, v_2 \geq v_1, w_2 \geq w_1, y_2 \geq y_1 \end{aligned}$$

Bounds on the exploration space:

$$\begin{aligned} x_3 &\geq 0 \\ t &\geq 0 \\ r_1 &\geq 0 \\ r_2 &\geq 0 \\ r_2 &\geq r_1 \end{aligned}$$

Solution: 4 regions.

Region 1:

$$r_2 \leq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$u_1 = r_1 - d, \quad v_1 = r_1 - d$$

$$u_2 = r_1, \quad v_2 = r_1$$

Region 2:

$$r_2 \geq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$u_1 = r_1 - d, \quad v_1 = r_1 - d$$

$$u_2 = r_2 - d, \quad v_2 = r_2 - d$$

Region 3:

$$r_2 \leq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$u_1 = t, \quad v_1 = t$$

$$u_2 = t + d, \quad v_2 = t + d$$

Region 4:

$$r_2 \geq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$u_1 = t, \quad v_1 = t$$

$$u_2 = r_2 - d, \quad v_2 = r_2 - d$$

Situation: MPLP problem structure:

$$\begin{aligned} x_1 = 1 & \quad \mathbf{U} = [u_1 \ v_1 \ v_2 \ w_1 \ w_2 \ y_1 \ y_2 \ z_1 \ z_2]^T \\ x_2 = 0 & \quad \mathbf{H} = [-\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ 0 \ 0 \ 1 \ 1]^T \\ & \quad \mathbf{F} = [0 \ 0 \ 0 \ 0]^T \text{ and } \varepsilon = \frac{1}{100} \ll 1 \end{aligned}$$



Constraints:

$$\begin{aligned} v_2 &\geq u_1 \\ w_1 &\geq v_1, w_2 \geq v_2 \\ w_2 &\geq u_1 \\ y_1 &\geq w_1 + d, y_2 \geq w_2 + d \\ w_2 &\geq y_1 \\ z_1 &\geq y_1 - r_1, z_1 \geq r_1 - y_1, z_2 \geq y_2 - r_2, z_2 \geq r_2 - y_2 \\ u_1 &\geq t, v_1 \geq t, v_2 \geq t, w_1 \geq t, w_2 \geq t, y_1 \geq t, y_2 \geq t \\ v_2 &\geq v_1, w_2 \geq w_1, y_2 \geq y_1 \end{aligned}$$

Bounds on the exploration space:

$$\begin{aligned} x_3 &\geq 0 \\ t &\geq 0 \\ r_1 &\geq 0 \\ r_2 &\geq 0 \\ r_2 &\geq r_1 \end{aligned}$$

Solution: 4 regions.

Region 1:

$$r_2 \leq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$u_1 = r_1, \quad v_1 = r_1 - d$$

$$v_2 = r_1$$

Region 2:

$$r_2 \geq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$u_1 = r_2 - d, \quad v_1 = r_1 - d$$

$$v_2 = r_2 - d$$

Region 3:

$$r_2 \leq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$u_1 = t + d, \quad v_1 = t$$

$$v_2 = t + d$$

Region 4:

$$r_2 \geq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$u_1 = r_2 - d, \quad v_1 = t$$

$$v_2 = r_2 - d$$

Situation: MPLP problem structure:

$$\begin{aligned}
 x_1 &= 2 & \mathbf{U} &= [v_1 \ v_2 \ w_1 \ w_2 \ y_1 \ y_2 \ z_1 \ z_2]^T \\
 x_2 &= 0 & \mathbf{H} &= [-\varepsilon \ -\varepsilon \ -\varepsilon \ -\varepsilon \ 0 \ 0 \ 1 \ 1]^T \\
 & & \mathbf{F} &= [0 \ 0 \ 0 \ 0]^T \text{ and } \varepsilon = \frac{1}{100} \ll 1
 \end{aligned}$$

**Constraints:**

$$\begin{aligned}
 w_1 &\geq v_1, w_2 \geq v_2 \\
 y_1 &\geq w_1 + d, y_2 \geq w_2 + d \\
 w_2 &\geq y_1 \\
 z_1 &\geq y_1 - r_1, z_1 \geq r_1 - y_1, z_2 \geq y_2 - r_2, z_2 \geq r_2 - y_2 \\
 v_1 &\geq t, v_2 \geq t, w_1 \geq t, w_2 \geq t, y_1 \geq t, y_2 \geq t \\
 v_2 &\geq v_1, w_2 \geq w_1, y_2 \geq y_1
 \end{aligned}$$

Bounds on the exploration space:

$$\begin{aligned}
 x_3 &\geq 0 \\
 t &\geq 0 \\
 r_1 &\geq 0 \\
 r_2 &\geq 0 \\
 r_2 &\geq r_1
 \end{aligned}$$

Solution: 4 regions.**Region 1:**

$$r_2 \leq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$\begin{aligned}
 v_1 &= r_1 - d \\
 v_2 &= r_1
 \end{aligned}$$

Region 2:

$$r_2 \geq r_1 + d, \quad r_1 \geq t + d$$

Control action:

$$\begin{aligned}
 v_1 &= r_1 - d \\
 v_2 &= r_2 - d
 \end{aligned}$$

Region 3:

$$r_2 \leq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$\begin{aligned}
 v_1 &= t \\
 v_2 &= t + d
 \end{aligned}$$

Region 4:

$$r_2 \geq t + 2d, \quad r_1 \leq t + d$$

Control action:

$$\begin{aligned}
 v_1 &= t \\
 v_2 &= r_2 - d
 \end{aligned}$$

Situation: MPLP problem structure:

$$\begin{aligned}
 x_1 &= 0 & \mathbf{U} &= [u_1 \ v_1 \ w_1 \ y_1 \ y_2 \ z_1 \ z_2]^T \\
 x_2 &= 1 & \mathbf{H} &= [-\varepsilon \ -\varepsilon \ -\varepsilon \ 0 \ 0 \ 1 \ 1]^T \\
 & & \mathbf{F} &= [0 \ 0 \ 0 \ 0]^T \text{ and } \varepsilon = \frac{1}{100} \ll 1
 \end{aligned}$$

**Constraints:**

$$\begin{aligned}
 v_1 &\geq u_1 \\
 w_1 &\geq v_1 \\
 w_1 &\geq u_1 \\
 y_1 &\geq t + x_3 \\
 y_2 &\geq w_1 + d \\
 w_1 &\geq y_1 \\
 z_1 &\geq y_1 - r_1, z_1 \geq r_1 - y_1, z_2 \geq y_2 - r_2, z_2 \geq r_2 - y_2 \\
 u_1 &\geq t, v_1 \geq t, w_1 \geq t, y_1 \geq t, y_2 \geq t \\
 y_2 &\geq y_1
 \end{aligned}$$

Bounds on the exploration space:

$$\begin{aligned}
 x_3 &\geq 0 \\
 t &\geq 0 \\
 r_1 &\geq 0 \\
 r_2 &\geq 0 \\
 r_2 &\geq r_1
 \end{aligned}$$

Solution: 4 regions.**Region 1:**

$$r_2 \leq r_1 + d, \quad t + x_3 \leq r_1$$

Control action:

$$u_1 = r_1, \quad v_1 = r_1$$

Region 2:

$$r_2 \geq r_1 + d, \quad t + x_3 \leq r_1$$

Control action:

$$u_1 = r_2 - d, \quad v_1 = r_2 - d$$

Region 3:

$$r_2 \leq t + x_3 + d, t + x_3 \geq r_1$$

Control action:

$$u_1 = t + x_3, \quad v_1 = t + x_3$$

Region 4:

$$r_2 \geq t + x_3 + d, t + x_3 \geq r_1$$

Control action:

$$u_1 = r_2 - d, \quad v_1 = r_2 - d$$

Situation: MPLP problem structure:

$$\begin{aligned}
 x_1 = 1 \quad & \mathbf{U} = [v_1 \ w_1 \ y_1 \ y_2 \ z_1 \ z_2]^T \\
 x_2 = 1 \quad & \mathbf{H} = [-\varepsilon \ -\varepsilon \ 0 \ 0 \ 1 \ 1]^T \\
 & \mathbf{F} = [0 \ 0 \ 0 \ 0]^T \text{ and } \varepsilon = \frac{1}{100} \ll 1
 \end{aligned}$$

**Constraints:**

$$\begin{aligned}
 w_1 &\geq v_1 \\
 y_1 &\geq t + x_3 \\
 y_2 &\geq w_1 + d \\
 w_1 &\geq y_1 \\
 z_1 &\geq y_1 - r_1, z_1 \geq r_1 - y_1, z_2 \geq y_2 - r_2, z_2 \geq r_2 - y_2 \\
 v_1 &\geq t, w_1 \geq t, y_1 \geq t, y_2 \geq t \\
 y_2 &\geq y_1
 \end{aligned}$$

Bounds on the exploration space:

$$\begin{aligned}
 x_3 &\geq 0 \\
 t &\geq 0 \\
 r_1 &\geq 0 \\
 r_2 &\geq 0 \\
 r_2 &\geq r_1
 \end{aligned}$$

Solution: 4 regions.**Region 1:**

$$r_2 \leq r_1 + d, \quad t + x_3 \leq r_1$$

Control action:

$$v_1 = r_1$$

Region 2:

$$r_2 \geq r_1 + d, \quad t + x_3 \leq r_1$$

Control action:

$$v_1 = r_2 - d$$

Region 3:

$$r_2 \leq t + x_3 + d, t + x_3 \geq r_1$$

Control action:

$$v_1 = t + x_3$$

Region 4:

$$r_2 \geq t + x_3 + d, t + x_3 \geq r_1$$

Control action:

$$v_1 = r_2 - d$$

The feedback law has also been computed for different values of maximum buffer storage capacity N and different control horizons N_c . The feedback laws are not presented here, but Table 4.1 shows for each combination the number of MPLP problems that need to be solved and the number of regions within each MPLP problem. Computations have been made with the MPT toolbox [68]. From the table it can be concluded that the number of MPLP problems is determined by the storage capacity of the buffer. This is an expected result, since for every combination of x_1 and x_2 a separate MPLP problem is constructed. The number of regions within the MPLP problems however depends solely on the control horizon N_c . This is also not a surprise, since each individual MPLP problem is not dependent on x_1 and x_2 anymore. The number of regions grows exponentially with the control horizon.

In this section a feedback controller for a single workstation has been computed. Control horizon N_c has a big influence on the size of the optimization problems: a larger control horizon results in more regions per MPLP problem. Unfortunately, this curse of dimensionality works exponentially. For a control horizon of 2 lots, the feedback control law has completely been characterized in this section. With this controller, it is possible to compute optimal manufacturing schedules, by means of a simple evaluation of the control law, given the state of a manufacturing system, current time and the due dates of the lots. Recall that measuring the state of the workstation can be done instantaneously: the system does not need to be observed for a certain time span to obtain full state information.

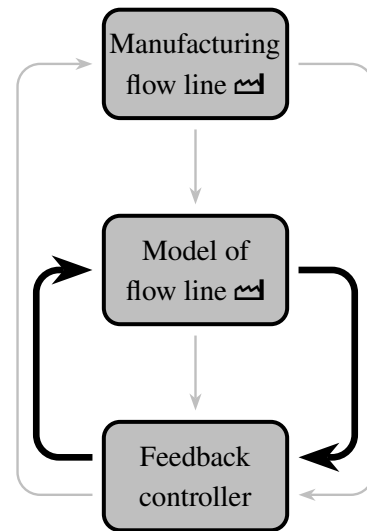
The presented method for developing a feedback controller can also be applied to other manufacturing entities, such as buffers with infinite storage capacities, batch machines or conveyors. In general, manufacturing systems in which product recipes, routes and order of processing are predetermined are suitable to be controlled with the method that is presented in this section.

Table 4.1: Number of MPLP problems and regions for varying workstation storage capacities and control horizons.

N_c	N	number of MPLP problems	number of regions within each MPLP problem
2	2	5	4, 4, 4, 4
3	2	6	10, 10, 10, 10, 10, 10
3	3	7	10, 10, 10, 10, 10, 10, 10
4	2	6	24, 24, 24, 24, 24, 24
4	3	8	24, 24, 24, 24, 24, 24, 24, 24
4	4	9	24, 24, 24, 24, 24, 24, 24, 24, 24
5	2	6	56, 56, 56, 56, 56, 60
5	3	8	56, 56, 56, 56, 56, 56, 60, 60
5	4	10	56, 56, 56, 56, 56, 56, 56, 60, 60, 60
5	5	11	60, 60, 60, 60, 60, 60, 60, 60, 60, 60, 60
6	1	4	148, 148, 148, 148
7	1	4	370, 370, 370, 370
8	1	4	920, 920, 920, 920
9	1	4	2300, 2300, 2300, 2300

4.2.5 Implementation of feedback controller

In the on-going process of modelling and control of manufacturing flow lines, the next step is taken: validation of the controller. In the previous section, a feedback control law was developed. For a single workstation consisting of a FIFO buffer with finite storage capacity and a single-lot machine, the control law has been presented for a control horizon $N_c = 2$. In this section, the control law is implemented in a simulation, to show its working (see the modelling and control framework in Figure 4.6). The results are plotted in a special type of graph, in which both axes represent time. The graphs show the schedules (control actions) at each time instant, they show the moments of rescheduling and whether the events occur as scheduled. In Section 4.2.4 it was also explained that enlarging the control horizon leads to larger optimization problems, resulting in more regions in the MPLP problem solutions. In this section, it is shown that enlarging the control horizon leads to smaller cumulative penalties on earliness and tardiness and can therefore be profitable. Deterministic simulations are carried out first. Afterwards, a simulation study is carried out in which the process times of the machine vary. It is shown that the controller keeps

**Figure 4.6:** Next step in the framework: implementation of controller.

updating its optimal schedule every time new state information is available.

The controller has been implemented in a simulation study in MATLAB. The process time of the machine $d = 3$ time units (hours) and the buffer has a storage capacity of two lots. The control horizon (i.e. the number of lots that are scheduled to finish as close as possible to their due dates) is set to $N_c = 2$. The due dates of lots are listed in chronological order in list \mathbf{R} :

$$\mathbf{R} = [7 \quad 12 \quad 14 \quad 15 \quad 20 \quad 25 \quad \infty \quad \dots]$$

which means that in this particular simulation study only the first six lots are taken into account. Lot number 7 and beyond have a due date which is artificially set to ∞ , so they do not influence the schedule for the first six lots. Due dates vector \mathbf{r} that is used to compute the control actions consists of the first two elements of \mathbf{R} and everytime a lots leaves the system, the corresponding (first) due date is removed from \mathbf{R} . The due dates at infinity ensure that the dimension of \mathbf{r} can always be N_c .

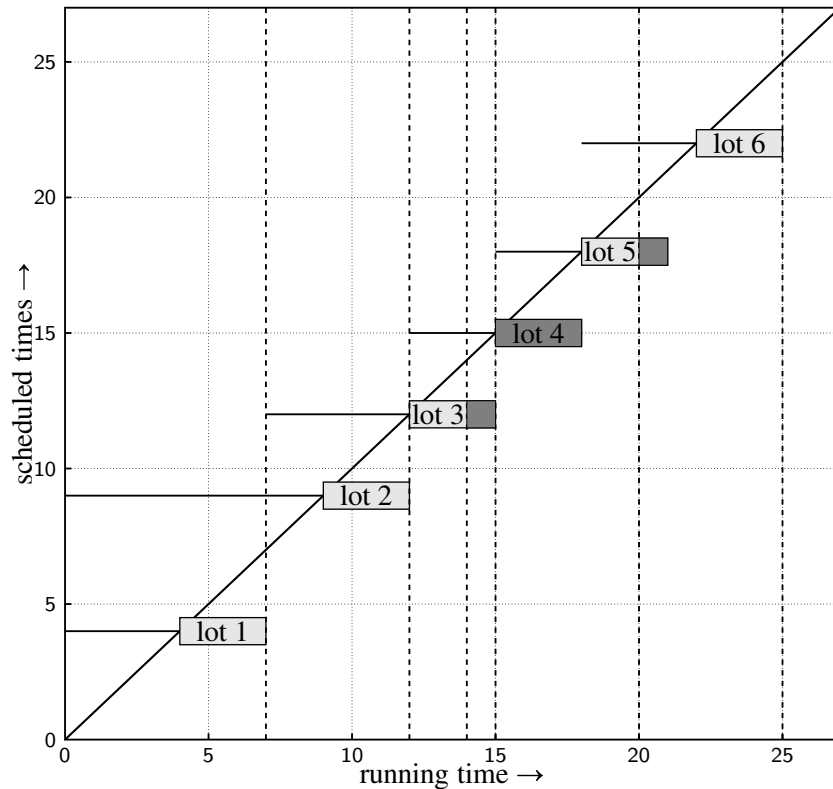


Figure 4.7: Implementation of feedback controller in a simulation study. Vertical dashed lines: due dates. Blocks: processing of lots on machine. Darkgray shading: earliness and tardiness. Horizontal lines: scheduled machine starts.

The simulation is started with an empty workstation: $x_1 = 0$, $x_2 = 0$. At $t = 0$, the control law is evaluated, resulting in an optimal schedule for manufacturing the first two lots. The schedule is computed in Region 2 of the MPLP problem solution. The arrival of a lot, authorization of the machine and actual start of the machine take place at the same instant. In Figure 4.7

the schedule is shown. The horizontal axis represents the actual (simulated) time, whereas the vertical axis represents the scheduled times of events. In the figure, only the scheduled starts of lots on the machines are plotted, the arrivals of lots have the same value. For example, at $t = 0$, two jobs are scheduled to start, the first at $t = 4$ and the second at $t = 9$. The vertical dashed lines show the due dates of the lots. The gray boxes show the processing of lots on the machine. If these boxes start on the diagonal line, it means that they started as scheduled. So the first lot was scheduled to start at $t = 4$ (from $t = 0$) and it indeed started at $t = 4$. All lots start at their scheduled times, so the controller produces feasible schedules. At $t = 7$, the first lot leaves the workstation, exactly on its due date. From that time, the due dates of the second and third lot are parameters for the control law. The second lot (that had already been scheduled from $t = 0$) remains scheduled to start at $t = 9$ and the third lot is scheduled to start at $t = 12$. The darkgray shaded parts of the boxes show the earliness or tardiness (in Figure 4.7 only tardiness) of lots. The cumulative penalty on the output error $\sum_{i=1}^6 |y_i - r_i|$ equals 5 for this simulation (the additional just-in-time control objective has a negligible contribution to the objective function).

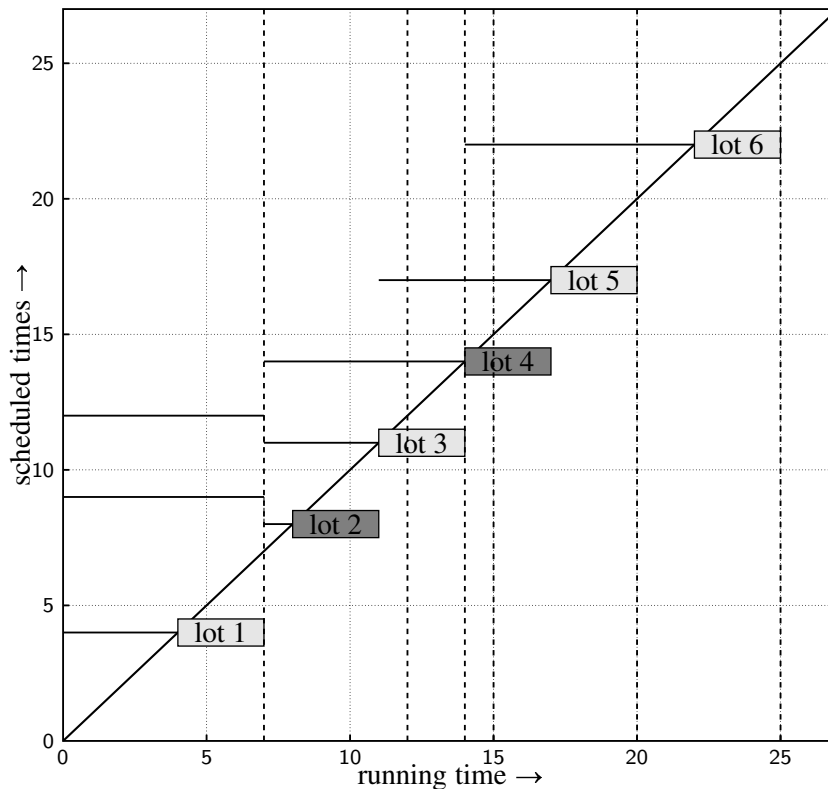


Figure 4.8: Implementation of feedback controller in a simulation with control horizon $N_c = 3$. Looking further ahead leads to early adaptation of the schedule to reduce penalties on earliness and tardiness.

Another deterministic simulation is performed to show the effect of larger control horizons. For the same workstation with the same due dates as in the first simulation, a feedback control law is computed with control horizon $N_c = 3$ lots. The controller looks further ahead and it is expected that possible large output errors can be avoided by early adjusting the manufacturing

schedule. From Table 4.1 it is known that this control horizon leads to an optimization problem consisting of 6 MPLP problems and that each MPLP problem has a solution with 10 regions. Implementation of this feedback controller in a deterministic simulation yields the results as shown in Figure 4.8. Initially, three lots are scheduled. After the first lot has been sent away, the reference vector \mathbf{r} is updated with the due date of the fourth lot. This due date is close to the due date of the third lot. Therefore, the schedule is adapted: the second and third lot are to be processed earlier, to keep the cumulative output error as small as possible. As a result, lot number 2 finishes 1 hour early and the fourth lot finishes 2 hours late. The third lot is scheduled to depart exactly on its due date. The total cumulative output error in the implementation with control horizon $N_c = 3$ equals 3, which is indeed less than the result with a control horizon of two lots. For these specific due dates, enlarging the control horizon even more does not result in lower output errors anymore (only considering the first six lots), but in general enlarging the control horizon leads to better overall solutions in a deterministic environment.

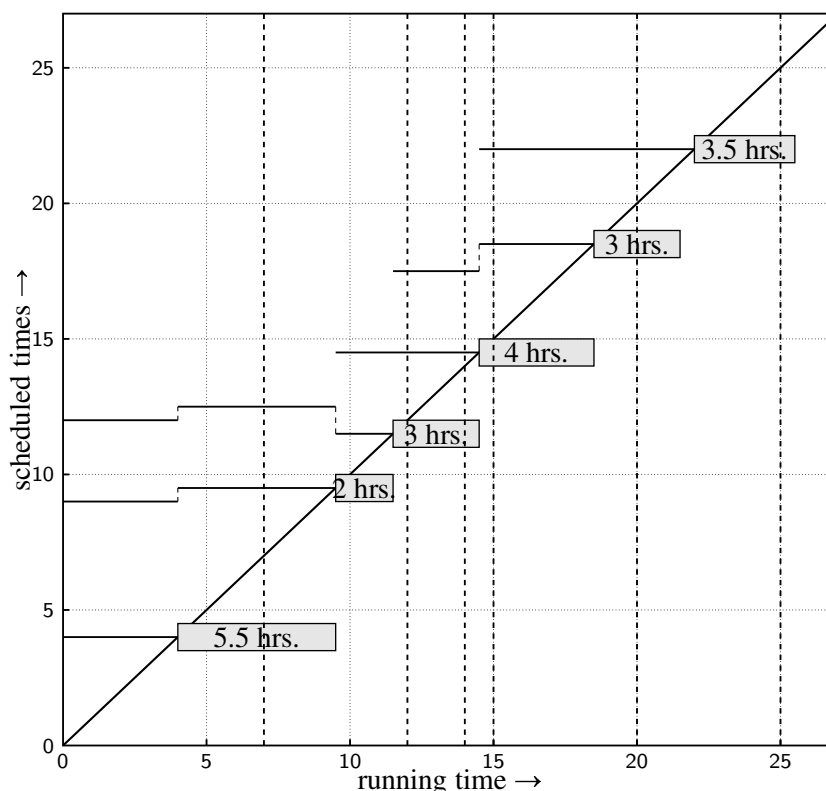


Figure 4.9: Simulation of feedback controller with control horizon $N_c = 3$ and varying process times.

Due to different sources of variability, process times are almost never constant and deterministic in practice. The feedback controllers that are obtained based on the constant process times can also be used in an environment where variation exists. Suppose that lots are scheduled based on the knowledge that the mean process time equals 3 hours and that the exact process time becomes known only when a lot is started on the machine (e.g. based on a manual inspection before processing starts). With this new information, the schedule can be recomputed by evaluating the feedback control law. An example of the schedule adaptation is presented in Fig-

ure 4.9, for control horizon $N_c = 3$. Inside the lot processing boxes, the process time is given. The graph shows that at the very moment it becomes clear that the first lot needs longer to process than average, the schedule for the second and third lot is adapted. The second lot takes shorter to process than planned, so the third lot is re-scheduled again. In this way, the schedule tries to keep up with the due dates as well as possible, always producing feasible schedules.

With three controller implementation simulations, the working of the controllers has been made insightful. The most important feature of the controllers is that optimal schedules are available continuously over time without re-optimization. Moreover, evaluating the feedback control law is only necessary when events occur in the manufacturing system, which also saves a lot of computation power. Looking further ahead by enlarging the control horizon results in schedules that give better performance in the sense of lower cumulative output errors. Finally, it was shown that also in an environment where variations occur, the controllers keep presenting feasible manufacturing schedules. In Section 4.3 the method for developing feedback controllers is used to determine a controller for a flow line of workstations. At the end of the chapter, a few remarks are made on the implementation of the feedback controller.

4.3 Flow line: interconnection and case study

In the previous sections, the receding horizon control method using the MPLP feedback was used to control a single workstation. Based on the due dates of products, the controller determined at which time instants lots needed to arrive at the workstation and when the machine should start processing a lot. The objective was to minimize the absolute output error, which is the absolute difference between due date and actual departure time of a finished lot. In this section, the method is propagated for control of a manufacturing flow line. The coupling between workstations appears to be very elegant and powerful. A generic approach is used to construct the MPLP problems: the method is suitable for an arbitrary number of workstations and arbitrary number of buffer places in each workstation. It is assumed that all workstations contain a single-lot machine. However, with slight adaptations, batch machines or conveyors can be incorporated in a similar way.

Consider the flow line as depicted in Figure 4.10, consisting of $N \in \mathbb{N}$ workstations. In this particular case study, $N = 3$. The workstations consist of a buffer B_n with finite capacity $C_n \in \mathbb{N}$ and a single-lot machine M_n with constant process time $d_n \in \mathbb{R}_+$, $n \in \{1, 2, 3\}$. Lots move from workstation 1 through 2 to 3 after which they are finished and leave the system. It is assumed that lots can always leave the flow line without delays.

The goal is to schedule jobs on the machines in such a way that the cumulative output error is minimized over a certain control horizon N_c (which is equal to the prediction horizon). This control horizon is the number of lots that is scheduled ahead. Within this goal, an additional objective exists: all process steps should be performed as late as possible (just in time) for economic reasons. This ensures that value is added as late as possible. In this case study, the

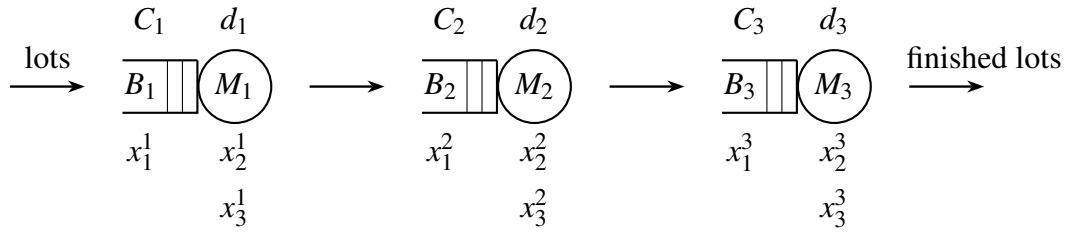


Figure 4.10: Flow line consisting of three workstations. Each workstation contains a buffer with finite capacity and a single-lot machine.

control horizon $N_c = 4$ lots. The objective function now becomes:

$$\min_{\mathbf{u}, \mathbf{v}} -\lambda \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix}^T + \sum_{i=1}^{N_c=4} |y_i - r_i| \quad (4.10)$$

in which λ is a weighting vector between the main and additional (just in time) control objective. Vectors \mathbf{u} , \mathbf{v} and \mathbf{w} are explained below. A little nuance has to be made for the secondary (just in time) control objective. If a job has been finished on one of the workstations which is not the most downstream workstation and the succeeding buffer has an empty space, the lot is immediately moved there. This introduces negative values in vector λ for the \mathbf{y}^n vectors, $1 < n < N$.

The state of this flow line is here characterized as:

$$\mathbf{x}(t) = \begin{bmatrix} x_1^1(t) & x_2^1(t) & x_3^1(t) & x_1^2(t) & x_2^2(t) & x_3^2(t) & x_1^3(t) & x_2^3(t) & x_3^3(t) \end{bmatrix}^T \quad (4.11)$$

where $x_1^n \in \mathbb{N}$ denotes the number of lots residing in the buffer of workstation n at time t . State elements $x_2^n \in \{0, 1\}$ represent the number of lots on the machine of workstation n and $x_3^n(t)$ is the remaining process time of the lots that is currently being processed. When no lot is being processed, the remaining process time is zero: $x_2^n(t) = 0 \implies x_3^n(t) = 0$, $n \in \{1, 2, 3\}$.

For each workstation n , vectors \mathbf{u}^n , \mathbf{v}^n , \mathbf{w}^n and \mathbf{y}^n are constructed. These vectors are the design variables of the optimization problem and contain the time instants at which lots arrive at the workstation, are authorized to be processed, are started being processed and leave the workstation respectively. The number of elements in these vectors depends on the distribution of lots that are already in the flow line. For example, if the most downstream workstation contains more lots than the control horizon, all other workstations do not have to process any lots anymore. For this reason, every possible combination of x_1^n and x_2^n over all workstations n results in a separate optimization problem, with different numbers of design variables.

The dimensions of the vectors \mathbf{u}^n , \mathbf{v}^n , \mathbf{w}^n and \mathbf{y}^n for all workstations $n \in \{1, 2, \dots, N\}$ are

computed as follows:

$$\begin{aligned}
 \dim \mathbf{y}^N &= N_c \\
 \dim \mathbf{y}^n &= \dim \mathbf{u}^{n+1} && \text{for } 1 \leq n < N \\
 \dim \mathbf{v}^n &= \max(\dim \mathbf{y}^n - x_2^n, 0) && \text{for } 1 \leq n \leq N \\
 \dim \mathbf{w}^n &= \dim \mathbf{v}^n && \text{for } 1 \leq n \leq N \\
 \dim \mathbf{u}^n &= \max(\dim \mathbf{w}^n - x_1^n, 0) && \text{for } 1 \leq n \leq N.
 \end{aligned} \tag{4.12}$$

For generating and solving the set of MPLP problems a MATLAB script was used, which has been included in Appendix B.1. An elaborate explanation of the script is given in the appendix, but in short the script works as follows. For arbitrary numbers of workstations and buffer places, a set of subproblems is generated which consists of all combinations of the buffer levels and presence of lots on machines. The number of subproblems is:

$$\text{number of MPLP problems to solve} = 2^N \times \prod_{n=1}^N (C_n + 1).$$

The parameter vector \mathbf{X} in the MPLP problems consists of the following elements:

$$\mathbf{X} = \begin{bmatrix} r_1 & r_2 & \dots & r_{N_c} & t & x_3^1 & x_3^2 & \dots & x_3^N \end{bmatrix}^T$$

which gives a parameter vector \mathbf{X} with eight elements for control horizon $N_c = 4$ and the number of workstations $N = 3$. Parameter t denotes the current time. For each MPLP problem, the constraint sets $\mathbf{G}\mathbf{U} \leq \mathbf{W} + \mathbf{E}\mathbf{X}$ and $\mathbf{A}\mathbf{X} \leq \mathbf{b}$ are constructed. Each workstation has similar constraints on \mathbf{u} , \mathbf{v} , \mathbf{w} and \mathbf{y} . These constraints are constructed for each workstation separately. Afterwards, the constraints are coupled in order to interconnect the workstations. This interconnection is facilitated as follows. Vector \mathbf{y} represents the time instants at which lots leave a workstation. Vector \mathbf{u} contains the time instants at which lots arrive at a workstation. In a flow line, the departure times of one workstation are the arrival times of the subsequent workstation. In (4.12) it was already shown that the dimension of these vectors are equal. Assuming zero transportation time between workstations, one may conclude:

$$\mathbf{y}^n = \mathbf{u}^{n+1}, \text{ for } 1 \leq n < N. \tag{4.13}$$

The \mathbf{G} matrix from the constraint set is now built up as shown schematically in Figure 4.11. The interconnection of workstations by (4.13) also reduces the number of design variables in \mathbf{U} . Figure 4.11 also shows the addition of the constraints of type (4.7) that are involved with the absolute value function on the output error in the objective function. The gray blocks in the figure typically consist of a collection of 0s, 1s and -1 s to form the precedence relations.

For the flow line consisting of three workstations, the set of MPLP problems are constructed and solved. The workstations have the following parameters:

$$C_1 = 2, \quad C_2 = 4, \quad C_3 = 2, \quad d_1 = 4, \quad d_2 = 3, \quad d_3 = 2.$$

The number of MPLP problems for this configuration is 360. Each MPLP problem has been solved with the MPT Toolbox for MATLAB (see [68]). The average number of regions within

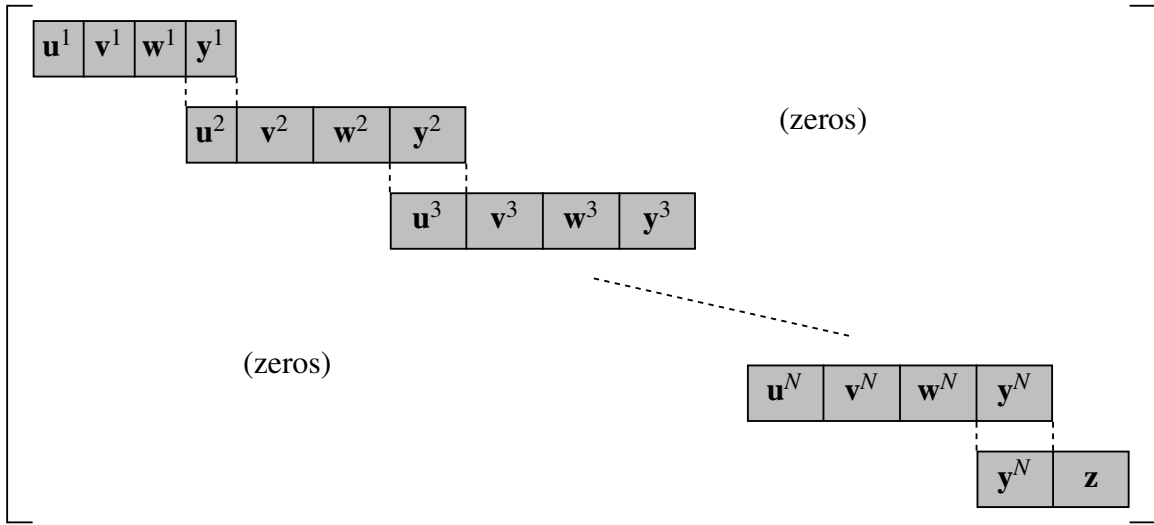


Figure 4.11: Construction of matrix \mathbf{G} with intrinsic interconnection of workstations.

the MPLP problems is 72. The maximum number of regions is 178 and the minimum is 24. Although the number of MPLP problems and regions looks high, the computations take only 1.5 hours on a PC (3GHz, 2GB memory). In addition, once all MPLP problems have been solved, no optimizations need to take place anymore in an implementation of the controller. The feedback law has been fully characterized now and can be used continuously in time as if drawing a record from a database.

The resulting feedback control law is implemented in both a deterministic and a stochastic simulation. In the deterministic simulation, process times of the machines are constant as indicated above, whereas in the stochastic simulation the process times of the machines are subject to variations due to a gamma distribution, with mean d_n as indicated above and a variance of 0.1. For the due dates, the following vector is used:

$$\mathbf{R} = \begin{bmatrix} 10 & 11 & 13 & 14 & 15 & 25 & 29 & 36 & 37 & 45 & \infty & \dots \end{bmatrix}.$$

Again, the artificial ∞ at the end of \mathbf{R} ensures that scheduling of the 8th, 9th and 10th lot is not influenced by future due dates and that \mathbf{r} can always have the right dimension. The following initial condition is used in both simulations:

- Workstation 1 has one lot in the buffer and an empty machine: $x_1^1 = 1, x_2^1 = 0, x_3^1 = 0$.
- Workstation 2 has two lots in the buffer. The machine processes a lot which is half-way processing: $x_1^2 = 2, x_2^2 = 1, x_3^2 = 1\frac{1}{2}$.
- Workstation 3 is empty: $x_1^3 = 0, x_2^3 = 0, x_3^3 = 0$.

The controller obtained by the MPLP problems is implemented in a discrete event simulation using MATLAB. The results of the deterministic simulation are shown in Figure 4.12, while the results from a stochastic simulation (gamma distributed with mean d_i and variance 0.1 for each workstation i) are presented in Figure 4.13. Instead of presenting graphs with scheduled times against simulation time, lot-time diagrams are shown in which the progress of lots through the

flow line is visualized. The due dates are indicated with the black arrowheads. Lightgray boxes show the presence of lots on a machine and darkgray boxes indicate that a lot resides in a buffer. The initial condition is clearly visible in the graphs for the first four lots. The cumulative output

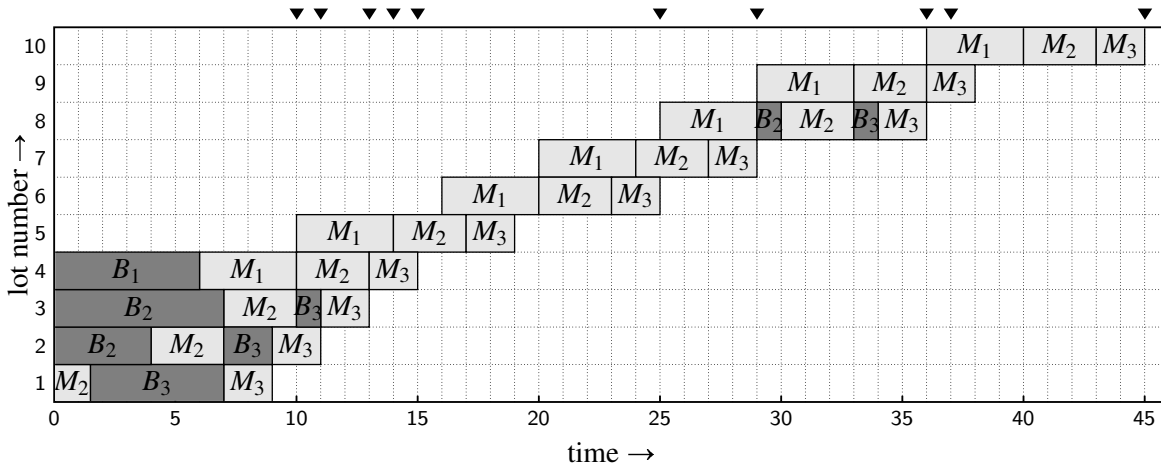


Figure 4.12: Lot-time diagram of deterministic simulation.

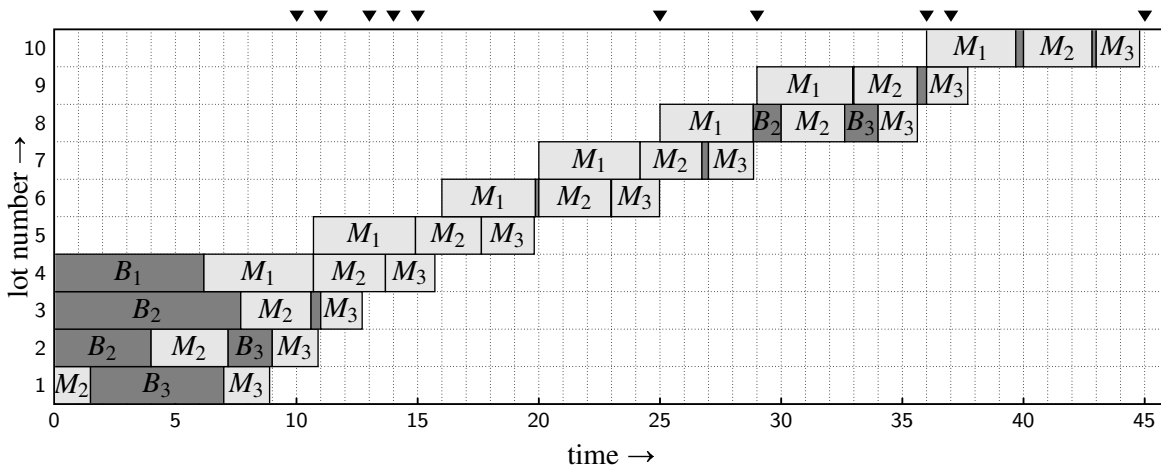


Figure 4.13: Lot-time diagram of stochastic simulation.

error $\sum_{i=1}^{10} |y_i - r_i|$ for the deterministic simulation is 7 hours (the additional just-in-time control goal has a negligible contribution to the objective function). Lot number 5 has a large output error. This is caused by the fact that at the moment lot 4 starts on machine 1, it is not known yet what the due date of lot 5 is. The control horizon equals 4 and no lot has left the flow line yet. By the time lot 1 leaves the flow line, lot 5 is scheduled and can only start on M_1 after lot 4 has been finished on M_1 . In case the control horizon N_c would have been 5, lot 4 would have been started earlier to keep the output error of lot 5 lower. Another interesting phenomenon happens with lot 8. It is started quite early to keep the output error of lot 9 small. As a result, lot 8 has to wait in buffers B_2 and B_3 . For the simulation of Figure 4.13, the cumulative output error

is 9.43 hours. However, this is just one simulation. In order to obtain a more insightful number for the cumulative output error in the stochastic environment, 100 simulations are carried out with the same initial conditions and due dates. The mean cumulative output error is 9.58 hours, with a standard deviation of 1.52 hours. The minimal cumulative output error in these 100 simulations is 6.20 hours (which is a better performance than the deterministic case) and the maximum realization is 13.96 hours. Only 2% of all stochastic simulations perform better than the deterministic simulation. This is not surprising, since only for lot numbers 1, 4, 5 and 9 an improvement is possible; all other lots had an error of zero in the deterministic case. In the stochastic case all lots have an output error, because due to the variations lots never leave the flow line exactly at their due dates. Any output error of a lot cannot be compensated: both earliness and tardiness are penalized equally.

The case study on the manufacturing flow line showed that the MPLP receding horizon control method can be implemented for a flow line. The coupling between workstations is a matter of equalizing departure and arrival times of the subsequent workstations. This case also showed that it is possible to generate all individual MPLP problems for a flow line of arbitrary length and control horizon in a generic way. This provides a powerful reference point for future research on the subject.

Remarks on the computation of MPLP problems and controller implementation

- The current computation of MPLP problems is for workstations with finite buffer capacity. In case of infinite capacity, an infinite number of problems is generated. This problem can be overcome by putting the buffer capacity to N_c . Because no more than N_c lots are optimized, the buffer is never exceeding this value. All buffer levels higher than N_c result in the same MPLP problem.
- It is possible that for different combinations of x_1^n and x_2^n the same MPLP problem emerges. The MATLAB script that computes and solves all problems does not check for duplicates. More extensive bookkeeping in this area can save a lot of time, since solving the MPLP problems takes far most of the computation time.
- The elements of weighting λ should be chosen *small*. When lots are scheduled over longer horizons or further away in time, the $\lambda \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix}$ sum can become too large with respect to the cumulative output error. This problem can be overcome by subtracting the current time t from the design variables in the objective function: $\lambda \begin{bmatrix} \mathbf{u} - t & \mathbf{v} - t & \mathbf{w} - t \end{bmatrix}$. This does not influence the solution of the optimization problem, since a constant is added to the objective function.
- In a situation where a workstation is empty ($x_1^n = 0$ and $x_2^n = 0$), after which a lot enters in that workstation which is to be processed immediately, two events occur: arrival of the lot ($x_1^n = 1$ and $x_2^n = 0$), followed by the start of the lot on the machine ($x_1^n = 0$ and $x_2^n = 1$). Since optimal schedules are available continuously in time, two updated schedules are determined at the same time instant. In the graphs, only the second schedule is shown, which corresponds to the right continuous variant of the schedule.

- If a lot resides on the machine ($x_2 = 1$) and its due date is still far away ($r - t > d$), then it is still possible to obtain an output error ($y - r$) of zero. This is possible when the lot does not leave the machine before the due date. Output time y is not the time of completion of a lot, but the time of departure of the lot. For the most downstream workstation, the completion time of a lot should be interpreted as the time of departure. This departure should not be delayed anymore. An example of this unwanted behavior is the eighth lot in Figure 4.13. That lot is completed earlier than expected, but the ninth lot does not start on M_3 immediately. The eighth lot is only sent away at $t = 36$. A way to overcome this problem is by adding equality constraint $y_1^N = t + x_3^N$, which makes sure that the departure time of the first lot that leaves the flow line equals the current time plus its remaining process time. However, the current implementation of the MPT toolbox for MATLAB does not support inclusion of equality constraints.
- Enlarging the control horizon or adding more workstations to the flow line results in larger MPLP problems. Unfortunately, the curse of dimensionality applies here: the size of the MPLP problems and the number of MPLP problems increase exponentially (For example, enlarging the control horizon to five lots yields MPLP problem solutions consisting of about 700 regions per MPLP subproblem; a control horizon of six lots yields in solutions consisting of about 3000 regions). High computation times are not a big problem, because the computation of the feedback law takes place off-line. However, for even larger MPLP problems, computer memory problems may occur.

4.4 Summary

In the previous chapters a state space representation for a manufacturing workstation has been introduced. The state vector consists of only three scalars per workstation and full state information can be measured instantaneously.

In this chapter the introduced state space representation has been used in a feedback control loop. A receding horizon control method was introduced to schedule the production steps of a certain number of jobs ahead. This number of jobs is called the *control horizon* N_c . Based on the current state of a workstation, the first N_c jobs are scheduled. However, different states lead to different schedules. For example, in case of an empty workstation all lots need to arrive at the workstation before they can be processed, while in case of a fully occupied workstation less or even no lots need to arrive anymore. Optimization of the production schedule is therefore performed for all separate states that lead to a different structure (in terms of design variables and constraints) of the optimization problem. In the optimization problems, the due dates of lots are parameters. In addition, the current time t and the remaining process times of lots that are currently being processed (part of the state) are parameters. This leads to a multi-parametric mathematical programming problem. In this chapter the objective function for the optimizations was taken linear, so the solutions of the complete set of multi-parametric linear programming (MPLP) problems represent the feedback control law. This control law is computed completely off-line. In an implementation, no optimizations need to take place anymore. At any point in

time, an optimal schedule is available as a kind of look-up table, based on the information that is known at that time. Even in a disturbed environment (where process times are random or with machine breakdowns), an optimal schedule is available based on the current state. With hindsight, earlier schedules might have been suboptimal, but they were optimal at the time of determining those schedules.

The feedback control method based on the receding horizon strategy has been implemented on a single workstation and on a flow line of workstations. In the flow line, individual workstations are interconnected elegantly by equalizing departure and arrival times of lots at succeeding workstations. A generic method was developed for generating and solving the MPLP problems for a flow line of arbitrary length, with arbitrary buffer capacities and control horizon. As an MPLP problem solver, the MPT toolbox for MATLAB [68] has been used.

Although this chapter only deals with workstations consisting of a buffer with finite storage capacity and a single-lot machine, the method that has been described is perfectly suitable for incorporating other manufacturing entities.

A switching server

In a lot of applications, servers have to share capacity over competing resources. Such servers can be found in manufacturing industry, food processing facilities and traffic flow networks. In general, these systems are discrete event systems. Switching between the competing resources might take time, the so-called setup time. In this chapter the scheduling problem of switching servers with non-zero setup times is regarded. The discrete event system behavior is modelled with hybrid fluid model dynamics, as introduced in Section 2.3.2. Buffer levels are modelled as continuous variables (as fluid levels) and switching between the product types is the discrete event part of the model. Since a server is assumed to process only one lot type at a time, steady state behavior results in a periodic *process cycle* rather than a fixed point. First, an optimal process cycle with respect to work in process levels is derived for a switching server processing two lot types. Next, a state feedback controller is proposed that brings the trajectory of a system to the derived optimal cycle. After the analysis for a single workstation has been completed, the analysis of switching server flow lines for two product types is treated in Chapter 6.

With respect to scheduling of a single switching server, in [97–99] a method is proposed to stabilize a server that serves n queues. A stable limit cycle solution is found with minimal cycle period. The control policy consists of fixed process periods for each lot type, actually a feed forward controller. A disadvantage of this approach is that it does not deal with disturbances and it does not reduce the number of lots in the system in case this number of lots is larger than necessary. In this chapter a feedback controller is used to deal with disturbances and with techniques as described in Matveev and Savkin [81], stability can be shown. Contrary to the control method in their work, the control strategy that is proposed in this thesis also reduces the number of lots in the system if possible.

Hybrid fluid model dynamics are also used by Boccadoro and Valigi [15, 16] and Del Gaudio et al. [34]. In these studies, scheduling problems for two competing queues with both infinite and finite buffer capacities are considered. However, those studies are based on completely symmetric systems, i.e. equal setup times, equal arrival rates and equal process rates for both lot types, whereas in this research these assumptions are dropped. Another assumption in [15, 16, 34] is that an optimal process cycle is never influenced by the maximum buffer capacities, whereas buffer constraints are embodied in the analyses in this chapter. On the other hand, they also study optimal transient behavior, which is not considered here. Other work by Martinelli and Valigi [80] focuses on the impact of finite buffer capacities on optimal scheduling of a single machine that processes two part types. However, this study does not include non-negligible setup times, whereas in this chapter setup times are involved. Khmelnitsky and Caramanis [64] use a different objective function: cumulative backlog costs (which can be translated to cumulative work in process levels) over a cyclic period, instead of mean wip level over a cyclic period. In addition, in [64] only systems with infinite buffer capacities are considered.

For queueing systems, policies to obtain stable process cycles are proposed by Boxma et al. [21, 22]. Some papers, e.g. [47, 48, 79], use a heavy traffic assumption on top of the proposed policy. In most work, first a control policy is proposed and then analysis and (sometimes) optimization is done within the given policy. Clearing policies (serve a queue until it is empty then switch to another queue) or threshold services (serve a queue until a value has been reached) are mostly considered in this area for both stochastic and deterministic environments. Eisenstein [41] assumes that a target cycle is given with idle times therein to be robust against disturbances. A recovering policy is proposed that uses this idle time to get back on the target cycle after a disruption. Gallego [46] also proposes a target cyclic scheduling recovering method, by means of introducing safety stocks and with infinite buffer capacities. In this chapter however, a different procedure is used: first the desired process cycle is derived (without necessary idle times and incorporating finite buffer capacity constraints) and then a control policy is looked for. A similar approach for traffic control is used by Haijema and Van der Wal [50], in which a control strategy is looked for to reach a desired system trajectory. The control policies that are proposed in this thesis follow from methods described in Lefeber and Rooda [73]. The derivations are not given here, the interested reader is referred to [73]. Related work for stochastic systems has been done by Hofri and Ross [60], but only equal maximum process rates and infinite buffer capacities are considered therein.

In Lan and Olsen [69] a fluid model is presented for a multi-product server which has to choose between the competing resources. A convex optimization problem is defined which results in a theoretical lower bound on the work in process levels in a deterministic environment. It is stated that the polling table resulting from the optimization is rare to find and in most cases unachievable. In this chapter the exact lower bound on the mean work in process level for a server with two lot types is computed, which in most cases differs from the lower bound found by Lan and Olsen [69].

A well known scheduling heuristic is based on the $c\mu$ -rule (see e.g. Buyukkoc et al. [23]) where

switching (without setup times) takes place according to a $c\mu$ index where c is a cost rate and μ the process rate. The lot type with highest index has priority. In this chapter, an optimal process cycle is derived, in which a *slow-mode* may occur (also referred to in literature as ‘idling’ [25] or ‘cruising’ [69]). In this mode, lots are processed at a lower rate than the maximum rate. If the slow-mode occurs, it takes place at the queue with the highest $c\lambda$ index, even if the $c\mu$ index of the other lot type is higher, as is shown in the case study in Section 5.5.

The remainder of this chapter is organized as follows. In Sections 5.1 and 5.2, the hybrid fluid model dynamics of a single switching server are described and an optimal process cycle with respect to time averaged weighted work in process levels of such a server is derived. In Section 5.3 a state feedback controller for a single switching server is proposed and convergence is proved analytically. The influence of finite buffer capacities on optimal process cycles and the feedback controllers is then studied in Section 5.4. For a case study with finite buffer capacities, the controller is implemented in simulations with the original hybrid fluid model, a deterministic discrete event model and a stochastic discrete event model. Results from the simulations show that the controller gives satisfactory results in an environment where disturbances (the stochastic behavior) occur. Sections 5.6 and 5.7 give remarks on deriving an optimal cycle based on pure discrete event dynamics and for situations with multiple product types (more than two types).

In Sections 5.1–5.7, it is assumed that products arrive at the workstation with a constant arrival rate. However, this is not always the case. Especially in manufacturing networks, in which the outflow of lots of one machine serves as inflow for the other machine, the arrival process of lots for the latter is not constant. In Section 5.8 again a switching server is considered that serves two lot types. The arrival pattern is now considered piecewise constant. Optimal process cycles (under certain assumptions) are derived and illustrated by means of a case study.

5.1 Characteristics and dynamics of two queues switching server

For switching servers, the modelling and control framework as laid down in the introductory chapter is passed through completely in this chapter. A model type is chosen to describe the dynamics of such a server. Based on the desired behavior of the closed-loop (controlled) system, a controller is proposed, which is implemented on the model and on other representations of switching servers. This section focuses on the modelling part. First the characteristics and dynamics of a switching server are presented. In Figure 5.1 a schematic representation of a *workstation* is given. In this chapter, a workstation consists of a number of parallel first-in-first-out (FIFO) lot-type specific buffers and a server. A *lot* is the to be processed entity. The number of lots in buffer i is denoted by x_i . The server can process only one type of lots at a time. Switching from lot type i to lot type j takes $\sigma_{ij} > 0$ time units. Without loss of generality, ‘hours’ is used as time unit. Furthermore, unless indicated otherwise, subscript i refers to the

lot type: $i \in \{1, 2, \dots, n\}$, with n the number of different lot types in the system. Lots of type i arrive at the workstation with a constant inter-arrival time of $1/\lambda_i$ hours. The server processes this type of lots with a constant process time of $1/\mu_i$ hours. In Figure 5.1, a workstation is shown that processes two lot types. This workstation is used in the analysis in this chapter: $n = 2$.

Since it is hard to describe the discrete event dynamics of the workstation with the purpose to perform analysis with it, a hybrid fluid model is used. Instead of using the interarrival *times* and process *times* of lots, the arrival *rate* $\lambda_i > 0$ and process *rate* $\mu_i > 0$ are used. A consequence is that the buffer levels $x_i \in \mathbb{R}_+$, with $\mathbb{R}_+ = [0, \infty)$, so they lose their natural-valued character.

The partial workload of a lot type on a server is denoted by ρ_i : $\rho_i = \lambda_i/\mu_i$. For reasons of stability, the total workload of a server must be strictly smaller than 1: $\sum_i \rho_i < 1$. This workload must not equal 1, since then no time is left to process the lots that arrive during a setup and buffer levels eventually explode.

Remark 5.1. Only switchover times σ_{ij} strictly greater than zero are considered. When the switchover times are zero and $\rho < 1$, the server is able to keep both queues empty by switching infinitely many times in finite time. This phenomenon is not studied in this thesis.

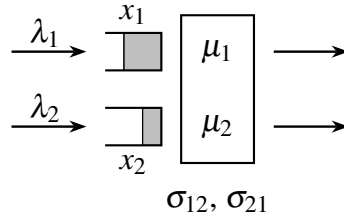


Figure 5.1: Switching server with two lot types.

The state of the system consists not only of the buffer levels x_1 and x_2 , but also of the *mode* m of the server and the remaining setup time x_0 . The mode is the lot type that the server is processing or setting up for. The remaining setup time is positive and decreases linearly while a setup is performed and zero otherwise. The state of the system at time t is given by:

$$\mathbf{x}(t) = \begin{bmatrix} x_0(t) & x_1(t) & x_2(t) & m(t) \end{bmatrix}^T \in [0, \max(\sigma_{12}, \sigma_{21})] \times \mathbb{R}_+^2 \times \{1, 2\}. \quad (5.1)$$

It is assumed that the process rate of the machine can be manipulated by a controller, as long as it does not exceed the maximum rate μ_i . Let manipulative input $u_i \leq \mu_i$ be the rate at which lots are processed. Another input of the system is the required activity u_0 of the server. Possible activities are:

- $u_0 = \textcircled{1}$: setup for type 1 lots
- $u_0 = \textcircled{1}$: process type 1 lots
- $u_0 = \textcircled{2}$: setup for type 2 lots
- $u_0 = \textcircled{2}$: process type 2 lots.

Note that ‘idling’ is not a separate activity, since it can be regarded as processing lots at rate zero, thus processing no lots. The complete input vector \mathbf{u} is given by:

$$\mathbf{u}(t) = \begin{bmatrix} u_0(t) & u_1(t) & u_2(t) \end{bmatrix}^T \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\} \times [0, \mu_1] \times [0, \mu_2]. \quad (5.2)$$

The inputs are bounded by constraints:

- lots cannot be processed while the server is busy with a setup;
- when the server is processing lots of a specific type, other lot types cannot be processed;
- if a buffer contains lots, the process rate for that type cannot exceed the maximum process rate;
- if a buffer is empty, the server can process lots with at most the arrival rate.

Formally, these input constraints can be written as:

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

Input u_0 causes state variables to jump between values. Not only mode m jumps between 1 and 2, but also the remaining setup time jumps from 0 to σ_{ij} when the mode switches:

$$x_0 := \sigma_{21}, \quad m := 1 \text{ for } u_0 = \mathbf{1} \text{ and } m = 2 \quad (5.3a)$$

$$x_0 := \sigma_{12}, \quad m := 2 \text{ for } u_0 = \mathbf{2} \text{ and } m = 1. \quad (5.3b)$$

Note that during a setup, an intervention may cause a switch to a different lot type. In that case the ongoing setup is interrupted and the new setup starts. This new setup takes its nominal duration σ_{ij} from the moment of interruption.

In addition to the discrete event dynamics, the following continuous dynamics apply:

$$\dot{x}_0(t) = \begin{cases} -1 & \text{for } u_0(t) \in \{\mathbf{1}, \mathbf{2}\} \\ 0 & \text{for } u_0(t) \in \{\mathbf{3}, \mathbf{4}\} \end{cases} \quad (5.4a)$$

$$\dot{x}_1(t) = \lambda_1 - u_1(t) \quad (5.4b)$$

$$\dot{x}_2(t) = \lambda_2 - u_2(t). \quad (5.4c)$$

The workstation with two product types has a fixed process cycle:

process type 1 \rightarrow setup to type 2 \rightarrow process type 2 \rightarrow setup to type 1 \rightarrow process type 1 $\rightarrow \dots$

The only freedom in this process cycle is the duration of each step in the cycle, in particular the steps in which lots are processed. The question now arises what is the *best* way to perform this cycle. This *best* adjective can be translated into an optimization problem. In the next section, optimal process cycles with respect to time averaged weighted work in process (wip) levels are investigated.

5.2 Optimal process cycle of two queues switching server

In this section, a switching server with two incoming lot types and constant arrival rates is considered, as presented in Section 5.1. An optimization problem is formulated for averaged weighted work in process levels. For strictly increasing cost functions, the general form of the process cycle is presented and for linear costs, an explicit solution is derived.

Starting from an arbitrary initial state, the goal is to achieve a switching policy that in the end minimizes the costs related to wip levels. The cost function J has the following form:

$$J = \limsup_{t \rightarrow \infty} \frac{1}{t} \int_0^t [g_1(x_1(s)) + g_2(x_2(s))] ds \quad (5.5)$$

with $g_i : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ functions that relate costs to wip levels (buffer levels). If assumed that higher buffer levels result in higher costs, i.e. if only strictly increasing functions g_i are taken into account, $g_i(x_i) > g_i(y_i)$ if $x_i > y_i$, then the following statements can be made.

Lemma 5.2. *When serving type i , optimal policies first serve at the highest currently possible rate, after which they might idle. This highest possible rate equals μ_i when the buffer contains lots, or arrival rate λ_i otherwise.*

Proof. Suppose that a policy is given for which after having completed the setup to type i , buffer i contains x_i^0 lots and at the end of mode i , buffer i contains x_i^f lots. Then one can consider the alternative policy which serves type i equally long in mode i and first serves at the highest possible rate, i.e. at the maximal processing rate as long as the buffer contains lots or at the arrival rate in case the buffer is empty. In the end, this alternative policy idles to make sure that at the end of mode i the buffer contains x_i^f lots (see Figure 5.2). Clearly, during mode i the number of lots in the buffer cannot decrease faster and in the end cannot increase faster than in this alternative strategy. Therefore, for the alternative policy at each time instant the wip level of type i is minimal. In particular, if the given policy is different, the wip level of type i is less at certain time instants. Since the time evolution of the other lot type(s) remains the same for both policies and g_i is strictly increasing, the costs are strictly less using the alternative strategy. \square

Note that Lemma 5.2 holds for arbitrary number of lot types $n \geq 2$.

Lemma 5.3. *Optimal policies do not idle, i.e. process lots at rate zero.*

Proof. Suppose that an optimal policy would idle in mode i . Given the result in Lemma 5.2 this is at the end of mode i . Furthermore, suppose that for this policy service in the next mode stops at time t_f . Consider an alternative policy which does not idle in mode i , but switches immediately to the next mode and stays in this mode until t_f , serving an equal amount of lots as the supposed optimal strategy. For this alternative strategy the evolution of the buffer contents of type i does not change. Also the amount of lots remaining in the buffer of the next mode at t_f is equal. However, (some of) the lots that are served in the next mode are served sooner, due

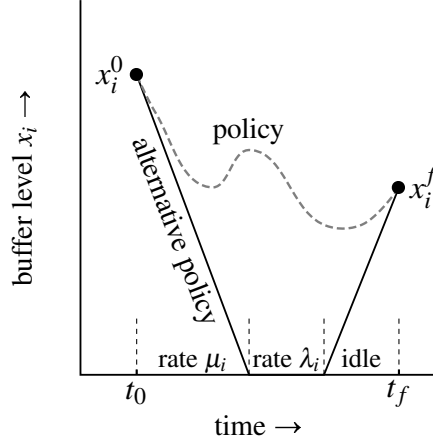


Figure 5.2: Graphical representation of Lemma 5.2.

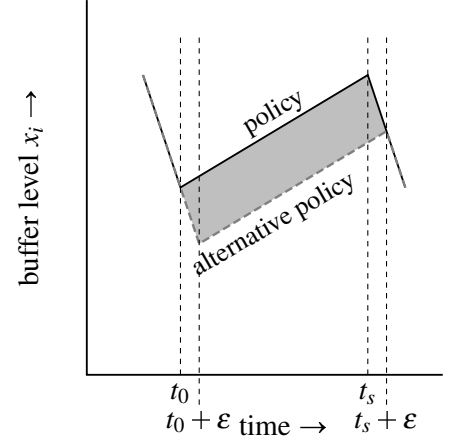


Figure 5.3: Graphical representation of Lemma 5.4.

to the constant arrival rate and non-negative setup time. Therefore, for some period of time the number of lots in the buffer of the next mode is strictly less in the alternative strategy. Since g is strictly increasing, the costs up to time t_f are strictly less for the alternative strategy. Therefore, an optimal policy does not idle. \square

Note that Lemma 5.3 also holds for arbitrary number of lot types $n \geq 2$.

In order to derive optimal steady state behavior for a server serving two types of lots, first properties of optimal behavior for one type only without considering the other type are investigated. Note that in general it might not be possible to have both types behave optimally, but at least this provides us with a lower bound for optimal system behavior. Next, it is shown that both types can behave optimally, i.e. that this lower bound can be achieved.

Lemma 5.4. *For optimal steady state behavior of type i , buffer i is always emptied.*

Proof. Consider a policy where at time $t = t_0$ the system stops serving buffer i which is not yet empty, and that at $t = t_s$ type i is served again. Consider an alternative policy where the system stops serving buffer i at $t = t_0 + \epsilon$ (with $0 < \epsilon \leq x_i(t_0)/(\mu_i - \lambda_i)$), mimics the given policy with a time delay of ϵ , and then starts serving type i at $t = t_s + \epsilon$. Mimicking is possible, since delaying the start of processing lots results in an increase of the buffer level due to the constant arrival rate. Since some lots of type i are served sooner in the alternative strategy, the number of lots in buffer i is strictly less at each time instant for the alternative strategy. Since g_i is strictly increasing, the contribution to the costs by type i is strictly less for the alternative strategy (the gray shaded area represents the difference). \square

Remark 5.5. Note that the contribution to the costs by the other type(s) might be higher for the proposed alternative strategy, but as mentioned earlier currently the focus is only on optimizing a single lot type without taking into account the other lot type(s).

Lemma 5.6. *For optimal steady state behavior of type i , vacations always take equally long, i.e. setups and serving the other type(s) between two successive services of type i always takes equally long.*

Proof. Assume that for an optimal steady state behavior of type i , service for type i stops at time $t = 0$. From Lemma 5.4 it is known that buffer i is empty then. Since lots of type i arrive at a constant rate, the contents of buffer i grow linearly, until type i is served again. Then buffer i is emptied, the contents of buffer i grow linearly until type i is served again, and buffer i is emptied again, see also Figure 5.4. Now assume that two successive periods of not serving

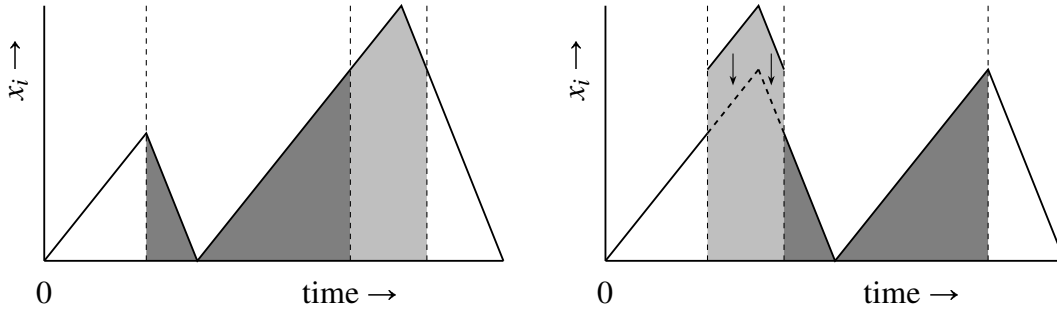


Figure 5.4: Times between successive services are equally long.

type i are not equally long. Without loss of generality, assume that the first period is shorter than the second. The resulting evolution of the buffer contents of buffer i is depicted in the left hand side of Figure 5.4. In the right hand side of Figure 5.4 the resulting evolution of the buffer contents of buffer i is depicted in case both periods of not serving type i are equally long. The graph of the buffer contents of type i in the left hand side figure can be divided into four parts. The first part consists of the period that type i is not served. The second part consists of the period that type i is first served and then not served, until the buffer reaches the value it would have had if both periods of not serving type i would have been equally long. The third part consists of the period that the buffer contents exceed this value and return at this value. And, finally, the fourth part consists of the remaining period. The different parts can be distinguished based on the fill color in Figure 5.4.

By interchanging the second (dark gray) and third (light gray) part of this graph, the right hand side of Figure 5.4 can be obtained. Next, it can be seen that the alternative strategy (not serving type i for equally long duration) results in strictly less lots of type i for a certain amount of time. Since g_i is strictly increasing, the resulting costs for this alternative strategy are strictly less for type i . \square

Lemmas 5.4 and 5.6 are valid if a lot type is optimized in isolation. Since multiple lot types occur in switching servers, optimizing each lot type separately might not lead to a feasible solution. However, in case of $n = 2$ lot types, this is not an issue: During the time span a lot type is not processed, processing the other type can take place. Because the vacation times are equally long, the process intervals are also equally long: vacation time of one type is processing time of

the other type. Setups count on both sides in the vacation time. So in general (given (5.5)) for two types, the shape of an optimal process cycle looks as in Figure 5.5, which always fulfills the previous lemmas and leads to the following corollary:

Corollary 5.7. *An optimal steady state process cycle for a switching server with two different lot types and strictly increasing costs on wip levels defined as:*

$$J = \frac{1}{T} \int_0^T [g_1(x_1(s)) + g_2(x_2(s))] ds. \quad (5.6)$$

needs to have the shape as shown in Figure 5.5. In the left-hand side graph of this figure, the process cycle is plotted in the (x_1, x_2) -plane. The right-hand side graphs show the buffer levels over time, with the slopes of the lines annotated to them. The cycle has period 1 with period length T .

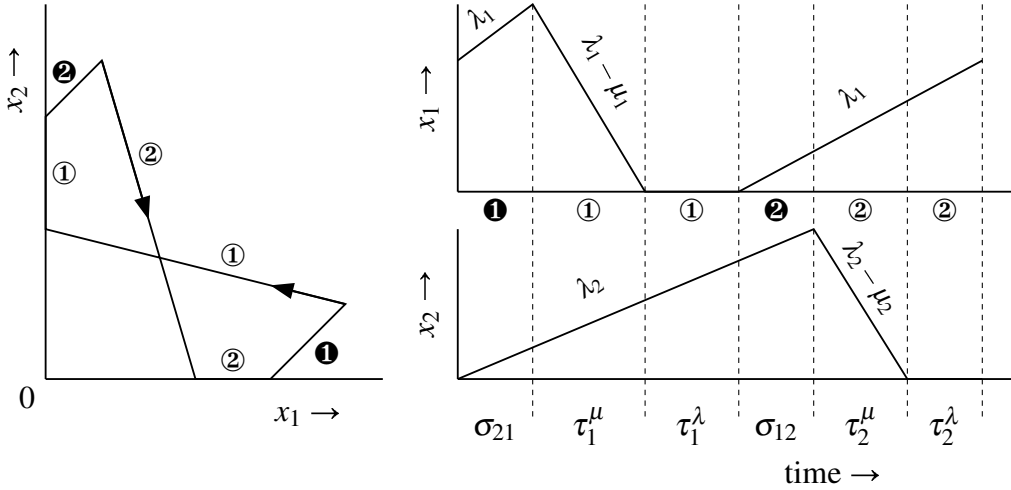


Figure 5.5: General form of an optimal process cycle for a switching server processing two lot types. Left: Periodic orbit. Right: buffer levels over time, with slopes of the lines.

Remark 5.8. When setup costs are involved, i.e. setups not only take time but also contribute to J , optimal cycles still have the shape of Figure 5.5, provided the number of setups to each type within a process cycle adds linearly to the objective function. In case of two lot types, only a constant is added to J , since the number of setups to each type is fixed: every lot type is processed only once in an optimal process cycle. Lan and Olsen [69] take setup costs into account in their analysis in addition to setup times.

Remark 5.9. When the server processes lots at the arrival rate, thus keeping the respective buffer empty, it is in *slow-mode*. An example of a slow-mode is visible in Figure 5.5, where the orbit stays on the vertical axis for a while. One might argue that this slow-mode does not occur for an optimal policy since it means that capacity is lost due to processing at lower rates than the maximum rate. However, introducing a slow-mode implies that the duration of the process cycle becomes longer, which in turn implies that the system switches less. Possibly

a trade-off exists between losing capacity due to processing at lower rates and losing capacity due to switching more often. The outcome of this trade-off depends on the specific choice for the functions g_i .

In the remainder of this chapter, linear costs on wip levels are assumed, resulting in the following costs for optimal process cycles:

$$J = \frac{1}{T} \int_0^T [c_1 x_1(s) + c_2 x_2(s)] ds \quad (5.7)$$

where c_1 and c_2 are constant weighting factors for lot type 1 and 2 respectively. In addition, without loss of generality, assume $c_1 \lambda_1 \geq c_2 \lambda_2$. Moreover, define $\sigma = \sigma_{21} + \sigma_{12}$. Corollary 5.7 describes a class of trajectories to which an optimal trajectory belongs (not all trajectories with the shape of Figure 5.5 result in optimal wip levels). From this class of trajectories, the optimal one obeys the following theorem.

Theorem 5.10. *Optimal process cycles with respect to time averaged weighted wip levels for a switching server with two different lot types and linear costs on wip levels (5.7), have a slow-mode (see Remark 5.9) for at most one lot type (type 1). During the slow-mode, lots are processed at their arrival rate, keeping the respective buffer empty. The slow-mode occurs if and only if $c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2)(1 - \rho_2) < 0$.*

Proof. See Appendix A.2. □

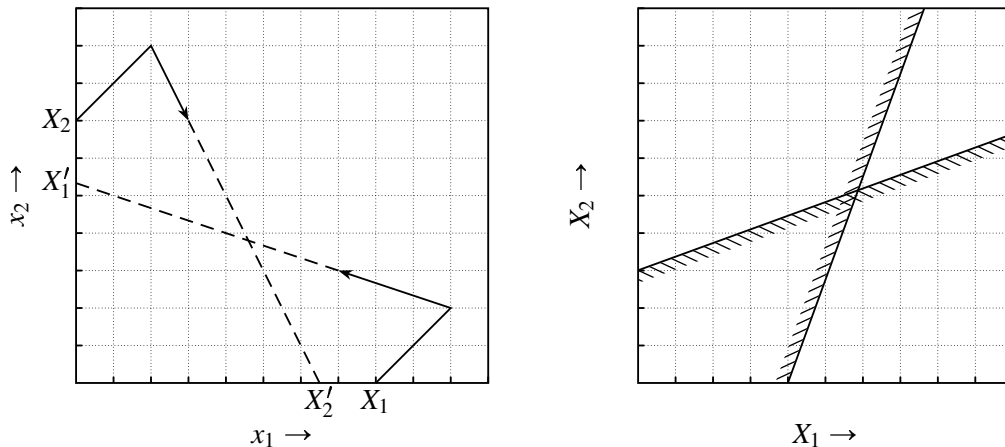


Figure 5.6: Construction of process cycle with minimal maximum buffer lengths.

In addition to the optimal process cycle, another process cycle is of particular interest: the cycle with a minimal period and with minimal maximum buffer lengths. This process cycle can be obtained by solving a mathematical program: minimize the maximum buffer levels such that the cycle remains feasible. In Figure 5.6 the construction of the cycle is sketched. In the left hand side, a cycle similar to Figure 5.5 is given. The solid lined arrows indicate parts of the

process cycle that are fixed: the setup times and the direction of the process lines. Capitals X_i denote the buffer levels of type i lots at which a setup starts to mode i . It can be seen that if a setup starts at X_i , eventually the arrow hits the other axis at X'_i . The cycle is feasible if and only if $X'_1 \leq X_2$ and $X'_2 \leq X_1$. The coordinates of X'_i can be determined:

$$X'_1 = \lambda_1 \left(\sigma_{12} + \frac{X_1 + \lambda_2 \sigma_{12}}{\mu_2 - \lambda_2} \right) \quad (5.8a)$$

$$X'_2 = \lambda_2 \left(\sigma_{21} + \frac{X_2 + \lambda_1 \sigma_{21}}{\mu_1 - \lambda_1} \right). \quad (5.8b)$$

Constraint lines $X'_1 \leq X_2$ and $X'_2 \leq X_1$ are shown (sketch) in the right hand side of Figure 5.6. The slopes of the lines are always positive, since $\lambda_i > 0$ and $\mu_i > \lambda_i$. Therefore, the minimal maximum buffer lengths are at the intersection of the constraint lines. The graph also shows that this is the optimal value for both X_1 and X_2 . The resulting periodic orbit of this cycle with minimal maximum buffer lengths is shown in the left hand side graph of Figure 5.7. As can be seen, no slow-modes occur: after having processed a specific lot type until its buffer is empty, the system immediately switches to the other lot type. The buffer values at specific corner points of this (*pure*) *bow tie* curve are indicated with an asterisk (*) and hats (^) are used to indicate maximum buffer levels for a given cycle. The coordinates of interest are:

$$\text{After ①: } (0, x_2^*) \quad \text{with } x_2^* = \lambda_2 \left(\sigma_{21} + \frac{\sigma \rho_1}{1 - \rho_1 - \rho_2} \right) \quad (5.9a)$$

$$\text{After ②: } (\lambda_1 \sigma_{12}, \hat{x}_2^*) \quad \text{with } \hat{x}_2^* = \lambda_2 \sigma \left(\frac{1 - \rho_2}{1 - \rho_1 - \rho_2} \right) \quad (5.9b)$$

$$\text{After ③: } (x_1^*, 0) \quad \text{with } x_1^* = \lambda_1 \left(\sigma_{12} + \frac{\sigma \rho_2}{1 - \rho_1 - \rho_2} \right) \quad (5.9c)$$

$$\text{After ④: } (\hat{x}_1^*, \lambda_2 \sigma_{21}) \quad \text{with } \hat{x}_1^* = \lambda_1 \sigma \left(\frac{1 - \rho_1}{1 - \rho_1 - \rho_2} \right). \quad (5.9d)$$

These coordinates are in accordance with the values in the examples of [15, 34] for the symmetric queues. In the right hand side graph of Figure 5.7 the periodic orbit of an optimal cycle has been shown for the case where a slow-mode occurs. This orbit is referred to as the *truncated bow tie* curve. The coordinates with flat (^b) and sharp ([#]) symbols denote the points where the slow-mode starts and ends respectively.

Remark 5.11. The lines corresponding to similar phases in the process cycle have the same slopes in the left hand side and right hand side graph. Notice that therefore $x_2^* \leq x_2^b$ and $x_1^* \leq x_1^{\#}$ (these hold with equality iff no slow-mode occurs). Although it looks counterintuitive, the right hand side graph has lower mean weighted wip levels than the left hand side graph, in case a slow-mode occurs according to the condition stated in Theorem 5.10. The counterintuitiveness is due to the fact that sense of time is lost in these graphs: the left hand side graph has a shorter period than the right hand side graph. The system spends a relatively large amount of time on the vertical axis ($x_1 = 0$) of the truncated bow tie, compared to the pure bow tie.

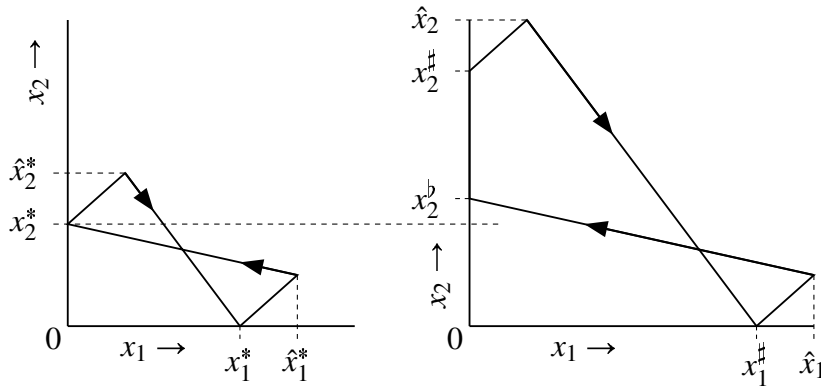


Figure 5.7: Pure bow tie curve and truncated bow tie curve.

The coordinates of interest for the truncated bow tie curve are:

$$\text{After ① at } \mu_1: \quad (0, x_2^b) \quad \text{with } x_2^b = \lambda_2 \left(\sigma_{21} + \frac{\sigma \rho_1 (1 + \alpha_1 \rho_2)}{1 - \rho_1 - \rho_2} \right) \quad (5.10a)$$

$$\text{After ① at } \lambda_1: \quad (0, x_2^\#) \quad \text{with } x_2^\# = \lambda_2 \left(\sigma_{21} + \frac{\sigma (\alpha_1 (1 - \rho_1) (1 - \rho_2) + \rho_1)}{1 - \rho_1 - \rho_2} \right) \quad (5.10b)$$

$$\text{After ②:} \quad (\lambda_1 \sigma_{12}, \hat{x}_2) \quad \text{with } \hat{x}_2 = \lambda_2 \sigma \left(\frac{(1 - \rho_2) (1 + \alpha_1 (1 - \rho_1))}{1 - \rho_1 - \rho_2} \right) \quad (5.10c)$$

$$\text{After ②:} \quad (x_1^\#, 0) \quad \text{with } x_1^\# = \lambda_1 \left(\sigma_{12} + \frac{\sigma \rho_2 (1 + \alpha_1 (1 - \rho_1))}{1 - \rho_1 - \rho_2} \right) \quad (5.10d)$$

$$\text{After ③:} \quad (\hat{x}_1, \lambda_2 \sigma_{21}) \quad \text{with } \hat{x}_1 = \lambda_1 \sigma \left(\frac{(1 + \alpha_1 \rho_2) (1 - \rho_1)}{1 - \rho_1 - \rho_2} \right). \quad (5.10e)$$

The value of α_1 can be computed using expressions in Appendix A.2 and is proportional to the duration of the slow-mode. Note that if $\alpha_1 = 0$, the pure bow tie trajectory actually is the optimal curve: $x_2^\# = x_2^b = x_2^*$ and consequently $x_1^\# = x_1^*$.

Example 5.12. Consider two uni-directional roads crossing each other (see Figure 5.8). Cars approach the intersection with constant speed and constant distance between each other. Traffic lights have been placed for safe traffic flow. Between successive passages of cars, all traffic lights are red for 10 seconds ($\sigma_{12} = \sigma_{21} = 10$ seconds) to clear the intersection completely. Acceleration and deceleration effects are neglected. Cars from the west arrive at a rate of 40 cars/minute, while cars from the south arrive at a rate of 10 cars/minute.

The intersection maximum capacity is 100 cars/minute for both car flows. Do the traffic lights have to turn red immediately after the queues are gone? Assume that both queues are of equal importance: $c_1 = c_2 = 1$. Indeed $c_1 \lambda_1 \geq c_2 \lambda_2$, so the expressions and conditions from this chapter can be used without re-labelling the car flows. This traffic lights problem is very much alike the switching server problem that was treated in this section. According to Theorem 5.10 the optimal traffic light cycle contains a slow-mode. This slow-mode has a duration of 10.4 seconds. This means that when the western queue has disappeared, the traffic light should remain green until the southern queue has reached a certain length $x_2^\#$, which according to (5.10) is approximately 6 cars. The average waiting time for a car in a queue is minimal then (≈ 12.8

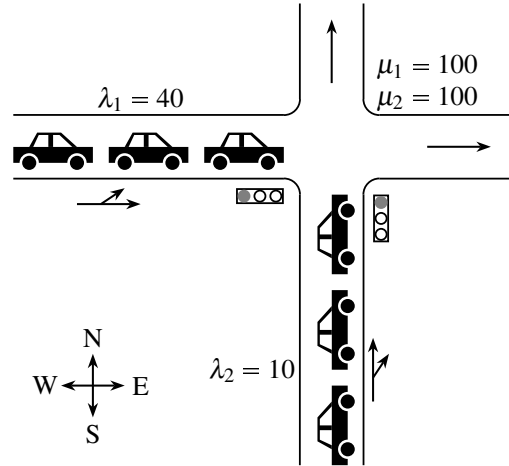


Figure 5.8: Cars approaching an intersection: what are the best switch-over times for the traffic lights?

seconds), according to Little's law, see [75] and the introductory chapter of this thesis. If the slow-mode would have been omitted, which results in the pure bow tie curve (clearing policy), the average waiting time would be 13.2 seconds. Introducing the slow-mode thus results in a (small) reduction of the average waiting times.

In this section, an optimal process cycle for a switching server with two lot types and setup times has completely been determined. Additionally, the process cycle with minimal period and minimal extreme buffer lengths has been characterized. In the next section, a state feedback controller is proposed that steers the trajectory of the system to this desired optimal trajectory, from any arbitrary starting point (initial state).

5.3 State feedback control of switching server

During the start-up of a factory, or due to disturbances, the trajectory of a workstation is in general not on the desired curve. Therefore, a controller is needed to steer the process trajectory to the desired curve. A possible controller can be derived using the theory presented in [73].

Proposition 5.13. *The following informal state feedback control law brings the system of Figure 5.1, with its dynamics described by (5.3) and (5.4), to the desired optimal periodic process cycle with respect to minimal time averaged wip level.*

- *Mode 1: Process lots of type 1 at rate μ_1 as long as $x_1 > 0$, then go to Mode 2.*
- *Mode 2: Process lots of type 1 at rate λ_1 as long as $x_2 < x_2^\sharp$, then go to Mode 3.*
- *Mode 3: Perform a setup to type 2 lots, after σ_{12} go to Mode 4.*
- *Mode 4: Process lots of type 2 at rate μ_2 as long as $x_2 > 0$, then go to Mode 5.*
- *Mode 5: Process lots of type 2 at rate λ_2 as long as $x_1 < x_1^\sharp$, then go to Mode 6.*
- *Mode 6: Perform a setup to type 1 lots, after σ_{21} go to Mode 1.*

A formal representation of this feedback control law is given in Appendix A.3.

Dependent on the state of the system, the controller is in one of the six controller modes initially. This follows trivially from the controller mode descriptions. Mode 2 and Mode 5 of this controller might have a duration of zero, immediately proceeding to Mode 3 and Mode 6 respectively. The controller modes are not to be confused with mode m of the server.

Proof. See Appendix A.3. □

The controller presented in Proposition 5.13 is a double threshold policy, i.e. exhaustive processing and switch only whenever the other buffer level has reached some value. Hofri and Ross [60] use a similar double threshold policy, though implemented in a stochastic environment. The difference between their result and the controller presented here is that their analysis has only been done for workstations with equal (stochastic) process rates for both lot types. Moreover, in [60], only infinite buffer capacities have been taken into account. In the next section, the optimal process cycle analysis and feedback controller are extended to situations with finite buffer capacities.

5.4 Finite buffer capacities: effects on optimal process cycle and controller

In this section, an optimal process cycle for a switching server with two lot types and setup times is determined for a workstation with finite buffer capacity. Buffer level limitations not only occur in physical systems, like manufacturing systems or food processing industry, but also in data flow or communication systems, where data storage is most often limited. In fact, storage capacity is always limited, but in special cases it can be regarded as virtually unlimited and then the analyses of Sections 5.2 and 5.3 are applicable. The results from these sections serve as a starting point to derive an optimal process cycle in the finite buffer case.

When the buffer level constraints are ‘large enough’ (which is specified below), the optimal process cycle as derived in the previous section does not change. However, when an optimal process cycle crosses a buffer level constraint, the process cycle is not feasible anymore and needs to be adjusted, if possible. In the previous section, it was derived that the process cycle with minimal maximum buffer lengths was the pure bow tie. If the pure bow tie curve violates a buffer level constraint, no feasible process cycle is possible anymore. In the remainder of this section, optimal process cycles are derived for situations where either the unconstrained optimal process cycle is not restricted by the maximum buffer capacities, or an adapted process cycle is possible, then serving as new optimal process cycle. For this constrained situation, again a feedback controller is proposed which is analytically proven to converge to the new optimal process cycle.

The maximum number of lots in the buffers is denoted by x_1^{\max} and x_2^{\max} . In Section 5.2, the pure bow tie curve was defined as the periodic orbit with minimal period and with minimal

maximum buffer levels: \hat{x}_1^* and \hat{x}_2^* . As mentioned, if the buffer capacity is less than these values, no periodic orbit can be found, so the first conditions on the new optimal process cycle are:

$$x_1^{\max} \geq \hat{x}_1^*; \quad x_2^{\max} \geq \hat{x}_2^* \quad (5.11)$$

with \hat{x}_1^* and \hat{x}_2^* as in (5.9).

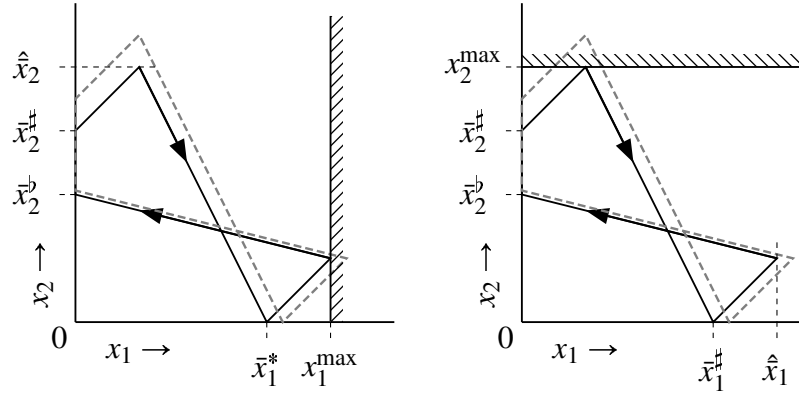


Figure 5.9: New optimal cycle, due to buffer capacity constraints. In dashed gray: original unconstrained optimal cycle.

Assume that the maximum buffer capacities cross an unconstrained process cycle: $x_1^{\max} < \hat{x}_1$ or $x_2^{\max} < \hat{x}_2$ (recall that the hats $\hat{\cdot}$ denote the maximum buffer lengths of a process cycle). The unconstrained optimal process cycle now has to be adjusted, to prevent violation of the buffer constraint. In Figure 5.9, buffer constraints have been drawn for type 1 lots (left hand side graph) and type 2 lots (right hand side). The coordinates of the optimal process cycle with buffer level constraints are denoted with bars ($\bar{\cdot}$). The original (unconstrained) process cycle is shown in dashed gray. Examining the geometrical implications of buffer level constraints on the optimal periodic orbit, the following expressions can easily be derived for the coordinates of the new optimal process cycle:

$$\begin{aligned} \bar{x}_1^\# &= \min \left(x_1^\#, \quad x_1^{\max} - \lambda_1 \sigma_{21}, \quad \lambda_1 \left(\sigma_{12} + \frac{x_2^{\max}}{\mu_2 - \lambda_2} \right) \right) \\ \hat{x}_1 &= \min \left(\hat{x}_1, \quad x_1^{\max}, \quad \lambda_1 \left(\sigma + \frac{x_2^{\max}}{\mu_2 - \lambda_2} \right) \right) \\ \bar{x}_2^b &= \min \left(x_2^b, \quad \lambda_2 \left(\sigma_{21} + \frac{x_1^{\max}}{\mu_1 - \lambda_1} \right), \quad \lambda_2 \left(\sigma_{21} + \frac{\lambda_1}{\mu_1 - \lambda_1} \left(\sigma + \frac{x_2^{\max}}{\mu_2 - \lambda_2} \right) \right) \right) \\ \bar{x}_2^\# &= \min \left(x_2^\#, \quad \frac{\mu_2 - \lambda_2}{\lambda_1} (x_1^{\max} - \lambda_1 \sigma) - \lambda_2 \sigma_{12}, \quad x_2^{\max} - \lambda_2 \sigma_{12} \right) \\ \hat{x}_2 &= \min \left(\hat{x}_2, \quad \frac{\mu_2 - \lambda_2}{\lambda_1} (x_1^{\max} - \lambda_1 \sigma), \quad x_2^{\max} \right). \end{aligned} \quad (5.12)$$

Note that if no slow-mode occurs in the unconstrained optimal process cycle, $\bar{x}_2^b = \bar{x}_2^\# = x_2^b = x_2^\# = x_2^*$, $\hat{x}_2 = \hat{x}_2 = \hat{x}_2^*$, $\bar{x}_1^\# = x_1^\# = x_1^*$ and $\hat{x}_1 = \hat{x}_1 = \hat{x}_1^*$. Another observation is that if the optimal process cycle with slow-mode changes due to the buffer capacity constraints, the duration of the slow-mode is shortened. In the minimum expressions in (5.12), the first term is ‘active’ (the smallest term) if the buffer level constraints are not restrictive for both lot types. The second

term is ‘active’ if the constraint on type 1 lots is most restrictive and the third term is ‘active’ if the buffer level constraint for type 2 lots is most restrictive.

Before the state feedback controller is adjusted to accommodate the buffer level constraints, first a closer look at the (x_1, x_2) -plane is taken. This plane can be divided into regions from which it is either possible or impossible to reach the desired steady state process cycle.

Lemma 5.14. *Regardless of the state feedback policy, the (x_1, x_2) -plane can be divided into regions from which it is impossible to reach the steady state process cycle when in a certain mode, see Figure 5.10. The regions marked with ①† and ②† indicate that if the trajectory enters that region processing type 1 or type 2 lots respectively, the trajectory eventually becomes infeasible (i.e. a buffer constraint is violated). If the trajectory is on the bow tie curve shifted into the upper right corner of the (x_1, x_2) -plane, the trajectory stays there.*

Proof. It is easy to see that once the trajectory has crossed the dashed lines in Figure 5.10 and enters the areas with a † next to an action ① or ②, a setup to the other lot type causes the buffer constraint to be violated. Moreover, if the setup is not initiated, the trajectory also crosses the borders of the feasible domain. For the regions in or above the upper-right bow tie, the proof is included in the feedback control law proof of Proposition 5.15 in Appendix A.4. \square

Optimal process cycles for a switching server with two product types and finite buffer capacity have completely been defined. In addition, insight has been obtained about feasible and infeasible areas in the (x_1, x_2) -plane. By using the theory presented in [73] the feedback controller as presented in Proposition 5.13 can be modified by taking into account the buffer constraints.

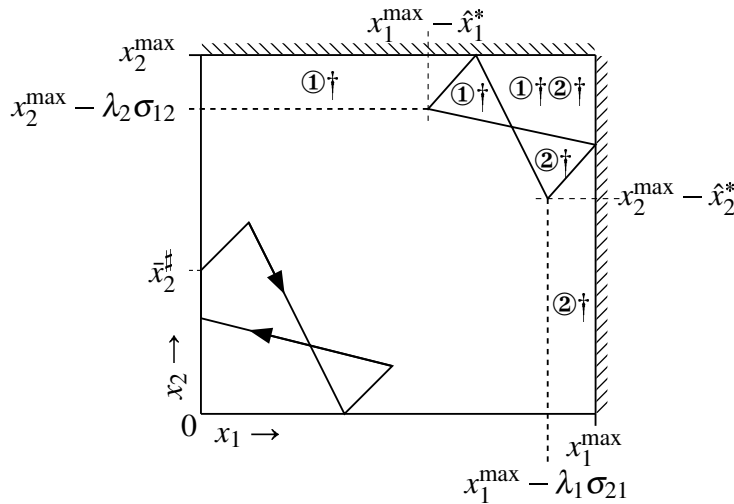


Figure 5.10: Feasible and infeasible regions for trajectories, subject to buffer level constraints.

Proposition 5.15. *A feedback which stabilizes a trajectory to the derived optimal process cycle if started from a feasible starting point (see Figure 5.10) is given by:*

- *Mode 1: ① at μ_1 as long as $x_1 > 0$ and $x_2 < x_2^{\max} - \lambda_2 \sigma_{12}$, then go to Mode 2.*
- *Mode 2: ① at λ_1 as long as $x_2 < \bar{x}_2^\sharp$, then go to Mode 3.*
- *Mode 3: perform ②, after σ_{12} go to Mode 4.*
- *Mode 4: ② at μ_2 as long as $x_2 > 0$ and $x_1 < x_1^{\max} - \lambda_1 \sigma_{21}$, then go to Mode 5.*
- *Mode 5: ② at λ_2 as long as $x_1 < \bar{x}_1^\sharp$, then go to Mode 6.*
- *Mode 6: perform ①, after σ_{21} go to Mode 1.*

Dependent on the state of the system, the controller is in one of these modes, which follows trivially from the mode description.

Proof. See Appendix A.4 □

The switching server with two lot types and setup times now has a suitable controller, for which convergence has been proven. In the next section, the controller is implemented in a case study, where the hybrid fluid model has been replaced with a discrete event simulation and with stochastic process times.

5.5 Case study: controller implementation

For both the unconstrained and the constrained buffer situation, optimal process cycles have been derived and state feedback controllers have been proposed. Although convergence was proven analytically, the controllers are validated in a case study, because the controller has been proposed based on a fluid model. Validation on discrete event models is needed. In this section an optimal process cycle is determined for the situation with finite maximum buffer capacities. A feedback controller (as proposed in Section 5.4) is implemented. The controller uses measurements of the state to determine its control actions. The proposed controller is implemented in three different simulations:

- original hybrid fluid model which was used in the derivation of optimal process cycles;
- deterministic discrete event simulation of the switching server;
- and a discrete event simulation with stochastic inter-arrival times and stochastic process times of lots.

The results of the simulations are not only compared with each other, but also with other controllers:

- a controller which only uses a clearing policy and a mechanism to prevent buffer constraint violations;
- a controller as proposed by Lan and Olsen [69], which eventually boils down to a clearing

policy for the specific numerical example;

- and a Savkin-type controller (see [97–99]) which uses fixed time spans for processing certain lot types.

The discrete event models have been specified in language χ (see [9]). The models have been included in Appendix B.2. The parameter settings and buffer level constraints used for the simulation are presented in Table 5.1. The cost weighting factors c_1 and c_2 are equal to 1. The initial state of the system is also given in Table 5.1.

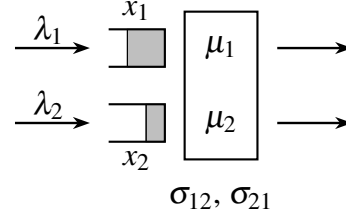


Figure 5.11: Switching server.

The condition for a slow-mode is evaluated:

$$c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2) (1 - \rho_2) = -\frac{23}{24} < 0 \quad (5.13)$$

so a slow-mode occurs for type 1 lots. Note that the slow-mode occurs for the lot type with highest $c\lambda$ index and lowest $c\mu$ index, (cf. Buyukkoc et al. [23]). The optimal process cycle is characterized as follows: $\bar{x}_2^b = 15$, $\bar{x}_2^\# = 18$, $\hat{x}_2 = 24$, $\bar{x}_1^\# = 27$ and $\hat{x}_1 = 45$. The process interval lengths are as follows: $\tau_1^\mu = 3$ hours, $\tau_1^\lambda = 1$ hour, $\tau_2^\mu = 1$ hour. The total period length in the steady state optimal process cycle is 9 hours. The buffer level capacity constraints did not influence the determined optimal periodic orbit.

Table 5.1: Parameter settings and initial conditions for controller implementation simulations.

λ_1 : 9 lots/hr.	x_1^{\max} : 70 lots
λ_2 : 3 lots/hr.	x_2^{\max} : 40 lots
μ_1 : 24 lots/hr.	$x_0(0)$: 0 hrs.
μ_2 : 27 lots/hr.	$x_1(0)$: 50 lots
σ_{12} : 2 hrs.	$x_2(0)$: 20 lots
σ_{21} : 2 hrs.	$m(0)$: 2

5.5.1 Hybrid fluid model simulation

The hybrid fluid model (5.1)–(5.4) has been simulated with MATLAB. The source of the simulation code has been included in Appendix B.2, along with a detailed explanation of the script.

Results of the simulation are shown in Figures 5.12 and 5.13. The orbit goes from light-gray to black in Figure 5.12, for better visual understanding. The graph axes limits correspond with the maximum buffer capacities. The first and second setup that take place are invoked by the buffer level constraint. That is why the buffer is not cleared before a setup to the other lot type takes place. After three setups, the trajectory touches the $x_1 = 0$ axis below $\bar{x}_2^\#$. From this point, the trajectory is on the determined optimal periodic orbit. A slow-mode is performed until the

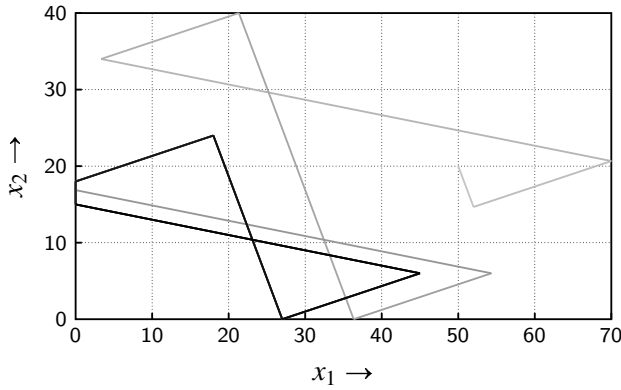


Figure 5.12: Orbit of hybrid fluid model simulation.

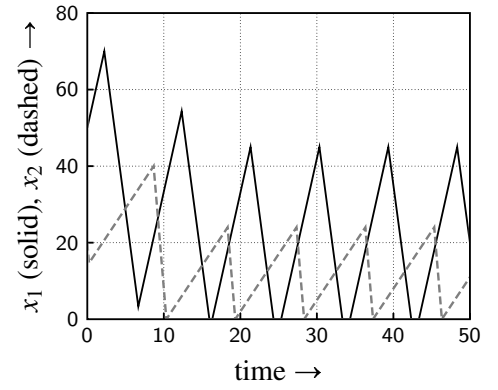


Figure 5.13: Buffer levels in hybrid fluid model simulation.

buffer level of type 2 lots reaches $\bar{x}_2^\#$, after which the setup to process type 2 lots takes place, etc. Figure 5.13 shows the individual buffer levels over time.

5.5.2 Deterministic discrete event simulation

Both a deterministic and a stochastic discrete event simulation have been carried out. Formalism χ 1.0 has been used [8, 9, 59], which has been introduced in Section 2.1.3. The model of the deterministic simulation has been included in Appendix B.2. An iconic model of this model is shown in Figure 5.14. Lots are sent from two generators G to buffer B . This buffer keeps two lists where lots are stored before they are processed by machine M . Finished lots are sent to the exit process E . The feedback controller has been implemented in the buffer process, since it makes use of the actual buffer contents and the mode of the system. The controller decides when the machine has to switch modes; this information is sent to machine M over channel e . The lot type in this model is simply an identification number, but this information is not used explicitly. Another possibility would have been to communicate by means of synchronization and use counters in the buffer process for the actual buffer levels. The latter has not been implemented to facilitate performance measurement (see Section 5.5.7): to measure quantities like flow time, more information about the lots has to be stored in the lot variable. The simulation results (Figures 5.15 and 5.16) show great resemblance with the hybrid fluid model simulation. The lines in the graph however look like stairs, due to the integer valued variables.

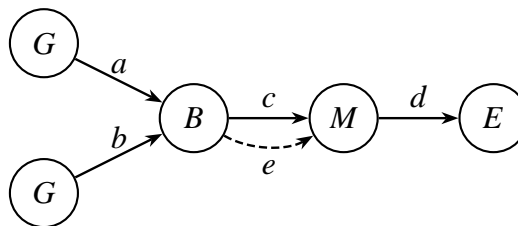


Figure 5.14: Iconic representation of χ model of two queues switching server.

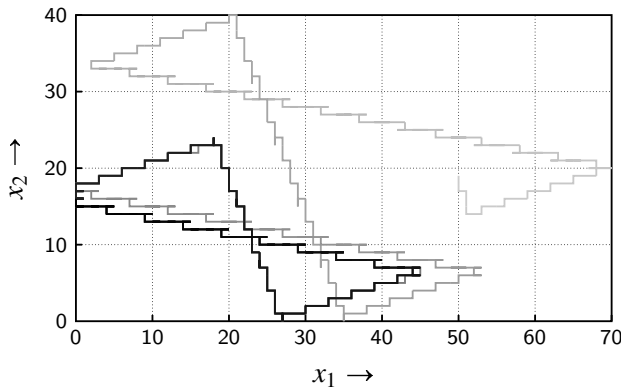


Figure 5.15: Trajectory of deterministic discrete event simulation.

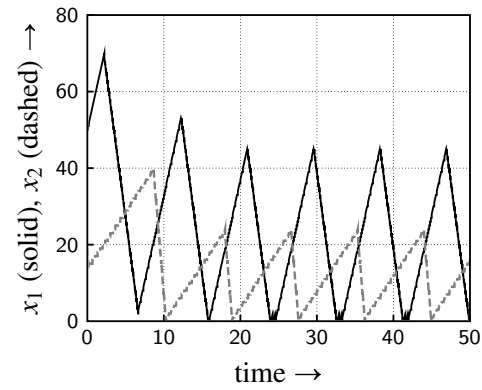


Figure 5.16: Buffer levels in deterministic discrete event simulation.

5.5.3 Stochastic discrete event simulation

A stochastic simulation has been carried out for the workstation whose characteristics have been presented in Table 5.1. The inter-arrival times and process times of lots are exponentially distributed (with their means equal to the means in the deterministic case). The χ model of this simulation has been included in Appendix B.2. The iconic representation is similar to the deterministic case, as shown in Figure 5.14. In the stochastic case, it is possible that buffer capacity constraints are violated, e.g. when many lots arrive within a short time. The discrete event model then interrupts the arrival of lots until space is available again in the buffer. This method is used in all stochastic discrete event simulations in this thesis. Results of one simulation are shown in Figures 5.17 and 5.18. As can be seen, the stochastic behavior has a great impact on the shape of the orbit, but the truncated bow tie can be distinguished (at least the slopes of the lines). The switchover coordinates are also very alike the deterministic simulation.

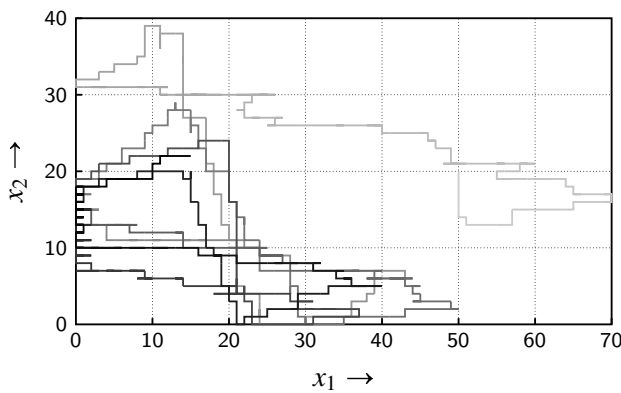


Figure 5.17: Trajectory of stochastic discrete event simulation.

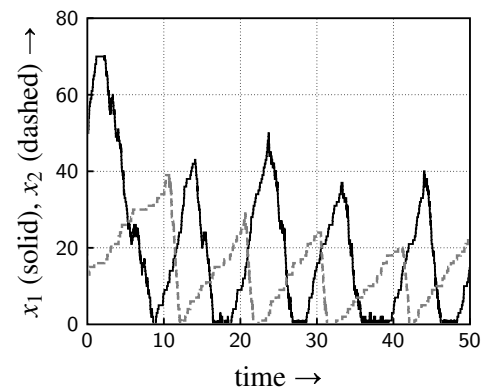


Figure 5.18: Buffer levels in stochastic discrete event simulation.

5.5.4 Clearing policy

A simple but effective strategy to stabilize a workstation (with $\rho < 1$) and reduce the number of lots in the system is by using a clearing policy. This policy states that buffers should be emptied and after a buffer has been emptied, a setup to another lot type takes place. A clearing policy is implemented in a simulation with the hybrid fluid model. An extension is made to make sure that buffer capacity constraints are not violated. To this purpose, similar switching rules as in the proposed controller (Proposition 5.15) are added to deal with the finite buffer capacities. It is expected that the trajectory is steered towards the pure bow tie curve (cf. the proof of Proposition 5.13 in which the trajectory converges to the pure bow tie curve if slow-modes are not taken into account). The same initial conditions are used in the simulation as in the previous simulations (Table 5.1). Results of the simulation are shown in Figures 5.19 and 5.20. Indeed, the trajectory is steered towards the pure bow tie curve and does not contain a slow-mode, as it did with the proposed controller.

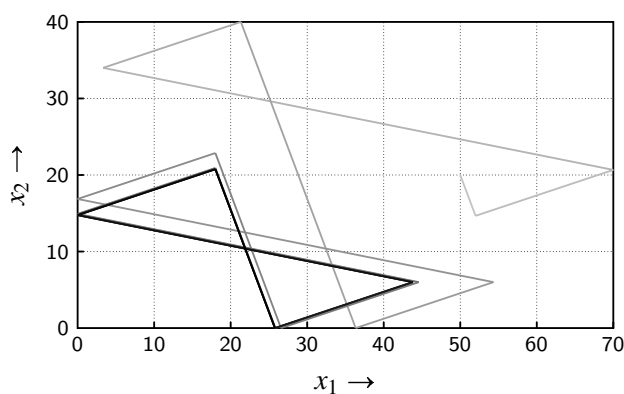


Figure 5.19: Trajectory of hybrid fluid model simulation with clearing policy.

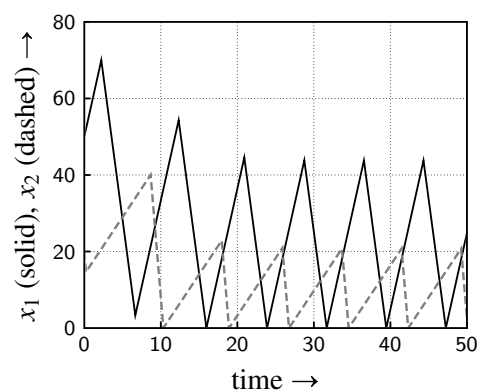


Figure 5.20: Buffer levels in hybrid fluid model simulation with clearing policy.

5.5.5 Lan and Olsen [69]

In Lan and Olsen [69] a theoretical lower bound on the mean wip level is computed for a multi-product server with both setup times and setup costs. The analysis is based on a similar fluid model assumption as used in this chapter. The theoretical lower bound that is computed might not be achievable in practice. Lan and Olsen introduce cruising conditions (they use ‘cruising’ as a term for processing lots in a slow-mode) to determine which queue is most effective to cruise at and whether enough capacity exists to cruise at all. Furthermore, they use a clearing policy with double thresholds to steer a trajectory to the desired one, similar to the controller that was proposed in this chapter. For the numerical example in this section, their cruising conditions do *not* hold. Eventually, they obtain the pure bow tie curve as desired periodical orbit, with equal performance as the clearing policy from the previous section. The work of Lan and Olsen is explained in more detail in Section 5.7, which treats workstations that serve

more than two product types. Finally, it should be noted that Lan and Olsen [69] do not handle buffer level constraints in their analysis and controller.

5.5.6 Savkin [98]

The control method that is proposed in [97–99] by Savkin is essentially a method to stabilize a network of servers. *Stable* here means that all buffer lengths remain bounded. A minimal period length for the network process cycle is derived and the process interval lengths (how long should the server process lots of each type) are then determined. The result is a feed forward controller, which does not base its control actions on measurements of the state, but follows a fixed time-based process cycle. A disadvantage of this approach is that it does not reduce the number of lots in the system to an optimal process cycle: from the starting point a feasible process cycle is obtained.

A single switching server processing two different lot types in fact is the smallest possible network. The Savkin controller is implemented in the hybrid fluid model with the same initial conditions as the previous simulations. Since [98] does not take finite buffer capacities into account, it is assumed that lots which arrive at a full buffer are scrapped. Note that this is a start-up effect, which takes place at most once per lot type in the deterministic case.

The minimal period length of a stable process cycle is the period length of the pure bow tie curve. For the parameters in Table 5.1, the minimal period length is $\frac{288}{37} \approx 7.78$ hours. Processing lots of type 1 takes $\frac{108}{37} \approx 2.92$ hours and processing type 2 lots takes $\frac{32}{37} \approx 0.86$ hours. The two setup times of two hours each complete the period length. The simulation is started with processing type 2 lots, where it is assumed that the setup to type 2 lots has just completed. Buffer 2 is emptied, after which a very small amount of time ($\frac{7}{222}$ hour) is left for processing type 2 lots in slow-mode. Then the setup to type 1 lots takes place. During this setup, buffer 1 gets fully loaded before the setup is complete. A total of $5\frac{58}{74}$ lots of type 1 is scrapped now. After this setup, the trajectory is in point (70, 6) in the (x_1, x_2) plane and is able to process at its bow tie curve. The trajectory and the buffer levels are visible in Figures 5.21 and 5.22.

As can be seen in the figure, buffer 1 is never emptied in its stable cycle. This already indicates that the obtained trajectory is not an optimal one (cf. Lemma 5.4). The minimal buffer length value of type 1 lots is $26\frac{8}{37}$, so with respect to the bow tie curve obtained by a clearing policy, the mean wip level is increased by $26\frac{8}{37}$ for the Savkin control policy. Note that for different initial settings for the simulation, this number may vary.

5.5.7 Performance comparison and evaluation

Different simulations with the proposed controller have been carried out: a hybrid fluid model simulation, a deterministic discrete event simulation, a stochastic discrete event simulation and a simulation with the hybrid fluid model using a clearing policy. The performance of the con-

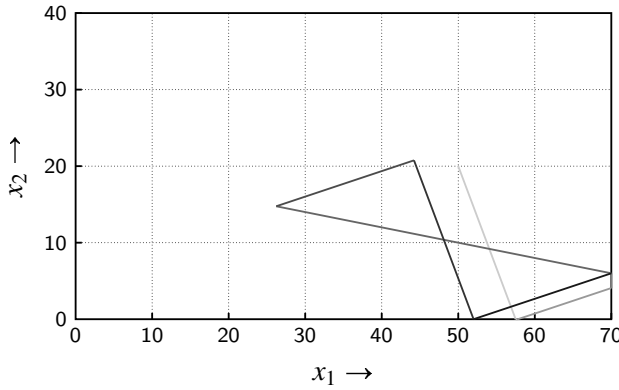


Figure 5.21: Trajectory of hybrid fluid model simulation with controller based on Savkin [98].

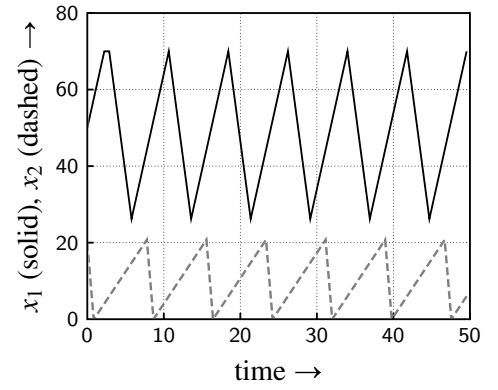


Figure 5.22: Buffer levels in hybrid fluid model simulation with controller based on Savkin [98].

trolled system is determined. For the given throughput (which equals the total inflow in steady state for reasons of mass conservation), the mean wip level and mean flow time are determined using Little's law [75]:

$$w = \delta \cdot \varphi$$

in which w denotes the mean work in process level, δ is the throughput and φ is the mean flow time of lots. For all simulations, results are shown in Table 5.2.

In the hybrid fluid model simulation, steady state is reached when buffer level x_1 becomes zero for the first time. From that point, the determined optimal process cycle is followed exactly. For both lot types, the mean wip level and flow time follow from the hybrid fluid model analysis and Little's law.

For the discrete event simulations, any possibly present transient part of the trajectory has been cut-off by computing the mean wip levels and flow times only after 30 process cycles, guaranteeing that the system was in steady state and that all lots in the system had been generated by the generators and not as part of the initial state. After this transient period, 100 process cycles have been performed. The mean flow times of the lot times were then determined. This simulation has been repeated 20 times to compute the mean and standard deviation of the mean flowtimes. With Little's law the mean wip levels were determined.

The differences between the deterministic discrete event model and the hybrid fluid model are caused by the fact that processing a lot takes time. So when a buffer is empty and the switching conditions are met, the switch cannot take place immediately, because first the final lot has to be finished processing. This causes small differences in the mean wip level. A more elaborate explanation of the effects of the discrete nature of the buffer levels is given in Section 5.6.

The stochastic simulation results show that rather good results are obtained with the controller. In Table 5.2, in addition to the mean values the standard deviations are shown. Taking these values into account, one can conclude that no significant rise of the mean wip levels and flow times occur due to the stochastic nature of the arrival and process rates. It can be investigated

Table 5.2: Performance comparison for different simulations with a single switching server, processing two lot types.

simulation	lot type	mean wip level	mean flow time
hybrid fluid model	1	20	$\frac{20}{9} \approx 2.22$
	2	12	4
	1+2	32	$\frac{8}{3} \approx 2.67$
deterministic discrete event	1	20.10	2.23
	2	11.68	3.89
	1+2	31.78	2.65
stochastic discrete event (20 simulations over 100 cycles)	1	20.34 ± 0.46	2.26 ± 0.052
	2	12.31 ± 0.16	4.10 ± 0.052
	1+2	32.63 ± 0.46	2.72 ± 0.038
clearing policy	1	$\frac{810}{37} \approx 21.89$	$\frac{90}{37} \approx 2.43$
Lan and Olsen [69]	2	$\frac{384}{37} \approx 10.38$	$\frac{128}{37} \approx 3.46$
	1+2	$\frac{1194}{37} \approx 32.27$	$\frac{199}{74} \approx 2.69$
Savkin [98]	1	$\frac{1780}{37} \approx 48.11$	$\frac{1780}{333} \approx 5.35$
	2	$\frac{384}{37} \approx 10.38$	$\frac{128}{37} \approx 3.46$
	1+2	$\frac{2164}{37} \approx 58.49$	$\frac{541}{111} \approx 4.87$

what the influence is of the values of $x_1^\#$ and $x_2^\#$ on the stochastic simulation results. Slight adaptation of these values (which have been obtained from the fluid model analysis without stochastics) might result in better performance for the stochastic situation. Another interesting topic is studying the influence of the parameters of the distributions on the process cycle and performance. This is not investigated in this thesis and is left for future research.

The clearing policy (with finite buffer constraint prevention) settles the trajectory down to the pure bow tie curve. For the given numerical example this is not an optimal process cycle, since introducing a slow-mode results in lower mean wip levels, as the hybrid fluid model simulation showed. The controller that is proposed by Lan and Olsen [69] eventually boils down to a clearing policy for the given numerical example. The cruising conditions, which indicate whether a lot type should be processed in slow-mode, do not hold for both lot types. Therefore the controller reduces to a clearing policy for both lot types. Finally, the Savkin control strategy performed as predicted: the number of lots in the system is not reduced where

possible. Due to the feed forward mechanism with fixed process interval lengths, the period of one cycle cannot be enlarged to process more lots in a cycle than arrive during the minimal cycle period. Therefore this controller only stabilizes the system and does not reach an optimal cycle. It is questioned how the trajectory evolves in a disturbed environment (e.g. stochastics), since the controller does not measure anything from the system under control. Whether the buffer levels remain stable (bounded) under the Savkin policy with disturbances has not been investigated in this thesis.

5.6 Discrete event analysis

The derivation of an optimal process cycle has been performed with a hybrid fluid model, in which buffers can have any non-negative real value. Individual lots cannot be distinguished. In the discrete event simulations of the previous sections, individual lots are distinguished. The buffer levels are integer valued then. For the deterministic case, the effects of the discrete nature of the buffer levels on the analysis of process cycles are studied in this section.

Given a buffer with lots arriving with a constant inter-arrival time, how many lots can be processed before the buffer is empty? And how long does this take? This depends on the ‘phase’ of the arrival process. Consider a constant arrival rate of λ lots/hour. This means that every $1/\lambda$ hours, a lot arrives at the buffer. For the continuous setting, being in the middle between two successive arrivals would be translated to a half lot in the buffer. Suppose that the last arrival took place δ hours ago and the discrete buffer level is x lots. In the continuous setting, the buffer level would be $x + \delta\lambda$. In Figure 5.23, the continuous approximation and discrete equivalent of the arriving lots are shown with the gray solid and dashed lines. Two examples are given in the figure. On the left hand side, $x = 2$ and $\delta = 4$, with $\lambda = \frac{1}{5}$. On the right hand side, $x = 2$ and $\delta = 2$. The black solid and dashed line represent the number of lots that have been taken from the buffer, for the discrete event and continuous case respectively. At $t = 0$, the machine starts processing lots ($\mu = \frac{1}{2}$), so immediately one lot is taken from the buffer (as can be seen in the graphs). To find the intersection point of the two continuous approximation lines, the following equation is to be solved:

$$\mu t + 1 = \lambda t + x + \delta\lambda \quad (5.14)$$

which gives as a result for the number of lots n that can be processed:

$$n = \mu \left(\frac{x + \delta\lambda - 1}{\mu - \lambda} \right) + 1. \quad (5.15)$$

If n is not an integer, the actual number of lots that has entered or left the buffer is:

$$n = \left\lfloor \mu \left(\frac{x + \delta\lambda - 1}{\mu - \lambda} \right) + 1 \right\rfloor = \left\lfloor \mu \left(\frac{x + \delta\lambda - 1}{\mu - \lambda} \right) \right\rfloor + 1 \quad (5.16)$$

in which $\lfloor A \rfloor$ denotes the largest integer less than or equal to A . Since $\mu > \lambda$ (stability issue), the dashed black line makes its discrete step earlier than the dashed gray line after the intersection

point of the solid lines. This would mean that a lot would be taken from the buffer that had not yet arrived. This is not possible, so $n = \left\lfloor \mu \left(\frac{x + \delta\lambda - 1}{\mu - \lambda} \right) \right\rfloor + 1$ is an upper bound on number of lots that can be processed. Left of the intersection point of the solid lines, again $\mu > \lambda$, so the final discrete step of the black dashed line prior or equal to the intersection point must be later than or equal to the last discrete step of the gray dashed line prior or equal to the intersection point. This means that the final lot before the intersection point arrived earlier or at the same time instant than it was started on the machine. The derived number of lots n therefore also is the lower bound. So at the moment a workstation switches modes and has completed the setup time, the number of lots that can be processed before the buffer is empty equals:

$$n = \left\lfloor \mu \left(\frac{x + \delta\lambda - 1}{\mu - \lambda} \right) \right\rfloor + 1. \quad (5.17)$$

The duration of this process interval is $n \cdot \frac{1}{\mu}$ time units. For the left hand side graph of Figure 5.23, the number of lots that can be processed is four, while in the right hand side situation, only three lots can be processed. These points are indicated in the figure by black dots.

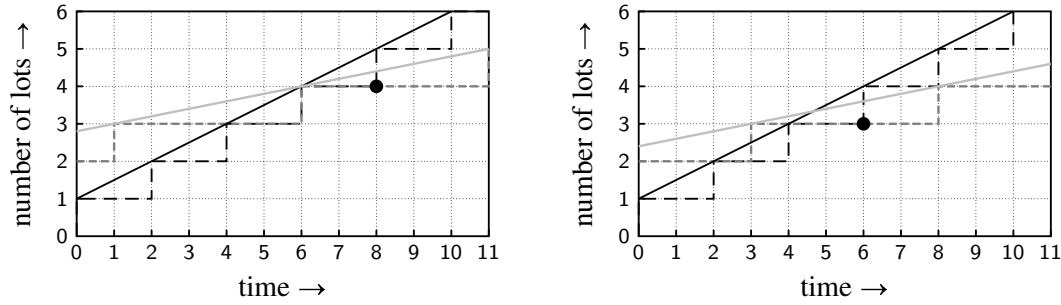


Figure 5.23: The black dots indicate the number of lots that can be processed given a buffer level and phase of the arrival pattern. Gray solid and dashed line: number of lots that have arrived at the buffer. Black solid and dashed line: number of lots that have been taken from the buffer.

When using expressions like (5.17) in analyses, rather complicated expressions for characterizing process cycles arise, if possible at all to state them explicitly. Also in further computations (e.g. convergence proofs) the discrete expressions may become too complex to show convergence. Therefore it is not convenient to use discrete event analysis for determining desired process cycles. Hybrid fluid models are an abstraction of the discrete event character of the workstation, but allow for relatively easy analysis and computations. The fluid approximation is generally accepted to be valid for situations where many lots are involved and where the process time is relatively short compared to the flow time of lots. These issues become visible when the state feedback controllers based on the fluid model analysis are implemented in a discrete event simulation. Especially when the threshold levels for switching, $x_1^\#$ and $x_2^\#$, are non-integer, one should decide at which moment to switch. For example, if $x_2^\# = 18\frac{1}{2}$, when should the workstation switch? Exactly between two successive arrivals of lots? And what to do with the lot that is currently being processed? These issues are not addressed here. In the case studies of Section 5.5, the workstation only switches after a lot has been completed and

the threshold level has been reached. The effects of different controller implementations can be studied, but are left for future research.

5.7 On multiple product types and optimal process cycles

So far, only workstations serving two product types have been considered. But what if a workstation processes more than two different lot types? The complexity of deriving an optimal process cycle is not comparable to the two-product case. For the situations with two lot types, the order in which mode switches take place is fixed: mode 1, mode 2, mode 1, mode 2, etc. However, for more than two lot types, the order of mode switches is unknown. For example, in the three-product type situation, after serving lots in mode 1, the workstation can switch to either mode 2 or mode 3. Moreover, it is not known beforehand how many switches have to be performed within one process cycle. It is tempting to serve each lot type once and then start a new cycle. Serving certain lot types more than one time during a cycle is also a very reasonable option: especially when the load of the different lot types on the server is unbalanced. Apart from the search for the best process cycle, it can be questioned whether optimal system behavior is periodic at all.

Lan and Olsen [69] provide a theory to compute a lower bound on the mean work in process level for a server with n lot types, $n \in \mathbb{N}$. This lower bound on the mean wip level is a good reference point, but not a practically reachable situation. Their lower bound is based on the assumption that in each buffer-time graph (e.g. Figure 5.13), the buffer peaks are equally high every time that particular mode is visited (cf. Lemma 5.4 and Lemma 5.6 of this chapter). For completely symmetric systems (equal process rates, arrival rates and setup times for all lot types) the theoretical lower bound can be achieved, but for all other situations this might not be the case. In addition, in [69] the number of switches to each mode in one cycle is expressed as a continuous variable: the number of switches to a particular mode per time unit. If these continuous variables are non-rational, it is not possible to find a process cycle that visits the queues exactly in the prescribed ratios.

In Lan and Olsen [69] slow-modes are taken into consideration. Processing in slow-mode is called *cruising* and the paper provides two *cruising conditions* which determine the queues which are most effective to cruise at and determine whether sufficient capacity exists to make cruising cost effective. However, for the two-product case of Section 5.5, the cruising conditions do not hold, while the optimal process cycle as derived in Section 5.2 does contain a slow-mode. The theoretical lower bound on the wip level by Lan and Olsen for this two product workstation is set to 31,2084. The exact and reachable lower bound based on the hybrid fluid model analysis is 32. For two lot types, [69] may yield unequal visit frequencies for the lot types, while for two product types the number of visits of each queue per time unit in a process cycle is always equal. Therefore the theoretical lower bound in [69] can never be achieved. An exact lower bound can be found by using the theory of Sections 5.2 and 5.4.

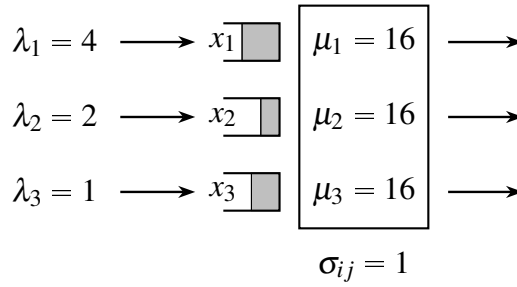


Figure 5.24: Switching server with three different product types.

For a workstation serving more than two lot types, [69] provides a good reference point for the performance of an arbitrary process cycle. As already mentioned, determining an optimal process cycle might be a job too difficult to perform (if possible or existing at all: enlarging the process cycles over and over again might result in ever decreasing mean wip levels for particular situations), because the number of mode switches and the order of mode switches are unknown. Experimenting with different choices may however give insights in the structure of optimal process cycles. Consider for example a workstation serving three product types, as shown in Figure 5.24. All parameters values are as indicated in the figure. All setup times are set to one time unit. Equal costs are put on the individual wip levels. The theoretical fluid bound according to Lan and Olsen [69] is 14.3875, and the cruising conditions do not hold, indicating that once a buffer has been emptied, the wisest thing to do is immediate switching to another lot type. Since $c\mu = 16$ for all queues (assuming equal weights $c_i = 1$, $i \in \{1, 2, 3\}$), a buffer should be cleared before switching to another queue, see Buyukkoc et al. [23]. The number of setups per time unit for lot type 1, 2 and 3 are 0.242, 0.185 and 0.135 respectively. This is roughly a 4:3:2 ratio of the number of setups to the different lot types in a period. For this ratio, and several other ratios, the mean wip level has been determined using the hybrid fluid model, similar to the two-product case of Section 5.1, and presented in Table 5.3. The small MATLAB source of these computations has been included in Appendix B.2. The table shows that for different mode ratios, different mean wip levels are obtained. Even within a certain mode ratio, different process cycles are possible, resulting in different mean wip levels. For the 4:3:2 and 5:4:3 ratio, only two of all possible process cycles are shown in the table.

After a desired steady state process cycle has been chosen for a server with multiple lot types, a feedback controller is to be developed. In [112] a controller is developed for the $\begin{bmatrix} 1 & 2 & 1 & 3 \end{bmatrix}$ process cycle (shown in gray in Table 5.3). This controller uses a clearing policy: it immediately switches to another lot type after a buffer has been emptied. After processing in mode 2 or mode 3, the control action is easy: switch to mode 1. After serving lots of type 1, the controller has to decide to which lot type to switch. The controller that has been developed in [112] makes this decision based on the buffer level of type 2 lots. When it is above a certain level, switch to mode 2, otherwise switch to mode 3. Convergence of the system's trajectory to the desired process cycle has been proven for this control strategy.

For workstations serving multiple product types, finding optimal process cycle with respect to mean weighted wip level is very complicated, if possible at all. One can imagine that the complexity of this problem grows with the number of lot types that is to be served. Therefore, one should reside to an *acceptable* process cycle, which could be a personally weighted combination of simplicity and mean wip level. This process cycle can then be regarded as the *desired* process cycle. Afterwards, a new challenge is to develop feedback controllers that steer a system to this desired process cycle. Theory to develop controllers for manufacturing networks based on Lyapunov's direct method has already been introduced by Lefeber and Rooda [73] and will be elaborated on in the near future.

Table 5.3: Steady state mean wip levels for a three product workstation with clearing policy and without slow-modes.

ratio	process cycle	mean wip
1:1:1	[1 2 3]	$\frac{91}{6} \approx 15.1667$
2:1:1	[1 2 1 3]	$\frac{403}{27} \approx 14.9259$
3:2:1	[1 2 1 2 1 3]	$\frac{1549}{99} \approx 15.6465$
4:3:2	[1 3 2 1 2 1 3 1 2]	≈ 15.0341
4:3:2	[1 2 3 1 3 1 2 1 2]	≈ 17.2194
5:4:3	[1 2 1 3 2 1 2 1 3 2 1 3]	≈ 15.1983
5:4:3	[1 2 1 3 2 1 2 3 1 2 1 3]	≈ 14.8930

5.8 Piecewise constant arrival rates

In the previous sections, a single workstation processing two product types has been considered. The arrivals of lots at the workstation have a fixed interarrival time and switching from one product type to the other takes a fixed amount of time. Optimal process cycles have been defined and feedback controllers have been proposed. However, in a practical industrial manufacturing system, the arrival rates, process rates and switchover times are in general non-constant. Dealing with disturbances (represented by variations on inter-arrival and process times) has been investigated in the discrete event simulations and the proposed state feedback controllers performed as expected. The disturbances that were applied to the single switching server were a distribution on the arrival and process rates, with the mean value taken from the fluid model. But what happens if for example the constant arrival pattern changes? In a manufacturing network, servers are connected to each other, resulting in highly varying arrival rates. If for example the outflow of a machine is the inflow for a downstream machine, the arrival pattern becomes piecewise constant: lots arrive at the second workstation when the preceding machine is pro-

cessing that specific type and no lots arrive if the preceding machine is in a different mode. This section investigates the influence of a time varying arrival pattern on optimal process cycles for a single switching server, processing two different lot types. More specifically, inflow patterns with piecewise constant arrival rates are considered.

Time varying arrival rates have been studied, but far less than constant arrival rates in manufacturing networks. Although in a manufacturing network of switching servers lots may arrive at workstations with a piecewise constant arrival rate, no literature has been found on servers with piecewise constant arrival rates. Queueing models for call centers have been studied extensively (see for example the survey paper by Koole and Mandelbaum [67] and references therein). A common assumption in call center modelling is that customers arrive according to a Poisson process. However, more customers call/arrive at the call center during office hours than at night. Therefore, a day is often partitioned into multiple time slots, in which the arrival rate is assumed to be constant. This results in piecewise constant Poisson arrivals. This assumption is reasonable when steady state is achieved relatively fast. An example of modelling call centers with piecewise constant Poisson processes is given in [63].

Call centers with time varying parameters have also been studied in [10], where the arrival rate and service rate are dependent on the workload of the system, where workload is interpreted here as the amount of work in the system. For a special case with two models in which the ratio between arrival and process rate is equal, steady state distributions are derived. The authors show that the workload relations are proportional and that the difference between the models is just a rescaling of time. This result is in accordance with the results presented in Chapter 5 of this thesis, where the shape of the steady state trajectories is determined by the partial workloads ρ_i (which is dimensionless) and scaled by the arrival rates λ_i .

In several papers, the number of servers in a workstation is adapted to the work in the system and the arrival rates of jobs, for example in [92], where two types of customers arrive at a workstation. The two types represent high and low priority customers. Low priority jobs that wait too long in queue are transferred to the high priority queue. Arrival rates are exponentially distributed. In this research however, the number of servers is fixed and lots do not jump from the one queue to the other.

Before optimal process cycles are derived in Section 5.8.2, first the characteristics and dynamics of a switching server processing two lot types and with two piecewise constant arrival rates are presented in Section 5.8.1.

5.8.1 Characteristics and dynamics of a single switching server with piecewise constant arrival rates

For a single switching server, processing two different lot types (see Figure 5.1), an optimal process cycle is looked for. Instead of constant arrival rates, piece-wise constant arrival patterns

of both lot types are considered. An important assumption is that only process cycles with the same period length as the arrival pattern period length are considered. In this section, first the characteristics of the arrival pattern and the dynamics of the workstation are stated. Goal is again to determine an optimal process cycle with respect to minimal weighted work in process levels (cf. (5.7)).

With respect to the arrivals of lots, a few assumptions are made:

- Lots of type i arrive either at rate $\hat{\lambda}_i$ or at rate 0.
- The arrival pattern is periodic with fixed period length P for both lot types.
- During one period, the arrival pattern consists of two parts: a time span in which lots arrive at rate $\hat{\lambda}_i$ and a time span in which no lots arrive.
- Without loss of generality, arrival of type 1 lots starts at time $t = 0$ in the periodic cycle. Type 2 lots start to arrive somewhere between $t = 0$ and $t = P$.
- The two lot types arrive independently from each other, so concurrent arrivals of both lot types are allowed.
- Arrivals are independent from the actions of the machine. In a manufacturing network, especially when re-entrant behavior is involved, the process cycle of a machine influences the arrival pattern at its own buffers. Here, the arrival pattern does not depend on the process cycle of the machine.

Examples of arrival patterns are shown in Figure 5.25. Let s_i denote the time instant type i lots start to arrive. From the assumptions, it is clear that $s_1 = 0$. In Section 5.8.2 expressions are derived for the mean wip levels for the two lot types. The shape of the input profiles (e.g. Figure 5.25) influences the complexity of the expressions that are to be used. Properly labelling lots as either ‘type 1’ or ‘type 2’ may decrease the complexity of the optimization problem of minimizing the mean work in process levels. For the most right-hand side situation in Figure 5.25, one could consider changing the labels of type 1 and type 2, resulting in a profile similar to the middle situation in the figure.

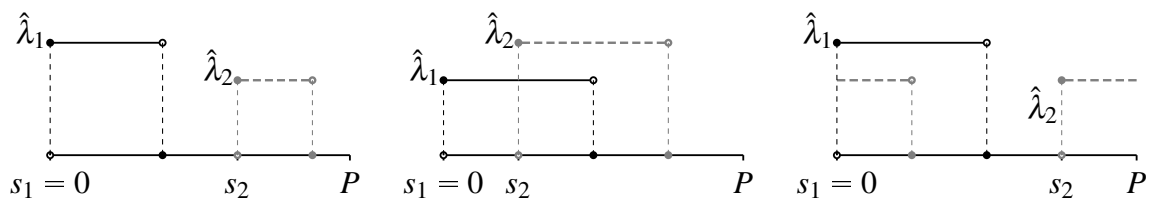


Figure 5.25: Examples of arrival patterns for switching server with two piecewise constant arrival rates. The length of the arrivals time span of type i lots is denoted by $\phi_i P$.

The duration of the interval in which lots arrive at the workstation is denoted by $\phi_i P$. In other words: lots arrive during fraction ϕ_i of the period length. In Figure 5.25 a few possible arrival patterns are shown. As can be seen in the figure, the signals are assumed to be right-continuous. Formally, the input pattern $\lambda_i(t)$ can be characterized as in (5.18), in which the first and second

expression for $\lambda_2(t)$ correspond to the middle and right hand side situation of Figure 5.25.

$$\lambda_1(t) = \begin{cases} \hat{\lambda}_1 & \text{for } 0 \leq t < \phi_1 P \\ 0 & \text{for } \phi_1 P \leq t < P \end{cases} \quad (5.18a)$$

$$\text{for } s_2 + \phi_2 P \leq P: \lambda_2(t) = \begin{cases} 0 & \text{for } 0 \leq t < s_2 \\ \hat{\lambda}_2 & \text{for } s_2 \leq t < s_2 + \phi_2 P \\ 0 & \text{for } s_2 + \phi_2 P \leq t < P \end{cases} \quad (5.18b)$$

$$\text{for } s_2 + \phi_2 P \geq P: \lambda_2(t) = \begin{cases} \hat{\lambda}_2 & \text{for } s_2 \leq t < P \\ 0 & \text{for } s_2 + \phi_2 P - P \leq t < s_2 \\ \hat{\lambda}_2 & \text{for } 0 \leq t < s_2 + \phi_2 P - P. \end{cases} \quad (5.18c)$$

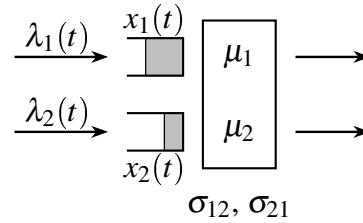


Figure 5.26: Switching server with two lot types and time-varying arrival rates.

Now that the arrival pattern of lots has been characterized, the workstation itself (Figure 5.26) is considered. Similar to the situation with constant arrival rates (in Section 5.1), the state elements and dynamics of the workstation with piecewise constant arrival rates are presented. Due to the varying input signal, the state and dynamics are slightly extended.

The state of the system consists of six elements. The remaining setup time x_0 , buffer levels x_1 and x_2 and mode m are again part of the state. In addition to these elements, it is necessary to ‘know’ the current position with respect to the input profiles. Let $\Delta_i \in (0, P]$ be the remaining time before a new arrival block of type i jobs starts. If Δ_i reaches zero, it is set to period length P again, resulting in a sawtooth, as shown in Figure 5.27. In this figure, an arbitrary arrival pattern and the resulting evolution of Δ_i has been plotted over two period lengths. The state $\mathbf{x}(t)$ can now be defined as:

$$\mathbf{x}(t) = \begin{bmatrix} x_0(t) & x_1(t) & x_2(t) & m(t) & \Delta_1(t) & \Delta_2(t) \end{bmatrix}^T \in [0, \max(\sigma_{12}, \sigma_{21})] \times \mathbb{R}_+^2 \times \{1, 2\} \times [0, P]^2. \quad (5.19)$$

The inputs of the system are equal to the inputs of the switching server with constant arrival rates:

$$\mathbf{u}(t) = \begin{bmatrix} u_0(t) & u_1(t) & u_2(t) \end{bmatrix}^T \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\} \times [0, \mu_1] \times [0, \mu_2]. \quad (5.20)$$

Input u_0 denotes the action which is to be performed: setting up for type 1 lots, processing type 1 lots (if available), setting up for type 2 lots or processing type 2 lots (if available). Inputs u_1

and u_2 denote the rate at which lots are to be processed (possibly zero). The *mean* workload $\bar{\rho}_i$ of lot type i can now be defined as:

$$\bar{\rho}_1 = \frac{\hat{\lambda}_1 \phi_1}{\mu_1} \quad (5.21)$$

in which the term $\hat{\lambda}_1 \phi_1$ can be interpreted as the mean arrival rate over the period.

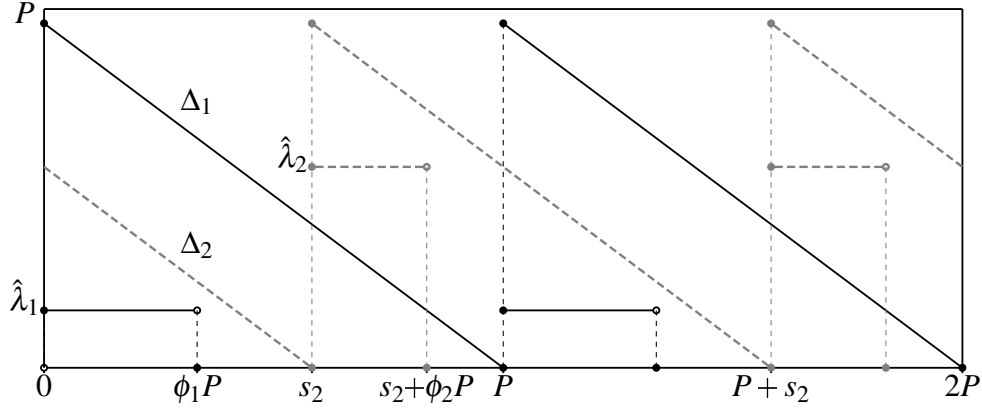


Figure 5.27: Evolution of Δ_1 and Δ_2 given arrival patterns.

The dynamics of the system can be divided into discrete event dynamics and continuous dynamics:

$$x_0 := \sigma_{21}, m := 1 \text{ if } m = 2 \wedge u_0 = \textcircled{1} \quad (5.22a)$$

$$x_0 := \sigma_{12}, m := 2 \text{ if } m = 1 \wedge u_0 = \textcircled{2} \quad (5.22b)$$

$$\Delta_i := P \text{ if } \Delta_i = 0, i \in \{1, 2\} \quad (5.22c)$$

$$\dot{x}_0(t) = \begin{cases} -1 & \text{for } u_0(t) \in \{\textcircled{1}, \textcircled{2}\} \\ 0 & \text{for } u_0(t) \in \{\textcircled{1}, \textcircled{2}\} \end{cases} \quad (5.22d)$$

$$\dot{x}_1(t) = \lambda_1(t) - u_1(t) \quad (5.22e)$$

$$\dot{x}_2(t) = \lambda_2(t) - u_2(t) \quad (5.22f)$$

$$\dot{\Delta}_1(t) = -1 \text{ with } \Delta_1(0) = P \quad (5.22g)$$

$$\dot{\Delta}_2(t) = -1 \text{ with } \Delta_2(0) = s_2. \quad (5.22h)$$

Similar to the situation with constant arrival rates, the inputs of the workstation are subject to constraints:

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

The constraints have the following interpretation:

- during a setup no lots can be processed;
- only one lot type can be processed at a time;
- maximum actual process rates exist based on the buffer levels and arrival patterns;
- at any time the system can switch to the other lot type (even while busy with another setup).

So far, the state vector, input vector and dynamics have been defined for the switching server with piecewise constant arrival patterns (taken from the class as described before). For this switching server, again an optimal process cycle with respect to minimal mean weighted work in process level is looked for. After determination of this process cycle, feedback control strategies can be investigated to steer the system to the desired process cycle from any arbitrary starting point. In the next section, an optimal process cycle is determined for the single switching server with piecewise constant arrival rates, under the assumption that the period length of the process cycle equals the period length of the input rate profile.

5.8.2 Optimal process cycle

Similar to the situation with constant arrival rates, the goal is to minimize the time averaged weighted work in process level:

$$\min_{t_1, t_2} (c_1 \bar{w}_1(t_1) + c_2 \bar{w}_2(t_2)) \quad (5.23)$$

in which \bar{w}_1 and \bar{w}_2 are the mean work in process levels for type 1 and type 2 products respectively and t_i is the start time of processing lot type i in a process cycle (which is explained later). Parameters c_1 and c_2 are weighting factors. This objective function is subject to constraints that couple the two lot types. In words, these constraints say: only one product type can be processed at a time and switching between types takes the setup time. These constraints are formalized at the end of this section.

Remark 5.16. The period length P is not a variable in the objective function, as it was in Section 5.2. The length of the process cycle is assumed to be equal to the period length of the arrival patterns. This facilitates the optimization procedure, because now only the (weighted) area underneath the buffer-time graph needs to be optimized. One should keep in mind that it is possible to find process cycles with a lower mean wip level when the period of a process cycle is not fixed to equal the period of the arrival pattern, but a multiple of this period.

An important assumption in the analysis is that during one period P , each lot type is served only once. In other words: only two setups take place within a period: from type 1 lots to type 2 lots and vice versa.

With the insights obtained in the first part of this chapter, a few lemmas can be stated that facilitate the derivation of optimal process cycles.

Lemma 5.17. *Minimizing mean wip level of type i implies serving lots of type i at the highest possible rate, after which the server may idle.*

Proof. See the proof of Lemma 5.2. □

Lemma 5.18. *In an optimal process cycle with respect to minimal time averaged weighted work in process levels, all buffers are emptied at least once.*

Proof. Consider a feasible process cycle that satisfies Lemma 5.17 in which the buffer level of type i has a minimal value of $\varepsilon > 0$ at the moment it switches to the other lot type. Consider also an alternative feasible process cycle with exactly the same process rate profile and with a buffer-time graph that is shifted ε downwards with respect to the first process cycle. This alternative process cycle has buffer level zero at that particular switch to the other lot type. The mean wip level is now reduced by ε , so the alternative process cycle has lower costs than the original process cycle. This result holds for any feasible process cycle that satisfies Lemma 5.17, so in optimal process cycles, all buffers are emptied at least once during the period. □

Optimization recipe

To find an optimal process cycle for the workstation serving two piecewise constant arriving lot types, a recipe is followed that is based on lot type distinction. For both lot types mean wip level expressions are developed. The fact that the machine can only process one lot type at a time is translated into constraints in the optimization procedure. The recipe can be summarized as follows:

- Mean wip level expressions are derived for the lot types separately, as a function of the start time of processing the lot type and the duration of the process interval.
- Combining the separate mean wip level expressions yields the objective function as stated in (5.23).
- The objective function can be optimized with respect to the start times and process interval lengths of the lot types. Additional constraints need to ensure that the workstation only serves one lot type at a time. These constraints are called *overlap constraints*.
- The overlap constraints cause the optimization problem to be split into two separate optimization problems: either type 1 lots are served before type 2 lots, or type 2 lots are served before type 1 lots. This results in two sets of overlap constraints. The two optimization problems are called *subproblems*.
- The solution of the two optimization subproblems with the lowest mean weighted wip level is the overall solution to the optimization problem. This solution defines the desired optimal process cycle.

The process interval length can be variable in case a slowmode may occur. The workstation can process all lots at maximum rate μ or at arrival rate $\hat{\lambda}$. It is also possible to process only a part of the lots at rate μ and the remaining lots at arrival rate $\hat{\lambda}$. However, a slow-mode is only possible if the arrival rate $\hat{\lambda}$ is smaller than the maximum process rate μ . Therefore, for each product type a distinction is made based on the difference between the arrival rates and maximum process rates. Proposition 5.19 states the mean wip level expressions for the two lot types, in

which this distinction is made. The two cases ($\hat{\lambda} \geq \mu$ and $\hat{\lambda} < \mu$) are shown in Figures 5.28 and 5.29 respectively. Each case consists of multiple situations, for which mean wip level expressions are derived. All situations are summarized in Proposition 5.19 and Appendix A.5. In the figures, the horizontal axis represents one period of the piecewise constant input signal and consequently also the period of the process cycle. Variable $t_1 \in [0, P)$ denotes the start time of processing type 1 lots, whereas $\tau_1 \in [0, P)$ represents the process interval length, i.e. how much time is reserved for processing lots of type 1. Similar variables are used for type 2 lots. In the figures, the arrival pattern of a lot type is shown with a solid line. The actual process rate profile is shown with a dashed line. The dotted line is the buffer level of the lot type.

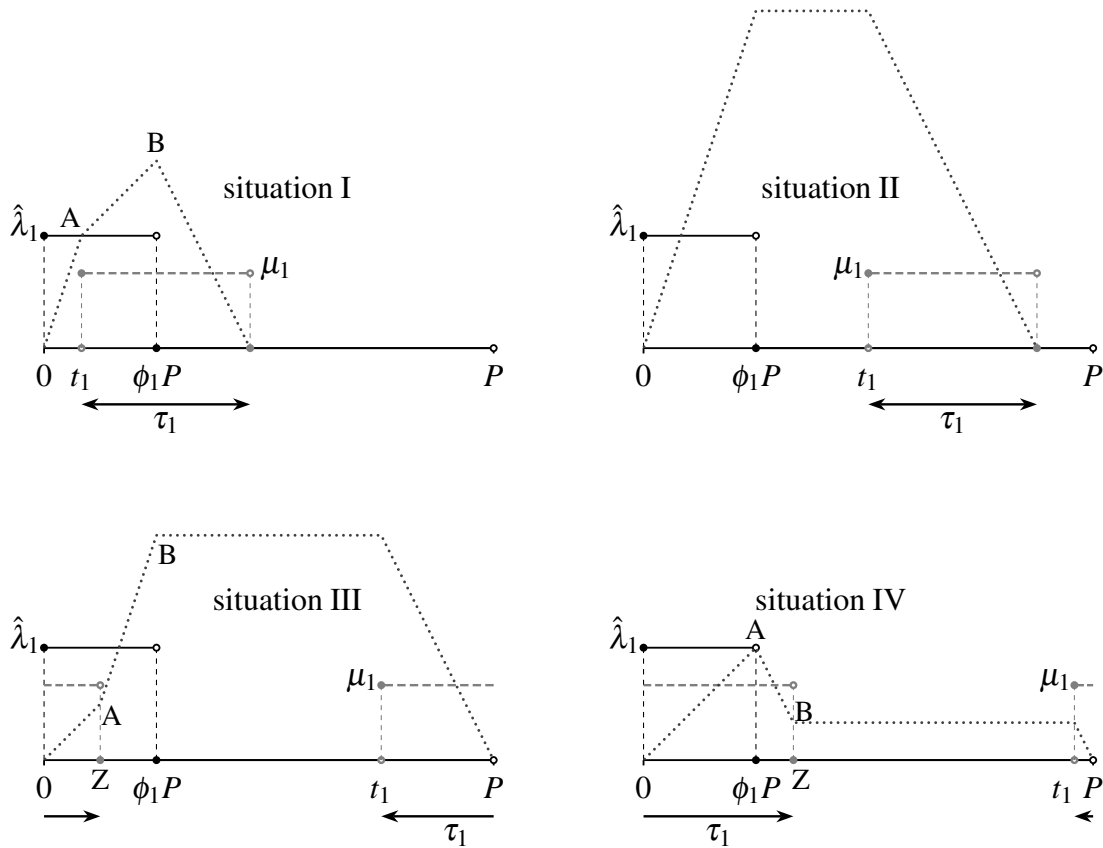


Figure 5.28: Different situations for type 1 and $\hat{\lambda}_1 \geq \mu_1$. Input rate profile (solid), process rate profile (dashed gray) and buffer level curve (dotted).

Proposition 5.19. *For a single switching server (as formally described by (5.22)) with two piecewise constant periodic arrival patterns with equal period lengths, the mean work-in-process levels \bar{w}_i are given by the following expressions. Roman numbers refer to different situations that can take place. These are all explained in detail in the proof. A graphical representation of the different situations is given in Figures 5.28 and 5.29. In case multiple situations can take place, optimization together with the other lot type determines which one is the actual mean wip level expression. In the mean wip level expressions, t_i and τ_i denote the start time of processing lots of type i and the process interval length of type i lots respectively.*

- For type 1 lots with $\hat{\lambda}_1 \geq \mu_1$ (Figure 5.28):

$$\bar{w}_1(t_1) = \begin{cases} \frac{1}{2}\mu_1\bar{\rho}_1P(\bar{\rho}_1 - \phi_1) + \mu_1\bar{\rho}_1t_1 & \text{for } 0 \leq t_1 < (1 - \bar{\rho}_1)P \\ \frac{1}{2}\mu_1\bar{\rho}_1P(\bar{\rho}_1 - \phi_1) + \mu_1(1 - \bar{\rho}_1)(P - t_1) & \text{for } (1 - \bar{\rho}_1)P \leq t_1 \leq P \end{cases}$$

$$\tau_1 = \bar{\rho}_1P.$$

- For type 1 lots with $\hat{\lambda}_1 < \mu_1$ (Figure 5.29):

$$\bar{w}_1^I(t_1, \tau_1) = \frac{\mu_1\bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} [\phi_1\tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1\bar{\rho}_1)\tau_1 - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 + \phi_1P^2(2(\phi_1 - \bar{\rho}_1)(1 - \phi_1) + \phi_1^2)]$$

for $0 \leq t_1 \leq (\phi_1 - \bar{\rho}_1)P$ and $\bar{\rho}_1P \leq \tau_1 \leq \phi_1P - t_1$

$$\bar{w}_1^{II}(\tau_1) = \frac{\mu_1\bar{\rho}_1(P - \tau_1)^2}{2P(\phi_1 - \bar{\rho}_1)}$$

for $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \phi_1P$ and $P - t_1 \leq \tau_1 \leq P$ and $\tau_1 \geq \bar{\rho}_1P$
and $\tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1^2}{\bar{\rho}_1}\right)P$

$$\bar{w}_1^{III-VI}(t_1, \tau_1) = \frac{\mu_1\bar{\rho}_1}{2\phi_1^2P} [((1 + \phi_1)P - \tau_1 - t_1)((\phi_1 - \bar{\rho}_1)t_1 - (\phi_1 + \bar{\rho}_1)\tau_1 + P(\bar{\rho}_1 + \phi_1 + \bar{\rho}_1\phi_1 - \phi_1^2))]$$

for $\tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1^2}{\bar{\rho}_1}\right)P$ and $\tau_1 \geq P - t_1$
and $\tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1}\right)P$ and $\tau_1 \leq (1 + \phi_1)P - t_1$

$$\bar{w}_1^{IV-V}(t_1) = -\frac{1}{2}\mu_1\bar{\rho}_1P(\phi_1 - \bar{\rho}_1) + \mu_1\bar{\rho}_1t_1 \text{ and } \tau_1 = \bar{\rho}_1P$$

for $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \max(\phi_1P, (1 - \bar{\rho}_1)P)$

$$\bar{w}_1^{VII}(t_1, \tau_1) = \frac{\mu_1\bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} [\phi_1\tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1\bar{\rho}_1)\tau_1 - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 - P^2(2\bar{\rho}_1 - 2\phi_1 + \phi_1^3 - 2\phi_1^2\bar{\rho}_1)]$$

for $(1 - \bar{\rho}_1)P \leq t_1 < P$ and $\bar{\rho}_1P \leq \tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1}\right)P$.

- For type 2 products, the expressions are similar to the mean wip expressions of type 1 products, but shifted in time, since the arrival pattern of type 2 lots starts s_2 time units later than the arrival of type 1 lots. The mean wip level expressions for type 2 products are given in Appendix A.5.

Proof. A detailed derivation of these expressions for the mean work in process level is provided in Appendix A.5. \square

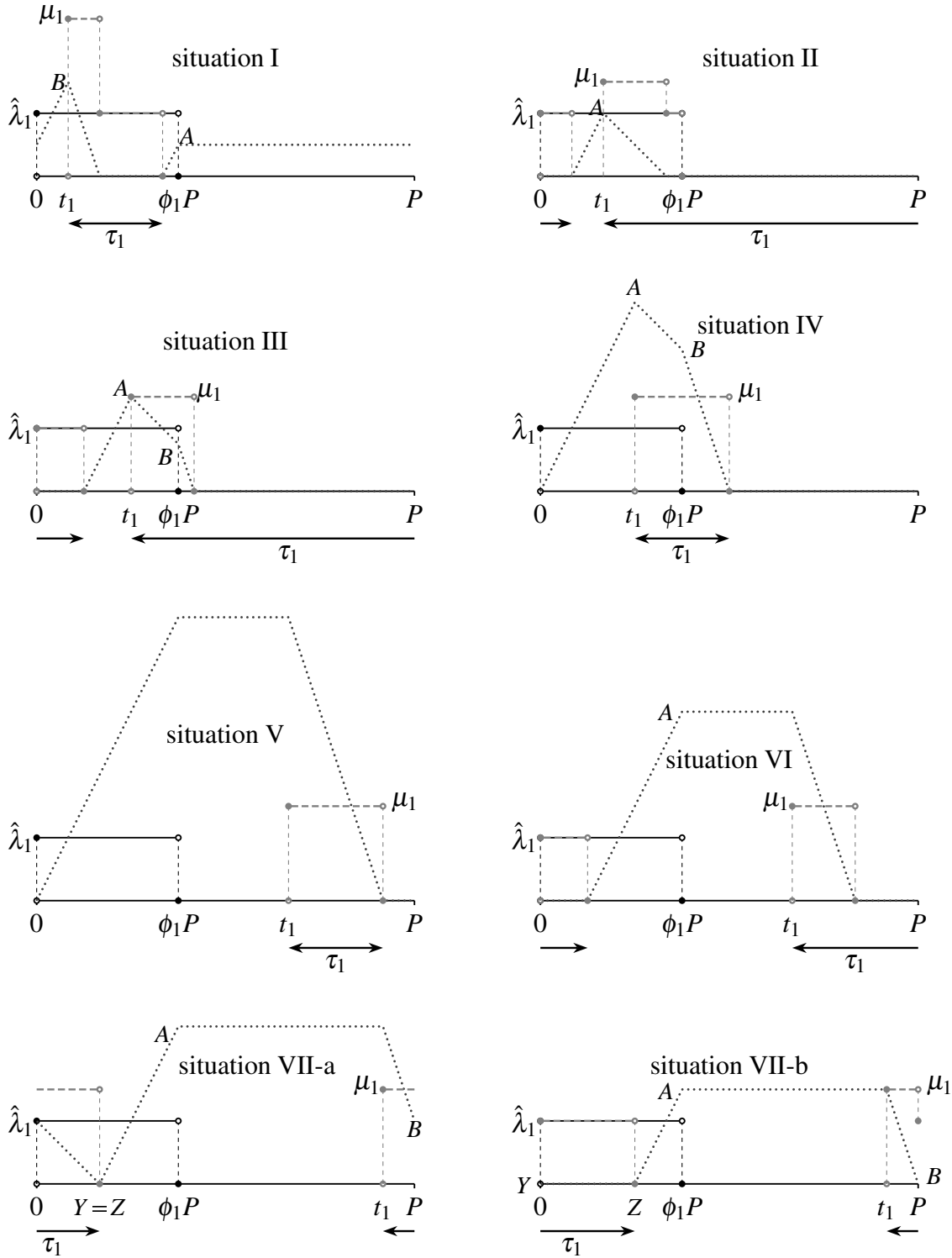


Figure 5.29: Different situations for type 1 and $\hat{\lambda}_1 < \mu_1$. Solid line: input rate profile. Gray dashed line: process rate profile. Dotted line: buffer level.

The mean wip level expressions as stated in Proposition 5.19 are a function of processing start time t_i and process interval length τ_i . In case $\hat{\lambda}_i \geq \mu_i$, the process interval length is fixed: $\bar{\rho}_i P$. From the derivation of the mean wip level expressions follows that different situations may occur. This causes the optimization recipe to be adapted slightly. Instead of solving two subproblems (due to the overlap constraints), optimization over all situations is required, after which the lowest solution of all subproblems is the overall solution. For example, if for both lot types the arrival rate $\hat{\lambda}$ is smaller than the maximum process rate μ , five different situations may occur for each lot type. All combinations are distinct subproblems, that in turn need to be solved with both overlap constraint sets separately. This results in 50 optimization subproblems. A number of these problems turns out to be infeasible due to the overlap constraints (it is not possible to process two lot types at the same time). The remaining subproblems are to be solved, after which the actual optimal process cycle can be determined by taking the ‘best’ solution of the subproblems.

The design variables of the optimization subproblems are t_1 , t_2 , τ_1 and τ_2 . The overlap constraint sets can be expressed explicitly in these design variables:

$$t_1 + \tau_1 + \sigma_{12} \leq t_2 \qquad t_2 + \tau_2 + \sigma_{21} \leq t_1 + P \qquad (5.24)$$

and the other constraint set is:

$$t_2 + \tau_2 + \sigma_{21} \leq t_1 \qquad t_1 + \tau_1 + \sigma_{12} \leq t_2 + P. \qquad (5.25)$$

The first overlap constraint set forces type 1 lots to be processed before type 2 lots. The inequality sign means that it is possible to have an idling period between processing type 1 and type 2 lots. The interpretation of the second overlap constraint set is similar to the first set, but this second set forces type 2 lots to be processed before type 1 lots.

A question that may arise is how these mean wip level expressions and optimizations relate to the work presented in Section 5.2 where lots with a constant inter-arrival time were considered for both lot types. Eventually, Section 5.2 has been a special case of the piece-wise constant arrival pattern, with the time fraction in which lots arrive set to one for both lot types. This leads to the following remark:

Remark 5.20. If the arrival patterns fractions ϕ_1 and ϕ_2 are set to 1, physically the original problem with constant arrival rates is re-obtained. Solving the optimization problem for the process interval lengths and period length results in the slow-mode condition of Theorem 5.10. A detailed proof of this statement is given in Appendix A.6.

5.8.3 Numerical example of an optimization problem for piecewise constant arrival rates

The recipe for determining optimal process cycles for a switching server and piecewise constant arrival rates has been treated in the previous section. For both lot types expressions for the

mean wip level have been derived and an optimization procedure regarding subproblems has been stated. To make the recipe more insightful, a numerical example is given in this section.

Consider the workstation as shown in the right hand side of Figure 5.30. All parameter values are indicated in the figure, and the arrival rate profiles have also been plotted. For type 1 lots, $\hat{\lambda}_1 < \mu_1$ and for type 2 lots, $\hat{\lambda}_2 > \mu_2$. The work in process levels have equal weight for both lot types: $c_1 = c_2 = 1$. With the general expressions for the mean wip level as proposed in Section 5.8.2, the following specific expressions for this numerical problem are found. For type 1 lots the expressions are rather complex since $\hat{\lambda}_1 \leq \mu_1$:

$$\bar{w}_1^I(t_1, \tau_1) = \frac{3}{16}\tau_1^2 - 3\tau_1 - \frac{3}{2}t_1 + 9 \quad (5.26a)$$

$$\text{for } 0 \leq t_1 \leq \frac{4}{3} \text{ and } \frac{8}{3} \leq \tau_1 \leq 4 - t_1$$

$$\bar{w}_1^{II}(t_1, \tau_1) = \frac{3}{16}(16 - \tau_1)^2 \quad (5.26b)$$

$$\text{for } \frac{4}{3} \leq t_1 \leq 4 \text{ and } 16 - t_1 \leq \tau_1 \leq 16 \text{ and } \tau_1 \geq \frac{8}{3} \text{ and } 2\tau_1 \geq t_1 + 28$$

$$\bar{w}_1^{III-VI}(t_1, \tau_1) = \frac{5}{48}\tau_1^2 - \frac{11}{3}\tau_1 - \frac{1}{48}t_1^2 - \frac{7}{6}t_1 + \frac{1}{12}\tau_1 t_1 + \frac{95}{3} \quad (5.26c)$$

$$\text{for } \frac{4}{3} \leq t_1 \leq 16 \text{ and } 2\tau_1 \leq t_1 + 28 \text{ and } \tau_1 \geq 16 - t_1$$

$$\text{and } 2\tau_1 \geq t_1 - 8 \text{ and } \tau_1 \leq 20 - t_1$$

$$\bar{w}_1^{IV-V}(t_1) = \frac{1}{2}t_1 - \frac{1}{3} \quad (5.26d)$$

$$\text{for } \frac{4}{3} \leq t_1 \leq \frac{40}{3} \text{ and } \tau_1 = \frac{8}{3}$$

$$\bar{w}_1^{VII}(t_1, \tau_1) = \frac{3}{16}\tau_1^2 - 3\tau_1 - \frac{3}{2}t_1 + 33 \quad (5.26e)$$

$$\text{for } \frac{40}{3} \leq t_1 \leq 16 \text{ and } \frac{8}{3} \leq \tau_1 \leq \frac{1}{2}t_1 - 4$$

and for type 2 lots, relatively simple expressions are obtained since $\hat{\lambda}_2 \geq \mu_2$:

$$\bar{w}_2(t_2) = 7 - \frac{5}{4}t_2 \text{ for } 0 \leq t_2 \leq 5 \text{ and } \tau_2 = 6 \quad (5.26f)$$

$$\bar{w}_2(t_2) = -\frac{1}{2} + \frac{1}{4}t_2 \text{ for } 5 \leq t_2 \leq 15 \text{ and } \tau_2 = 6 \quad (5.26g)$$

$$\bar{w}_2(t_2) = 27 - \frac{5}{4}t_2 \text{ for } 15 \leq t_2 \leq 16 \text{ and } \tau_2 = 6. \quad (5.26h)$$

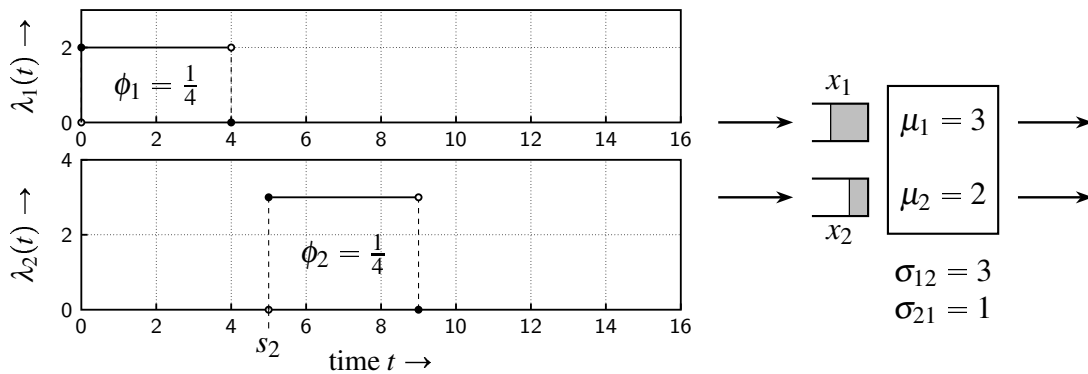


Figure 5.30: Numerical example for workstation with piece-wise constant arrival rates.

Five situations for type 1 lots exist and three situations for type 2 lots. All fifteen combinations form an optimization subproblem. Each subproblem itself yields two subproblems: the additional overlap constraints that couple the two lot types come in two forms. The first constraint set is:

$$t_1 + \tau_1 + \sigma_{12} \leq t_2 : \quad t_1 + \tau_1 + 3 \leq t_2 \quad (5.27a)$$

$$t_2 + \tau_2 + \sigma_{21} \leq t_1 + P : \quad t_2 + 7 \leq t_1 + 16 \quad (5.27b)$$

and the other constraint set is:

$$t_2 + \tau_2 + \sigma_{21} \leq t_1 : \quad t_2 + 7 \leq t_1 \quad (5.28a)$$

$$t_1 + \tau_1 + \sigma_{12} \leq t_2 + P : \quad t_1 + \tau_1 + 3 \leq t_2 + 16. \quad (5.28b)$$

These constraint sets couple the two lot types in a sense that they guarantee that the machine only serves one lot type at a time and that switching from lot type takes time. The first set computes the process cycle in which type 1 lots are processed earlier than type 2 lots and the second constraint set computes the process cycle the other way around. The total number of subproblems for this numerical example therefore is 30. All subproblems have been solved with MATLAB. Only nine of the subproblems turn out to be feasible. The solutions of the feasible subproblems are presented in Table 5.4. The first three columns in the table refer to the mean wip level expressions and the constraint set that were used for the subproblem. The optimal values for t_1 , τ_1 and t_2 are indicated with an asterisk. Some solutions are identical; they lie on the boundary of different situations and give the same result. Other solutions may look different but refer to the same physical solution, e.g. the first and last solution: since $P = 16$, the actual start time of processing type 1 lots is the same. These two solutions also have the minimal weighted mean wip level together with the third solution and therefore provide the solution to the overall problem (gray shaded rows). Note that the third solution looks different from the other two optimal solutions, but in fact the third solution starts with idling for type 1 lots (the length of processing lots at rate μ_1 is zero), followed by processing the lot types similar to the other two optimal solutions.

A physical interpretation of this solution is that the workstation processes the lots of type 1 completely in slow-mode, i.e. keeping the buffer empty all the time (in steady state). Due to the relatively long setup time σ_{12} , arrivals of type 2 lots have already started before the workstation processes those lots. The input and process rate profile, together with the buffer levels, are plotted in Figure 5.31.

A second optimization procedure is carried out with the numerical example. If the weights of the mean wip level are not equal, the optimal steady state cycle might differ. The following weighing factors are chosen: $c_1 = 1$ and $c_2 = 2$. Again, the thirty subproblems are solved. Of course, the same nine subproblems are feasible and the rest is infeasible (the constraints are not influenced by the weighing factors!). The solutions to the nine feasible subproblems are presented in Table 5.5. The wip level of type 2 lots (which can never be kept zero, since $\hat{\lambda}_1 \geq \mu_1$) is more important now. The area of the buffer level curve for type 2 lots is minimized now. This

Table 5.4: Solutions of the feasible subproblems of the numerical example ($c_1 = c_2 = 1$).

type 1	type 2	constraint set	t_1^*	τ_1^*	t_2^*	$c_1 \bar{w}_1 + c_2 \bar{w}_2$
(5.26a)	(5.26g)	(5.27)	0	4	7	$\frac{9}{4}$
(5.26c)	(5.26f)	(5.28)	12	6	5	$\frac{19}{6}$
(5.26c)	(5.26g)	(5.28)	14	6	7	$\frac{9}{4}$
(5.26d)	(5.26f)	(5.28)	12	$\frac{8}{3}$	5	$\frac{77}{12}$
(5.26d)	(5.26g)	(5.28)	12	$\frac{8}{3}$	5	$\frac{77}{12}$
(5.26d)	(5.26g)	(5.27)	$\frac{4}{3}$	$\frac{8}{3}$	7	$\frac{31}{12}$
(5.26d)	(5.26h)	(5.27)	7	$\frac{8}{3}$	16	$\frac{61}{6}$
(5.26e)	(5.26f)	(5.28)	$\frac{44}{3}$	$\frac{10}{3}$	5	$\frac{23}{6}$
(5.26e)	(5.26g)	(5.28)	16	4	7	$\frac{9}{4}$

means that processing type 2 lots starts as soon as the arrivals start, loosely speaking to keep the ‘damage’ the lowest. Due to setup time $\sigma_{12} = 3$, type 1 lots cannot all be processed in slow-mode anymore. The more lots are processed in slow-mode, the longer the buffer level remains zero and the less the buffer level of that type rises. After processing the ‘expensive’ type 2 lots ($c_2 = 2$), the buffer of type 1 is emptied as quickly as possible, after which the workstation idles until the arrivals of type 1 start again, processing them in slow-mode. This optimal solution (gray shaded in the table) is graphically shown in Figure 5.32.

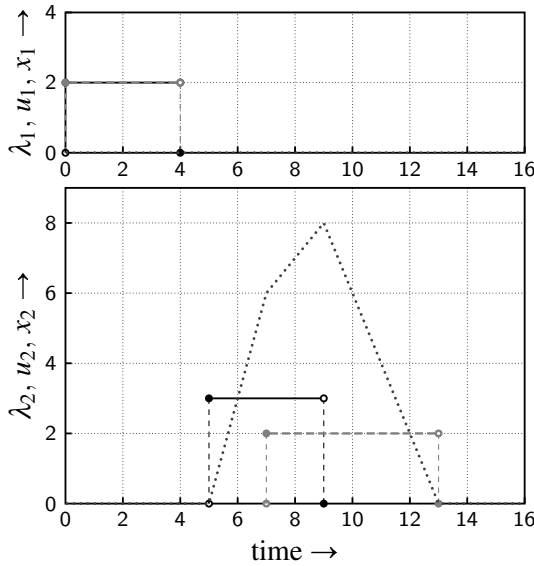
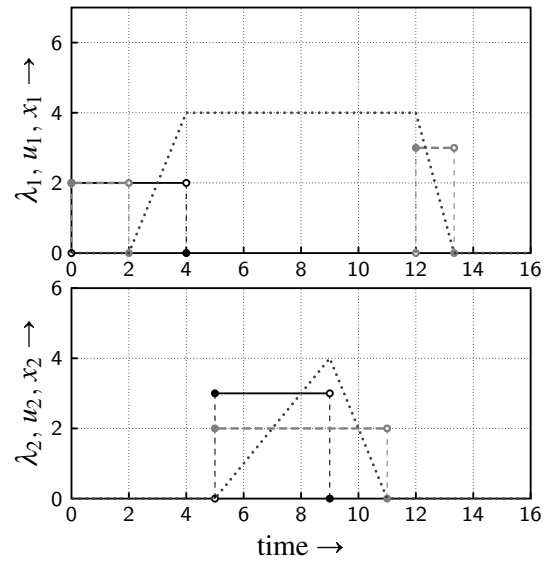
5.8.4 Remarks on optimal process cycles and feedback control

For piecewise constant arrival rates at a workstation, optimal process cycles have been defined. However, the analysis has only been made for two lot types and certain assumptions on the arrival rate profiles (see Section 5.8.1). For all situations that do not fit within the assumptions (e.g. unequal period lengths of the arrival processes, more than two piecewise constant levels, more than two lot types), optimal process cycles are still unknown. It is even unknown whether optimal solutions exhibit periodic behavior in these cases.

For multiple (more than two) lot types, a similar approach can be followed as laid out in this chapter. If the assumptions on the arrival rate profiles hold, then for each lot type the mean weighted wip level can be expressed as in Section 5.8.2. It is stressed that the number of subproblems that need to be solved increases (hyper)exponentially with the number of lot types. Even, when the assumptions on the arrival rate profiles are lifted or when the server is allowed to process a lot type more than one time within a period, the number of optimization subproblems

Table 5.5: Solutions of the feasible subproblems of the numerical example ($c_1 = 1$, $c_2 = 2$).

type 1	type 2	constraint set	t_1^*	τ_1^*	t_2^*	$c_1 \bar{w}_1 + c_2 \bar{w}_2$
(5.26a)	(5.26g)	(5.27)	0	4	7	$\frac{9}{2}$
(5.26c)	(5.26f)	(5.28)	12	6	5	$\frac{47}{12}$
(5.26c)	(5.26g)	(5.28)	12	6	5	$\frac{47}{12}$
(5.26d)	(5.26f)	(5.28)	12	$\frac{8}{3}$	5	$\frac{43}{6}$
(5.26d)	(5.26g)	(5.28)	12	$\frac{8}{3}$	5	$\frac{43}{6}$
(5.26d)	(5.26g)	(5.27)	$\frac{4}{3}$	$\frac{8}{3}$	7	$\frac{29}{6}$
(5.26d)	(5.26h)	(5.27)	7	$\frac{8}{3}$	16	$\frac{103}{6}$
(5.26e)	(5.26f)	(5.28)	$\frac{44}{3}$	$\frac{10}{3}$	5	$\frac{55}{12}$
(5.26e)	(5.26g)	(5.28)	16	4	7	$\frac{9}{2}$

**Figure 5.31:** Optimal steady state arrival/process rate profiles and buffer levels, $c_1 = c_2 = 1$. (Note that the buffer of type 1 stays empty.)**Figure 5.32:** Optimal steady state arrival/process rate profiles and buffer levels, $c_1 = 1$, $c_2 = 2$.

grows dramatically. Consequently, only for small problems (very low number of lot types) and within the assumptions, the proposed method is a practically tractable approach.

As for the feedback control problem with respect to the workstation with non-constant arrival rates, synchronization between the arrivals and processing of lots needs to be established. The start time of processing a certain lot type is related to the start time of the arrivals. Even if

an optimal process cycle implies only a clearing policy, unsynchronized clearing of buffers may not lead to this optimal process cycle, because of the ‘phase difference’ between arrivals and controller modes. The controller goal is therefore twofold: reducing the number of lots in the system and synchronizing with the arrival processes. A possible solution is to use similar controllers as in the situation with constant arrival rates, but with additional predicates for leaving controller modes. These predicates make sure that at the right time a switch takes place to the other job type. Synchronization is easy then. However, reducing the number of jobs in the system is not trivial anymore. Consider for instance two identical workstations processing at a pure bow tie process cycle. The arrival pattern for the second workstation is piecewise constant, since the first workstation switches between the job types. In theory, the buffers of the second workstation can be kept empty: the second workstation mimics the first workstation. But can a controller be found which achieves this mimic behavior while reducing the number of lots in the buffers of the second workstation? Only when enough ‘room’ exists to reduce the number of lots, this can actually be achieved. In other words: the process interval length needs to be larger than $\bar{\rho}_i P$ (the minimum amount of time needed to process all lots that arrive during a period) or some idling must be present before switching to the other type. Only then the workstation is able to serve more lots during a period than arriving. This requirement is necessary for all lot types. For the pure bow tie process cycle, the process interval length equals $\bar{\rho}_i P$, so different control strategies need to be found. In the next chapter, the optimization and control problem for two workstations in series is considered for some special cases.

If no room exists within a synchronized system to reduce the number of lots in the system, more advanced controllers need to be developed. An example could be enlarging the period length in the controller. This means that the relative amount of time that setups claim decreases, creating the necessary room to reduce the number of lots in the system. Then the period length of the controller can be decreased again to eventually synchronize with the arrival patterns.

5.9 Summary

In this chapter, a single workstation serving two product types has been analyzed extensively. A hybrid fluid model has been developed to describe the behavior of such a system. Based on this model, optimal process cycles with respect to time averaged weighted work in process levels have been determined for situations with and without maximum buffer capacities. Next, state feedback controllers were proposed. These controllers steer a trajectory to the desired (optimal) trajectory from an arbitrary starting point (buffer levels, time, mode of the workstation, etc.). Convergence to the desired trajectory has been proven mathematically.

The initial analysis was performed for a workstation where lots arrive at a constant arrival rate. In a manufacturing network setting however, this is not likely to be the case. Switching upstream servers cause the arrivals of lots to be non-constant. The arrival rate profile at a workstation shows piecewise constant behavior then. For a single switching server with two product types that arrive with periodic piecewise constant arrival rates, optimal process cycles

with respect to mean weighted work in process levels have been defined in the final part of this chapter. Analyzing all possible situations that may occur in this environment revealed that the number of situations grows rapidly. In a numerical example, thirty optimization subproblems had to be solved in order to obtain the global optimal solution. This number grows exponentially with the number of product types. Therefore, tractability issues rise when applying this theory to larger manufacturing systems. In practical (e.g. industrial) applications, one might not be interested in the *optimal* solution, but in a *better* solution than the current one. Instead of determining optimal process cycles, one will determine and prescribe *desired* process cycles, which may consist of a set of process cycles.

Nevertheless, the theory and analyses that have been performed in this chapter are not useless at all. Important insights have been obtained, for example the use of a slow-mode. During a slow-mode, lots are processed at their arrival rate, keeping the buffer of that type empty for a while. A slow-mode implies losing capacity due to processing at lower rates than the maximum rates. On the other hand, not using a slow-mode results in losing capacity due to relatively often switching in time. During switching between modes, no lots can be processed which can be regarded as losing capacity. In addition to the slow-mode, insight has been obtained about the shapes of optimal process cycles, the influence of buffer capacity constraints and the use of hybrid fluid models instead of a discrete event model.

The feedback controllers in this chapter have been implemented in discrete event simulations. Although the controllers had been developed using the hybrid fluid models, they perform well in a discrete event setting (as has been shown in the case studies), due to the feedback loop. Based on current measurements of the state, proper control actions are computed. The discrete / integer behavior is regarded as a disturbance with respect to the hybrid fluid model.

In the next chapter, the analysis of switching servers is extended towards flow lines. As mentioned earlier, the arrival rates of lots at workstations are in general not constant anymore. Switching causes a piecewise constant arrival pattern. For reasons just mentioned, Chapter 6 does not solve the overall general problem of finding an optimal process cycle. However, for some special situations, insightful results with respect to minimal work in process levels can be obtained.

Flow lines of switching servers

Switching servers have been studied in Chapter 5. For a certain class of servers (two product types, constant arrival rates, constant process rates and setup times) an optimal process cycle has been derived, for both unconstrained and constrained buffer capacities. Next, state feedback controllers were developed that steer a trajectory to the desired one, from an arbitrary starting point. In the final part of the previous chapter, the results had been extended to a class of switching servers with piecewise constant arrival rates.

This chapter elaborates on the results of Chapter 5 and studies flow lines of switching servers. The ideas and insights obtained in the previous chapter form the basis for the analyses that are performed here. In general, one could state that the number of lots in a manufacturing network is only determined by the workstations at which lots arrive and the workstations at which lots leave the system. For a flow line these are the most upstream and downstream workstation respectively. Throughout the complete chapter, it is assumed that lots arrive at a constant rate, so in fact the number of lots in the flowline (the total wip level) is only determined by the most downstream workstation: the other workstations only move work from the one workstation to the other workstation.

In general, optimal process cycles for networks of workstations, including flow lines, are currently unknown. However, for certain classes of flow lines, it is possible to find optimal process cycles. The common feature of the two classes that are examined in this chapter is that one of the workstations processes at its optimal process cycle as derived in the previous chapter and that the corresponding optimal work in process level is achieved for the complete flow line. The following classes are investigated:

- The first workstation in a flow line performs its stand alone optimal process cycle and all downstream workstations make sure that their buffers remain empty. This is elaborated in Section 6.2.1.
- The most downstream workstation in a flow line performs its optimal process cycle and upstream workstations make sure that this final workstation can actually do that. This is studied in Section 6.2.2.
- One of the middle workstations in the flowline performs its stand alone optimal process cycle and all other workstations make sure that the optimal wip level of this workstation is achieved for the flowline. Results of Sections 6.2.1 and 6.2.2 are combined then. This is also illustrated in Section 6.4.

For the first two classes of flow lines, optimal process cycles are derived and conditions for all workstations to achieve this optimal cycle are given. In addition, state feedback controllers are proposed that steer the trajectories of the workstations to the desired ones. Results are illustrated with case studies. The third class of flowlines can be treated with a combination of the first two classes. First, the characteristics and dynamics of switching server flow lines are given in Section 6.1, which is basically a recapitulation of Section 5.1 with extensions for a flow line.

6.1 Characteristics and dynamics of flow lines of switching servers

In this section the characteristics and dynamics of a switching server flow line are presented, based on a flow line with two workstations and serving two product types. The way of modelling is generic for multiple workstations and lot types.

Consider the flow line consisting of workstations A and B , each consisting of two parallel buffers and a switching server. The buffers have infinite capacity, store a specific lot type and the contents are denoted by $x_i^j(t)$, $i \in \{1, 2\}$, $j \in \{A, B\}$, e.g. $x_1^B(t)$ equals the number of lots of type 1 that are present in workstation B at time t . Lots arrive at workstation A with constant rates λ_1 and λ_2 for type 1 and type 2 lots respectively. The maximum process rates are μ_i^j . Switching from processing type 1 to type 2 lots takes σ_{12}^j time units and σ_{21}^j time units vice versa. The system is schematically shown in Figure 6.1.

For stability reasons, the total workload must not exceed 1 for each server: $\sum_i \rho_i^j < 1$ for all $j \in \{A, B\}$ with $\rho_i^j = \lambda_i / \mu_i^j$. Unless indicated otherwise, superscript $j \in \{A, B\}$ denotes the workstation number and subscript $i \in \{1, 2\}$ represents a lot type throughout the remainder of this chapter.

The state and input vector for this flow line are similar to those of the single switching server example. The state consists of the remaining setup times (x_0^j), the buffer levels (x_1^j and x_2^j) and

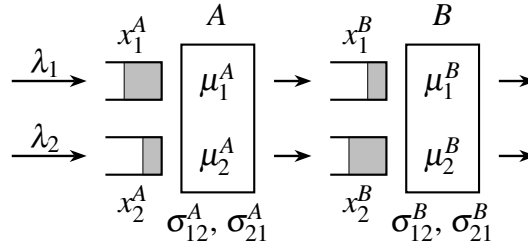


Figure 6.1: Flow line of two switching servers overview.

the modes (m^j) of the servers:

$$\mathbf{x} = \begin{bmatrix} x_0^A & x_0^B & x_1^A & x_2^A & x_1^B & x_2^B & m^A & m^B \end{bmatrix}^T \in [0, \max(\sigma_{21}^A, \sigma_{12}^A)] \times [0, \max(\sigma_{21}^B, \sigma_{12}^B)] \times \mathbb{R}_+^4 \times \{1, 2\}^2.$$

The input vector consists of the action that has to be performed by the servers (u_0^j) and the rates at which the servers are processing the lot types (u_1^j and u_2^j). Possible actions of the servers are:

$$\begin{aligned} u_0^j = \mathbf{1} &: \text{ setup server } j \text{ for type 1 lots} \\ u_0^j = \mathbf{1} &: \text{ server } j \text{ process type 1 lots} \\ u_0^j = \mathbf{2} &: \text{ setup server } j \text{ for type 2 lots} \\ u_0^j = \mathbf{2} &: \text{ server } j \text{ process type 2 lots} \end{aligned}$$

The input vector \mathbf{u} becomes:

$$\mathbf{u} = \begin{bmatrix} u_0^A & u_0^B & u_1^A & u_2^A & u_1^B & u_2^B \end{bmatrix}^T \in \{\mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{2}\}^2 \times [0, \mu_1^A] \times [0, \mu_2^A] \times [0, \mu_1^B] \times [0, \mu_2^B]$$

and similar to the single switching server, the inputs are constrained by the state:

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

Informally, these constraints mean the following:

- if a server is busy with a setup, no lots can be processed;
- after a setup to a lot type has been completed, only lots of that specific type can be processed;
- it is always possible to stay in the current mode, or to switch to the other mode.

The discrete event and continuous dynamics of the flow line look similar to the dynamics of the single switching server (5.3)–(5.4):

$$x_0^j(t) := \sigma_{21}^j, m^j(t) := 1 \text{ for } u_0^j(t) = \textcircled{1} \text{ and } m^j(t) = 2, j \in \{A, B\} \quad (6.1a)$$

$$x_0^j(t) := \sigma_{12}^j, m^j(t) := 2 \text{ for } u_0^j(t) = \textcircled{2} \text{ and } m^j(t) = 1, j \in \{A, B\} \quad (6.1b)$$

$$\dot{x}_0^j(t) = \begin{cases} -1 & \text{for } u_0^j(t) \in \{\textcircled{1}, \textcircled{2}\}, j \in \{A, B\} \\ 0 & \text{for } u_0^j(t) \in \{\textcircled{1}, \textcircled{2}\}, j \in \{A, B\} \end{cases} \quad (6.1c)$$

$$\dot{x}_i^A(t) = \lambda_i - u_i^A(t), i \in \{1, 2\} \quad (6.1d)$$

$$\dot{x}_i^B(t) = u_i^A(t) - u_i^B(t), i \in \{1, 2\}. \quad (6.1e)$$

The goal is to minimize the time averaged weighted work in process level of the flow line. The cost function J is defined as:

$$J = \limsup_{t \rightarrow \infty} \frac{1}{t} \int_0^t \left[g_1(x_1^A(s) + x_1^B(s)) + g_2(x_2^A(s) + x_2^B(s)) \right] ds \quad (6.2)$$

with $g_i : \mathbb{R}_+ \rightarrow \mathbb{R}_+$ strictly increasing functions. Note that it is assumed that work of a specific lot type has equal cost weights in both workstations.

An important observation is that the switching policy of workstation A does not affect the work in process level. Workstation A moves work from A to B , but work remains in the flow line. Workstation B actually removes work from the flow line. The switching policy of B therefore determines the wip level of the flow line. In general, when the arrival rate cannot be influenced, switches of the most downstream workstation determine the wip in the system.

From Section 5.2 the optimal process cycle of a single switching server is known. This is the process cycle which gives minimal mean work in process levels. Any other process cycle results in larger mean wip levels. Therefore, the optimal mean wip level for a single switching server is an absolute lower bound on the mean wip level for a flow line of switching servers. In this chapter it is investigated under which conditions it is possible to achieve this lower bound on the mean wip level for the complete flow line. The general idea is as follows: make one of the workstations perform its optimal cycle according to Section 5.2 and make sure that all downstream buffers remain empty and also make sure that all upstream workstations can make this particular workstation process at its optimal cycle. For the situation with two workstations two cases can be distinguished:

- Workstation A performs its optimal cycle and workstation B is able to keep its buffers empty. This is studied in Section 6.2.1.
- Workstation B performs its optimal cycle and workstation A is able to serve workstation B in such a way that B can actually perform at this optimal cycle. Section 6.2.2 investigates the necessary and sufficient conditions on workstation A .

For larger flow lines (more than two workstations), the aforementioned two situations can be combined. Consider for example a flow line consisting of three switching servers. Let the middle server perform its optimal process cycle, the upstream workstation facilitates this and the downstream server keeps its buffers empty. In that way the minimal mean wip level of the middle workstation is achieved for the complete flow line. In the next section, the two cases mentioned are studied in detail.

6.2 Process cycles and controllers for switching server flow line

The largest optimal mean wip level of all switching servers in a flow line regarded as stand alone switching servers is a lower bound for the mean wip level of that switching server flow line. Any other process cycle of that particular workstation results in higher wip levels and other workstations can only add more wip, since negative buffer levels are not allowed. In this section, two cases are elaborated. First, in Section 6.2.1, the situation is studied in which workstation *B* keeps its buffers empty. Necessary and sufficient conditions are derived. Secondly, in Section 6.2.2, necessary and sufficient conditions for workstation *A* are derived which have to be met in order to make the flow line behave (with respect to wip levels) as if it were only workstation *B*. For both classes, a feedback controller is proposed that steers the trajectory of the system to the desired behavior. Case studies illustrate the working of the controller and are used to compare the results with other control policies. In Section 6.3 some remarks are made on the general problem of finding an optimal process cycle for a flow line with two workstations (if existing at all, this is an open problem) and Section 6.4 focuses on using the earlier obtained results in the analysis of larger flow lines.

6.2.1 Optimal process cycle and feedback controller for restrictive upstream workstation

In this section an optimal process cycle is derived for a switching server flow line consisting of two workstations. Goal is to find the conditions which have to be met in order to make the optimal mean wip level of the upstream workstation the mean wip level for the flow line. After an optimal process cycle has been determined including the conditions, a state feedback controller is proposed that steers the system to the desired trajectory. Case studies that show results of the controller implementation conclude this section.

Conditions on downstream workstation

Consider again the flow line as presented in Figure 6.1. Assume that workstation *A* is to process at its optimal process cycle as derived in Section 5.2. The resulting minimal mean work in process level is to be established for the complete flow line. What are the conditions on

workstation B to achieve this?

Theorem 6.1. *In a switching server flow line consisting of two workstations serving two product types, the optimal mean weighted work in process level of the upstream workstation stand alone can be achieved for the complete flow line iff the downstream workstation is not slower than the upstream workstation, i.e. its process rates are not lower than the upstream workstation process rates and the setup times to a mode are not longer than the setup times to that mode of the upstream workstation.*

Proof. The minimal mean wip level of the upstream workstation can only be achieved for the complete flow line if the buffers of the downstream workstation can remain empty at all times. This implies that whenever A is processing lots, B must be processing the same lot type. It may never occur that B is busy with a setup, while A is processing lots. Therefore, $\sigma_{12}^B \leq \sigma_{12}^A$ and $\sigma_{21}^B \leq \sigma_{21}^A$. In addition, in order to keep the buffers in B empty, workstation B must be able to process lots at the same process rate as workstation A . The maximum process rates of B have to be greater than or equal to the maximum process rates of A : $\mu_1^B \geq \mu_1^A$ and $\mu_2^B \geq \mu_2^A$. This completes the necessity proof of the conditions. But are these conditions also sufficient?

Consider two workstation forming a flow line with $\sigma_{ij}^B \leq \sigma_{ij}^A$ and $\mu_i^B \geq \mu_i^A$. Is it always possible to find a feasible process cycle for workstation B to keep the buffer of that workstation empty? Assume that workstation B mimics workstation A with respect to the actual rates of processing lots: $u_i^B(t) = u_i^A(t) \forall t, i \in \{1, 2\}$. This is possible because of the assumption that setups in B do not take longer than in A and the maximum process rates of B are not exceeded. The mimicked process cycle in B ensures that its buffers remain at the same level during the complete cycle. Lemma 5.4 also holds here: this constant level can be zero, meaning that the buffers in workstation B can be kept empty: $x_i^B(t) = 0 \forall t$. Thus, the optimal mean wip level of workstation A indeed is the mean wip level for the complete flow line under the given conditions. \square

Example 6.2. Consider a flow line of two switching servers, A and B . The first workstation has the following characteristics: maximum process rates $\mu_1^A = 24$ lots/hour, $\mu_2^A = 27$ lots/hour and setup times $\sigma_{12}^A = \sigma_{21}^A = 2$ hours. At the upstream workstation, lots arrive at constant arrival rates: $\lambda_1 = 9$ and $\lambda_2 = 3$ lots/hour. The optimal mean work in process level for this workstation was determined in Section 5.5 and equals 32 lots. It is desired to achieve this mean wip level for the complete flow line. What are the required specifications for workstation B ? From Theorem 6.1 it is known that workstation B needs to be *at least as fast as* workstation A in both setup times and process rates. Therefore, workstation B is chosen to have the following specifications: $\mu_1^B = 30$ lots/hour, $\mu_2^B = 30$ lots/hour, $\sigma_{12}^B = \sigma_{21}^B = 1$ hour. The resulting periodic orbit, actual process rates and buffer levels are shown in Figure 6.2. Indeed, the graphs show that the periodic orbit of workstation A is obtained for the complete flow line (cf. Figure 5.12) and the buffers of workstation B remain empty. Note that for $t \in [1, 2)$ and $t \in [7, 8)$ server B is idling, or stated otherwise, performing a slow-mode with actual process rate zero.

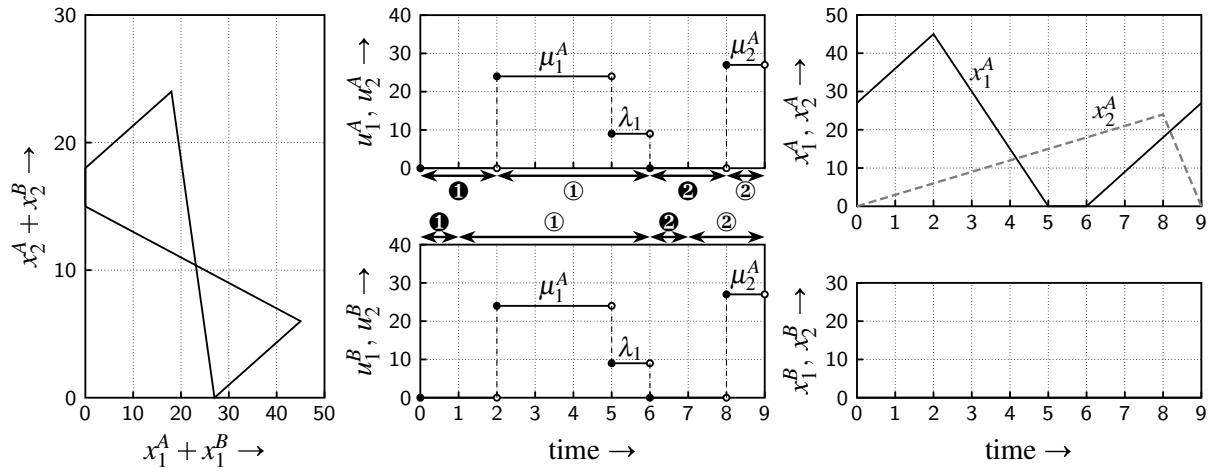


Figure 6.2: The optimal mean wip level for workstation A stand alone is achieved for the complete flow line A and B. Left: periodic orbit, center: process rates, right: buffer levels.

State feedback controller

Now that the conditions on workstation B are known to make the flow line (workstations A and B) behave as if it were only workstation A (with respect to work in process levels), a state feedback controller can be looked for that steers a system trajectory to the desired one, from an arbitrary starting point.

Without loss of generality, one may require to start the setups to a lot type at the same moment. See Figure 6.2 for an example of this. Another feasible solution would have been to idle 1 time unit longer in each mode in workstation B and make sure that the setup times end at the same moment. Requiring that the setups start synchronously simplifies the development of a feedback controller, since now in the optimal cycle $m^A(t) = m^B(t) \forall t$.

It is tempting to use the controller of the single switching server with unbounded buffer capacities (Proposition 5.13) and simply add the synchronous start of the setups in both workstations. Workstation B works ‘faster’ (Theorem 6.1) than A, so any amount of work in B is reduced over time until zero. However, when this feedback policy is applied to a server which is for at least one lot type identical to workstation A, i.e. equal process rate and equal setup time to that lot type, it is not guaranteed that the number of lots in B is reduced to zero. If workstation A reaches its optimal process cycle earlier than workstation B has emptied its buffer, the remaining buffer level is kept constant without reducing it any further. Therefore, the original feedback controller of Proposition 5.13 is used with the additional requirement that workstation A and B can only switch to the other lot type when both the buffer in A and the buffer in B are empty. This guarantees that after one process cycle the buffers in workstation B are empty and remain empty. This adaptation does not necessarily lead to optimal transients, but it is stressed that the original controller of Proposition 5.13 did not take into account optimal transients either. The aforementioned reasoning leads to the following proposition:

Proposition 6.3. *The following state feedback controller steers the system to the desired (optimal) periodic orbits, from any arbitrary initial state $\mathbf{x}(0)$:*

- *If at $t = 0$ the modes $\mathbf{m} = (m^A, m^B)$ of the machines are unequal in the initial state, then make B switch to the same mode as A.*
 - *After the initial control action (if necessary), the controller loops the following lines from top to bottom. Based on the state of the system, the controller (trivially) starts in one of the lines for each server.*
 - ① *at the highest actual possible rate in both workstations until $x_1^A = 0$, $x_1^B = 0$ and $x_2^A \geq x_2^{A\#}$;*
 - ② *in both workstations, after the setup time immediate ② in each workstation;*
 - ② *at the highest actual possible rate in both workstations until $x_2^A = 0$, $x_2^B = 0$ and $x_1^A \geq x_1^{A\#}$;*
 - ① *in both workstations, after the setup time immediate ① in each workstation.*
- in which $x_1^{A\#}$ and $x_2^{A\#}$ are given by the expressions in (5.10).*

Proof. See Appendix A.7. □

The feedback controller of Proposition 6.3 is validated by means of simulations. Both the original hybrid fluid model and discrete event models are used in the simulations to implement the feedback controller on. The controller is also compared with other control policies.

Case study: implementation of controller

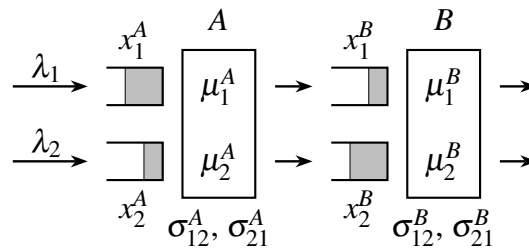


Figure 6.3: Flow line consisting of two switching servers.

For the situation in which the flow line achieves the mean wip level of the first workstation a state feedback controller has been developed, after the conditions on the second workstation have been defined. To validate proper working of the controller and to compare the results with other control policies, simulations are carried out. The following simulations are performed:

- Implementation of the controller of Proposition 6.3 on the original hybrid fluid model.
- Implementation of the controller of Proposition 6.3 on a deterministic discrete event model using χ .
- Implementation of the controller of Proposition 6.3 on a stochastic discrete event model using χ . Arrival rates and process rates are exponentially distributed with means λ_i

and μ_i^j respectively.

- Control of the flow line using local controllers, implemented on the hybrid fluid model.
- Open loop control with fixed process interval lengths, as proposed by Savkin [97–99].

Consider the flow line from Example 6.2, presented in Figure 6.3. All characteristics of the arrivals and workstation are given in Table 6.1. Note that the arrival rates and the upstream workstation are equal to the case study characteristics of Section 5.5. The threshold levels for switching to the other mode for workstation A are $x_1^{A\#} = 27$ and $x_2^{A\#} = 18$. Weighting functions g_1 and g_2 in the cost function (6.2) are assumed to be equal. Each simulation is carried out with the same initial conditions, which are also shown in the table.

Table 6.1: System parameters and initial conditions of case study.

λ_1 : 9 lots/hr.	μ_1^A : 24 lots/hr.	μ_1^B : 30 lots/hr.	$x_1^A(0)$: 20 lots	$x_0^A(0)$: 0 hrs.
λ_2 : 3 lots/hr.	μ_2^A : 27 lots/hr.	μ_2^B : 30 lots/hr.	$x_2^A(0)$: 20 lots	$x_0^B(0)$: 0 hrs.
	σ_{12}^A : 2 hrs.	σ_{12}^B : 1 hrs.	$x_1^B(0)$: 40 lots	$m^A(0)$: 1
	σ_{21}^A : 2 hrs.	σ_{21}^B : 1 hrs.	$x_2^B(0)$: 40 lots	$m^B(0)$: 1

The state feedback controller of Proposition 6.3 is implemented on the deterministic hybrid fluid model (6.1). It is expected that the buffers of workstation B are emptied during the first process cycle and remain empty. From that moment, the trajectory of workstation A converges to the truncated bow tie curve which is optimal for the given settings (cf. Section 5.5). Simulation results are shown in Figure 6.4. The figure shows that indeed the buffer levels of the second workstation decrease until zero. The figure also shows that when workstation A has emptied buffer 1 for the first time, it has to wait before setting up to the other lot type, although $x_2^A > x_2^{A\#}$. The setup is delayed because first the buffer of type 1 lots of the second workstation has to be empty, before the switchover takes place. The left hand graph shows that eventually the desired periodic orbit is reached for the lumped buffer levels of each type. This means that the mean wip level of workstation A stand alone has been achieved for the entire flow line.

A similar simulation is carried out on discrete event plant models. Both a deterministic and stochastic discrete event simulation are performed. The χ specification of the deterministic situation has been included in Appendix B.2, along with a few notes on the model and the implementation of the controller. Simulations are carried out with the same parameter settings and initial conditions, see Table 6.1. Results are shown in Figure 6.5 for the deterministic case and Figure 6.6 for the stochastic case (exponential distributions on inter-arrival times and process times with the same mean values as in the deterministic simulation). As expected, the integer-valued trajectories are steered to the desired (continuous) trajectory as good as possible. Due to the stochastics, workstation B is not able to keep its buffers empty anymore. However, the periodic orbit of the lumped buffer levels (left hand side of Figure 6.6) follows the desired periodic orbit quite well.

Another control policy with local controllers has been implemented. Workstation A uses the

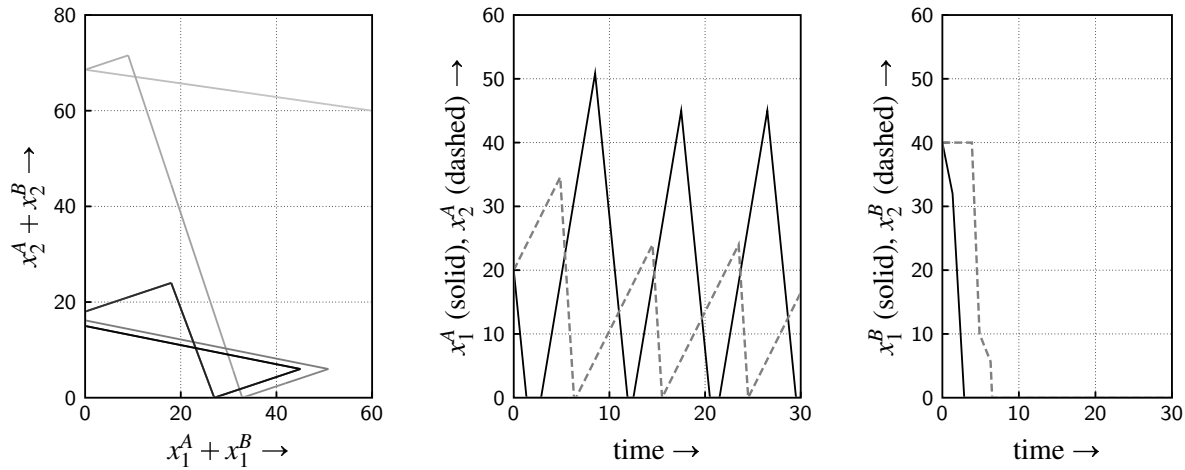


Figure 6.4: Simulation results for implementation of the state feedback controller of Proposition 6.3 on original hybrid fluid model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

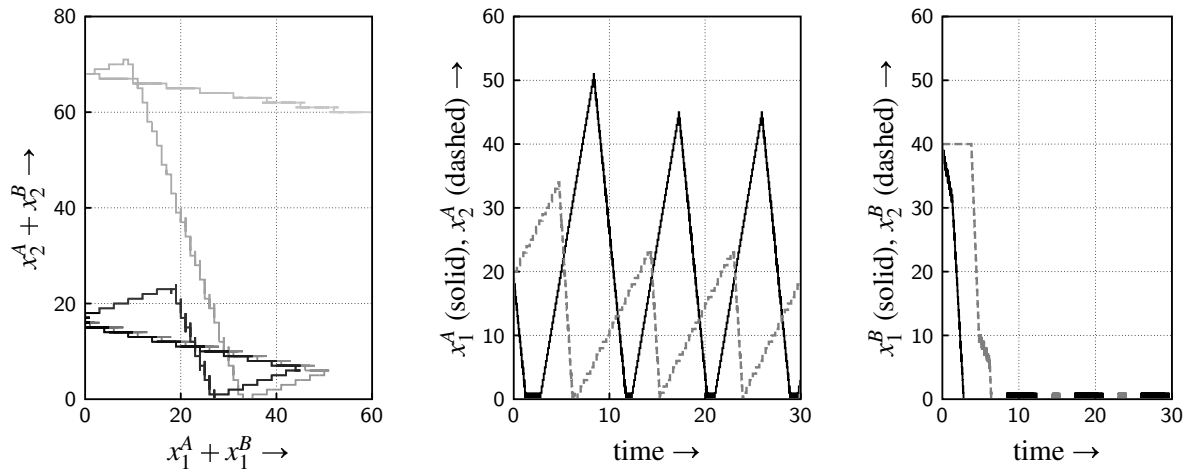


Figure 6.5: Simulation results for implementation of the state feedback controller of Proposition 6.3 on deterministic discrete event model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

original controller for a single switching server, as developed in Chapter 5, Proposition 5.13. Workstation B uses a clearing policy, i.e. empty a buffer at highest possible rate, then switch to the other lot type. The clearing policy is extended a little: when both buffers are empty and lots start to arrive at the buffer which is not served currently, a setup to that specific lot type takes place. It is assumed that in the first part of the simulation the buffers of B are emptied, because workstation B can work faster than A. Then B should be able to keep up with A. Workstation A is assumed to converge to its stand alone optimal process cycle, since it does not ‘feel’ what is going on downstream in the flow line. A simulation with the hybrid fluid model has been carried out, with the same initial conditions and parameters, as presented in

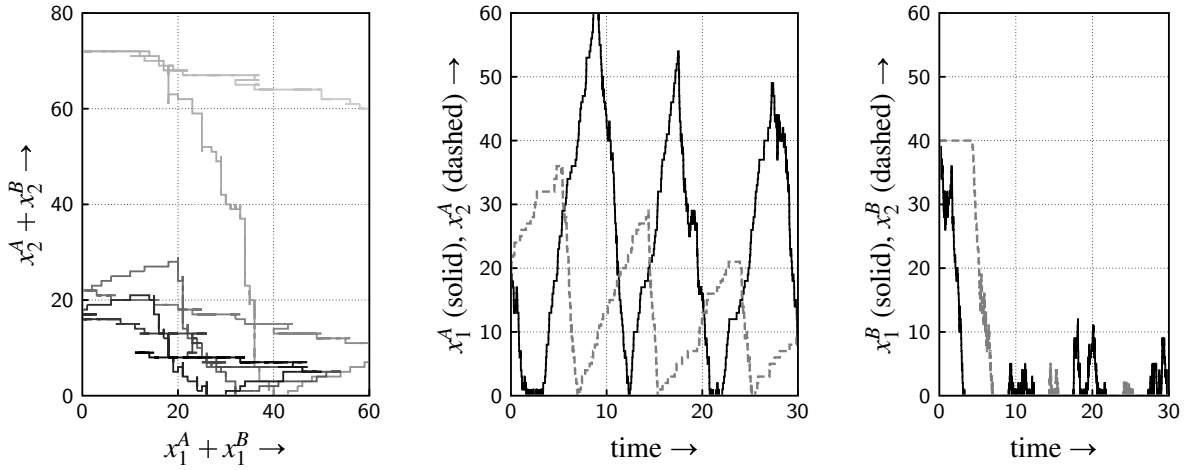


Figure 6.6: Simulation results for implementation of the state feedback controller of Proposition 6.3 on stochastic discrete event model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

Table 6.1. Simulation results are shown in Figure 6.7. As can be seen, the trajectory of B does not converge to the desired one, while the trajectory of A converges to the desired one, as expected. Buffers in workstation B are not kept empty. The reason for this is as follows: when both buffers are empty and workstation B is performing a slow-mode for a lot type, it can only switch to the other lot type when jobs start to arrive. During the setup time lots are stored in the buffer of workstation B , which are processed after the setup time. This causes an increase in the mean wip level of the flow line. Based on the characteristics of the workstations and the initial conditions, the increase of the buffer level in B takes place in at least one of the job types. It can be concluded that only local control does not settle the trajectory down to the desired periodic orbit. Note that different initial conditions may lead to different steady state periodic orbits, which is also not desirable.

Finally, a feed forward (open loop) controller is implemented on the hybrid fluid model. Such controllers have been proposed by Savkin in [97–99]. For the single switching server, the controller was implemented in Section 5.5 of this thesis. The controller uses fixed time spans in which the machine resides in a specific mode. The time spans for this example can be read from Figure 6.2: $\tau_1^A = 4$ and $\tau_2^A = 1$ for workstation A and for workstation B : $\tau_1^B = 5$ and $\tau_2^B = 2$. It is assumed that switching to a specific lot type is synchronized. The expected outcome is that the buffers of workstation B can be kept empty now after the transient period, because switching is synchronized and B works faster than A . On the other hand, the fixed process intervals might lead to unsatisfactory results for workstation A this time, as it did for the single switching server situation. The results of the simulation are shown in Figure 6.8. The buffers in workstation B indeed reach zero, although it takes longer than in the first experiment: the workstation has to wait until it can process type 2 lots for the second time, this is only allowed after $t = 14$. Workstation A does not reach its desired trajectory. Because it is only allowed to serve type 2 jobs for 1 hour during a cycle, it is never able to reduce the number of type 2 lots in the system.

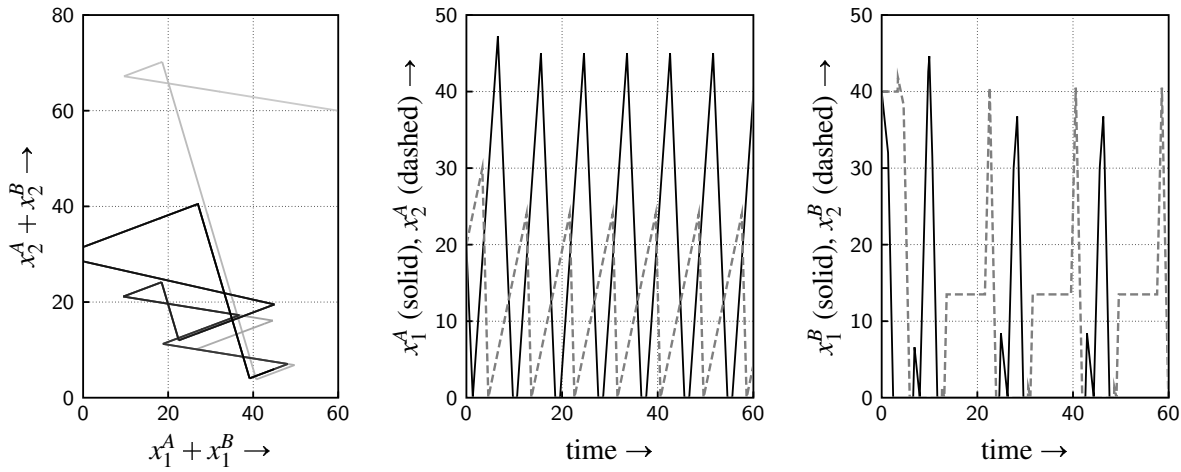


Figure 6.7: Simulation with only local controllers: workstation A with the controller of Proposition 5.13 and workstation B with clearing policy.

This is the same problem as encountered in Section 5.5. It must be noted that this controller, though not with state feedback, stabilizes the system in the sense that buffer levels do not explode over time. It should be noted that also in this case different initial conditions lead to different steady state solutions. Comparing the mean wip level of this specific solution with the wip level that results from the proposed controller (Proposition 6.3) is therefore not useful.

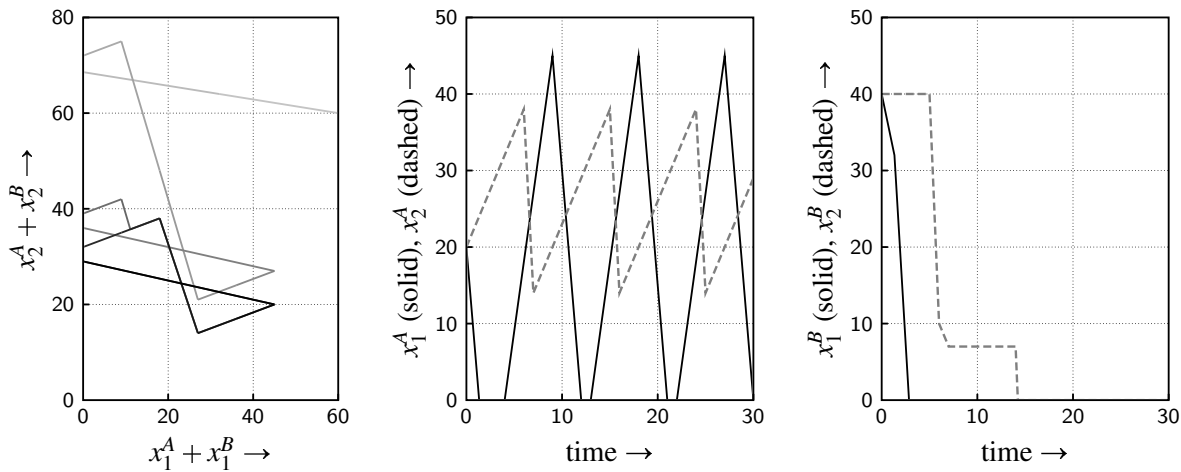


Figure 6.8: Simulation with feed forward (open loop) controllers: switching takes place according to a fixed time schedule.

Performance measures flow time and mean wip level have been measured for the implementation of the controller of Proposition 6.3. The results are shown in Table 6.2. Results of the other two controllers (local controller and open loop controller) have not been included, since different initial conditions lead to different trajectories and thus a different performance. In the table, means and standard deviations are shown for the stochastic simulations. The simulations have been carried out twenty times. In each simulation, a transient period of 30 process cycles

Table 6.2: Performance comparison for different simulations with a switching server flow line, processing two lot types. Control objective is to make the flow line of workstations *A* and *B* behave as if it were workstation *A* in isolation.

simulation	lot type	mean wip level	mean flow time
hybrid fluid model	1	20	$\frac{20}{9} \approx 2.22$
	2	12	4
	1+2	32	$\frac{8}{3} \approx 2.67$
deterministic discrete event	1	21.24	2.36
	2	12.10	4.03
	1+2	33.34	2.78
stochastic discrete event (20 simulations over 100 cycles)	1	22.73 ± 0.39	2.53 ± 0.043
	2	12.47 ± 0.17	4.16 ± 0.057
	1+2	35.19 ± 0.41	2.93 ± 0.034

was taken into account, after which the measurements started. The mean flow time of lots were determined after 100 complete process cycles. The mean wip levels have been computed using Little's law.

For a flow line of two switching servers, the situation has been described in which the first workstation performs its optimal process cycle and the second workstation makes sure that the corresponding work in process level is realized for the whole flow line. Conditions on the second workstation were relatively simple: it must not be slower than the first workstation for each lot type. This requirement concerns both setup times and process rates. A state feedback controller has been developed, which was proven to make the trajectory of a system converge to the desired trajectory. The controller has been tested in both discrete event and hybrid fluid model simulations. In the next section, the counterpart of this situation is examined: what are the conditions on the first workstation to let the second workstation perform its optimal cycle and establish that wip level for the complete flow line?

6.2.2 Optimal process cycle and feedback controller for restrictive downstream workstation

The general form of an optimal process cycle for a single switching server has been presented in Figure 5.5 and was made explicit in Theorem 5.10. The corresponding optimal work in process level is a lower bound for the wip level of a flow line with the same input rates of lots, since any other cycle of the server results in higher wip levels. In addition, other servers cannot reduce the mean wip level, since negative buffer levels are not allowed. In this section, conditions on the upstream workstation are derived to make the flow line behave as if it were only the downstream workstation with respect to work in process levels. With these conditions, a feasible process cycle for the upstream workstation can be determined. Once the process cycles of both workstations are known, a state feedback controller is proposed that stabilizes the system's trajectory to the desired one.

Conditions on upstream workstation

If it is possible to have the most downstream workstation process at its optimal cycle and have the other workstation make this possible, then optimal behavior for the complete flow line has been achieved. The buffer levels of a lot type can then virtually be added. For these lumped buffer levels, the optimal cycle as determined in Chapter 5 must be performed. The switching policy of the upstream workstations must then accommodate these virtually lumped buffer levels. This idea is schematically shown in Figure 6.9 (cf. Figure 5.1).

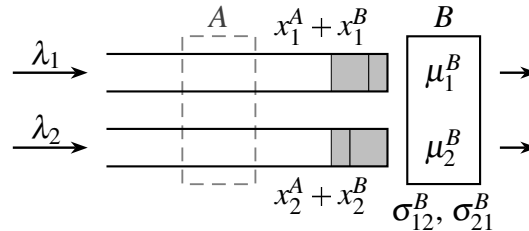


Figure 6.9: General idea of flow line behaving as single switching server.

The derived optimal cycle for a single switching server must be performed by workstation B and must also become the optimal cycle with respect to wip levels for the flow line. An important assumption is that the period of one process cycle in A equals the period of one process cycle in workstation B . This period is denoted by T . In Section 6.3 this assumption is further studied. If workstation A has to switch between lot types in such a way that it makes the flow line of workstation A and B behave like B stand alone, then some observations can be made:

1. If workstation B processes lots in slow-mode, i.e. its buffer is empty ($x_i^B = 0$) and it processes lots at the arrival rate, then workstation A also processes lots in slow-mode, because in an optimal trajectory the wip level of that type is zero for the whole flow line. Consequently, slow-modes in A should completely overlap slow-modes in B , if occurring

(see the conditions in Section 5.2). Define θ_i^- and θ_i^+ as the amount of time a slow-mode of lot type i in A starts earlier and ends later (respectively) than the corresponding slow-mode in B :

$$\theta_i^- + \tau_i^{\lambda B} + \theta_i^+ = \tau_i^{\lambda A}, \quad i \in \{1, 2\} \quad (6.3)$$

in which $\tau_i^{\lambda j}$ is the duration of the slow-mode in type i at workstation j . The overlap requirement yields:

$$\theta_1^- \geq 0; \quad \theta_1^+ \geq 0; \quad \theta_2^- \geq 0; \quad \theta_2^+ \geq 0. \quad (6.4)$$

2. Without loss of generality, let time $t = 0$ be the start of σ_{21}^B . From observation 1 it follows that at $t = 0$, A starts with a setup (if $\theta_2^+ = 0$) or is still in slow-mode of type 2 lots (if $\theta_2^+ > 0$). Therefore, σ_{21}^A starts at $t \geq 0$. Similarly, σ_{12}^A cannot start earlier than the start of σ_{12}^B .
3. Let $\tau_i^{\mu j}$ denote the duration of processing lots of type i at maximum process rate μ_i at workstation j . From observations 1–2 follows:

$$\theta_2^+ + \sigma_{21}^A + \tau_1^{\mu A} + \theta_1^- = \sigma_{21}^B + \tau_1^{\mu B} \quad (6.5a)$$

$$\theta_1^+ + \sigma_{12}^A + \tau_2^{\mu A} + \theta_2^- = \sigma_{12}^B + \tau_2^{\mu B}. \quad (6.5b)$$

4. Buffer levels are not allowed to become negative. Therefore, if $\tau_i^{\mu B}$ starts earlier than $\tau_i^{\mu A}$, the number of lots B processes before $\tau_i^{\mu A}$ starts may not exceed the number of lots A processes (in slow-mode) after B switched to the other mode:

$$\mu_i^B (\tau_i^{\mu B} - \theta_i^- - \tau_i^{\mu A}) \leq \lambda_i \theta_i^+ \quad (6.6)$$

or written differently:

$$\sigma_{21}^A + \theta_2^+ - \sigma_{21}^B \leq \rho_1^B \theta_1^+ \quad (6.7a)$$

$$\sigma_{12}^A + \theta_1^+ - \sigma_{12}^B \leq \rho_2^B \theta_2^+. \quad (6.7b)$$

This restriction is also valid if $\tau_i^{\mu B}$ starts after $\tau_i^{\mu A}$ started, since then the left-hand sides of (6.6)–(6.7b) become negative, while the right-hand sides are always positive. Therefore, these constraints may always be required.

5. The amount of lots A processes of each type during one cycle must be equal to the number of lots that is processed by B in one cycle. These mass conservation equations follow (with $i \in \{1, 2\}$):

$$\mu_i^A \tau_i^{\mu A} + \lambda_i \tau_i^{\lambda A} = \mu_i^B \tau_i^{\mu B} + \lambda_i \tau_i^{\lambda B} = \lambda_i T. \quad (6.8)$$

These observations have been summarized in Figure 6.10. The process cycles for workstations A and B are presented. Note that the time line for B is the same as in Figure 5.5. For reasons of symmetry and without loss of generality, the process cycle of B may contain slow-modes for both lot types, though it is known from Chapter 5 that in an optimal cycle the slow-mode takes place in at most one lot type. In addition, a slow-mode was added to both lot types

in workstation B for future extensions to larger flow lines. The overlapping slow-modes (observation 1) are clearly visible in Figure 6.10. Note that the situation which observation 4 refers to is visible for type 1 lots in the figure: the amount of lots workstation A processes during θ_1^+ must at least equal the number of lots that are processed by B before $\tau_1^{\mu A}$ starts again.

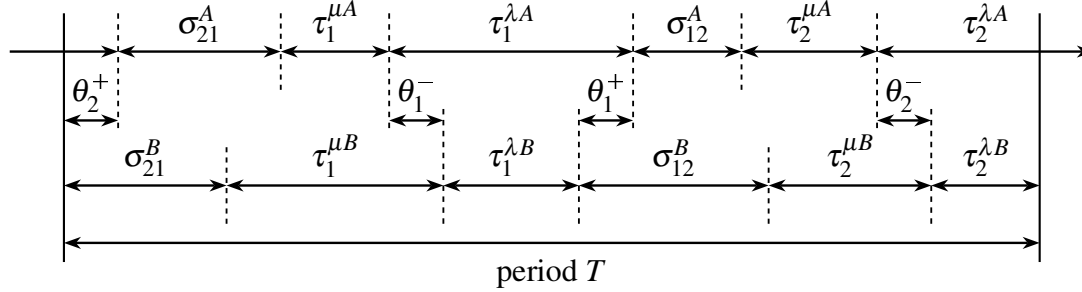


Figure 6.10: The period of one cycle, T , divided into subsequent phases.

With the given observations visualized in Figure 6.10 it is possible to derive conditions for server A which must be obeyed to make the flow line behave like B stand-alone with respect to work in process levels.

Remark 6.4. Observations 1–5 are also applicable for flow lines with more than two servers, e.g. a flow line with workstation A , B and C . In that case, first B has to make workstations B and C behave like C stand-alone and secondly, find a feasible trajectory for A to make workstations A and B behave like B stand-alone. Similar reasoning goes for larger flow lines. In Section 6.4 this is elaborated for four workstations in a flow line, where an optimal process cycle is determined all at once by means of solving a linear program.

Theorem 6.5. *Workstation A can make flow line of workstations A and B perform like B stand-alone with respect to work in process levels if and only if:*

$$R_2 \left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A \geq 0 \quad (6.9a)$$

and

$$R_1 \left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A \geq 0 \quad (6.9b)$$

with $R_1 = \max(\rho_1^A, \rho_1^B)$ and $R_2 = \max(\rho_2^A, \rho_2^B)$, the highest partial workloads of a lot type over the workstations.

Proof. See Appendix A.8. □

When more than one upstream workstation is present, similar conditions can be derived for these workstations. Each additional workstation adds two constraints similar to (6.9) to Theorem 6.5. Notice, however, that for larger flow lines, checking the conditions for all workstations in upstream direction might involve some iterations. In general, some freedom exists in the choice of process interval lengths within all conditions as described in the observations earlier

in this section. The one choice may give feasible results for other upstream servers, whereas the other choice might give infeasible results. Eliminating this freedom and developing explicit relations is not investigated in this thesis. To avoid this issue, a way to find feasible trajectories for upstream servers all at once given the parameters (μ and σ) is to cast the problem into a linear program with design variables $\tau_1^\mu, \tau_1^\lambda, \tau_2^\mu, \tau_2^\lambda$, and slow-mode extensions θ_1^+ and θ_2^+ for all servers. All constraints (6.3)–(6.8) are linear in the design variables. Any arbitrary objective function results in a feasible solution, (unless the problem is infeasible according to (6.9) and corresponding conditions for other workstations). In this way, for larger flow lines, feasible trajectories can be found relatively easy. The linear program solver can also be used to check if a feasible solution exists at all. An example of this linear programming approach for longer manufacturing flow lines is given in Section 6.4

The desired (optimal) process cycle has been defined for the entire flow line now and conditions for the upstream workstations have been derived. Now a controller that steers the state $\mathbf{x}(t)$ to the desired periodic orbits from any arbitrary initial state $\mathbf{x}(0)$ is developed.

State feedback controller

A state feedback controller that brings any arbitrary trajectory to the desired periodic orbits as defined in Sections 5.2 and 6.2 is presented in this section. The controller can be obtained using the ideas presented by Lefeber and Rooda [73].

Proposition 6.6. *The following state feedback controller steers the system to the desired (optimal) periodic orbits, from any arbitrary initial state $\mathbf{x}(0)$:*

- If at $t = 0$ the modes $\mathbf{m} = (m^A, m^B)$ of the machines are unequal in the initial state, then make A switch to the same mode as B.
- After initial switching (if necessary), each workstation separately loops the following lines from top to bottom. Based on the state of the system, the controller (trivially) starts in one of the lines for each server. Note that at some lines the workstations synchronize before a setup takes place. Lots are always processed at the actual highest possible rate.

Workstation A:	Workstation B:
① until $x_1^A = x_1^B = 0$	① until $x_1^B = x_1^A = 0$
① until $x_1^B \geq x_1^{B\#}$ and $m^B = 2$	① until $x_2^A \geq x_2^{A\#}$
perform ②	perform ②
② until $x_2^A = x_2^B = 0$	② until $x_2^B = x_2^A = 0$
② until $x_2^B \geq x_2^{B\#}$ and $m^B = 1$	② until $x_1^A \geq x_1^{A\#}$
perform ①	perform ①

Note that the servers always process at the current highest possible rate, obeying Lemma 5.2.

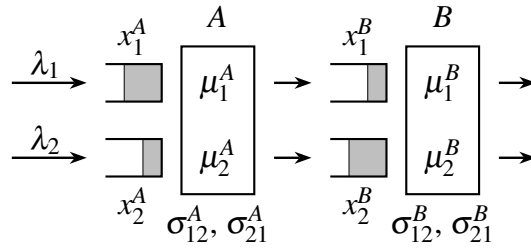
Proof. See Appendix A.9. □

Table 6.3: System parameters of flow line case study with two workstations.

λ_1 : 4 lots/hr.	μ_1^A : 20 lots/hr.	μ_1^B : 20 lots/hr.	$m^A(0)$: 1	$x_1^A(0)$: 25 lots
λ_2 : 4 lots/hr.	μ_2^A : 40 lots/hr.	μ_2^B : 20 lots/hr.	$m^B(0)$: 1	$x_2^A(0)$: 25 lots
	σ_{12}^A : 0.5 hrs.	σ_{12}^B : 5.0 hrs.	$x_0^A(0)$: 0 hrs.	$x_1^B(0)$: 25 lots
	σ_{21}^A : 1.0 hrs.	σ_{21}^B : 0.5 hrs.	$x_0^B(0)$: 0 hrs.	$x_2^B(0)$: 25 lots

Convergence to the desired steady state process cycle has been proven mathematically. Next, simulations are carried out in which the controller of Proposition 6.6 is implemented on the hybrid fluid model, a deterministic discrete event simulation and a stochastic discrete event simulation. In addition, the performance of the controller is compared with the performance of a controller in which only local state information of a workstation is available to determine the control action for that workstation.

Case study: controller implementation

**Figure 6.11:** Flow line consisting of two switching servers.

Consider again the switching server flow line consisting of two workstations, processing two lot types. A schematic overview of this flow line is given in Figure 6.11. The buffers have infinite storage capacity and the system parameters are as given in Table 6.3. First, all parameters are constant. Later on, some parameters are subject to variability, to check the working of the feedback controller under disturbing circumstances. It is assumed that wip in the buffers has equal weight in the cost function for both lot types. The state feedback controller of Proposition 6.6 is implemented in a few simulations:

- the original hybrid fluid model;
- a deterministic discrete event simulation using χ ;
- a stochastic discrete event simulation using χ in which the inter arrival times and process times of lots are distributed exponentially, with means $1/\lambda_i$ and $1/\mu_i^j$ respectively.

In addition to these simulations, the performance of the controller of Proposition 6.6 is compared with a control strategy which uses only local information of a workstation to determine the control action of that workstation, i.e. the workstations are not aware of each other's existence.

The determined optimal process cycle of workstation B does not contain a slow-mode (cf. Section 5.2), so the periodic orbit has the pure bow tie shape (see Figure 5.7). The optimal process cycle can be characterized as follows: $\tau_1^{\mu B} = 1\frac{5}{6}$ hours, $\tau_2^{\mu B} = 1\frac{5}{6}$ hours, resulting in a total period length of $9\frac{1}{6}$ hours. The threshold levels for switching to the other lot type are $x_2^\# = 9\frac{1}{3}$ and $x_1^\# = 27\frac{1}{3}$ lots. The mean number of lots in the system is $29\frac{1}{3}$, which is $14\frac{2}{3}$ per lot type (symmetric workstation with respect to process rates and arrival rates). Using Little's law, the mean flow time of both lot types is $3\frac{2}{3}$ hours.

For workstation A , first the conditions of (6.9) are checked and they are fulfilled. The process cycle of A is computed in the way as described in the second part of the proof of Theorem 6.5. The process cycle of workstation A has the following characteristics: $\tau_1^{\mu A} = \frac{11}{12}$ hours, $\tau_1^{\lambda A} = 4\frac{7}{12}$ hours, $\tau_2^{\mu A} = \frac{7}{9}$ hours, $\tau_2^{\lambda A} = 1\frac{7}{18}$ hours, $\theta_1^+ = 4\frac{7}{12}$ hours and $\theta_2^+ = \frac{5}{12}$ hours. Note that the process cycle of workstation A contains a slow-mode in both lot types and the period length of the process cycle of workstation A equals (by requirement) the period length of the cycle performed by workstation B : $9\frac{1}{6}$ hours.

For the controller implementation, the values of the buffers at which setup to the other lot type takes place have to be computed. For the values of slow-mode extensions θ_1^+ and θ_2^+ the largest possible values are chosen (see Appendix A.8 for details). This yields the following threshold values for switching: $x_1^{A\#} = 9$, $x_2^{A\#} = 7\frac{2}{3}$, $x_1^{B\#} = 18\frac{1}{3}$ and $x_2^{B\#} = 1\frac{2}{3}$.

First, a simulation with the original hybrid fluid model (with constant arrival and process rates) and the controller has been carried out (with Matlab). Initial conditions for this simulation are presented in Table 6.3. Simulation results are shown in Figure 6.12. In the left hand side graph, the lumped buffer levels are plotted against each other. Clearly, the threshold buffer levels for switching to the other type in case of stand alone workstation B are visible: $x_2^\# = 9\frac{1}{3}$ and $x_1^\# = 27\frac{1}{3}$ lots. These threshold values are now achieved for the flow line of workstations A and B . The middle and right hand side graph show the individual buffer levels over time.

In the discrete event simulation, the fluid model is abandoned, i.e. buffer levels only take on integer (natural) values and processing lots takes a real amount of time, contrary to the fluid model approximation. Two different discrete event simulations are performed: a completely deterministic simulation and a stochastic simulation. In the stochastic simulation, the inter-arrival times of lots and process times of lots are chosen to be exponentially distributed. Setup times remain constant. The variability on the arrival and process pattern and the integer character of the buffer levels can be regarded as disturbances with respect to the original hybrid fluid model. This hybrid fluid model was used to develop the feedback controller. Therefore it is interesting to see how the controller behaves when it is implemented on other plant models. Similar to the results obtained in previous discrete event simulations, it is expected that the feedback controller settles the trajectories of the systems down to the desired trajectories. Results of the discrete event simulations (with the same initial conditions as in the previous simulation) are shown in Figures 6.13 and 6.14. Compared with the results of the hybrid fluid model simulation, one can conclude that the controller is very well capable to steer the trajectories of the

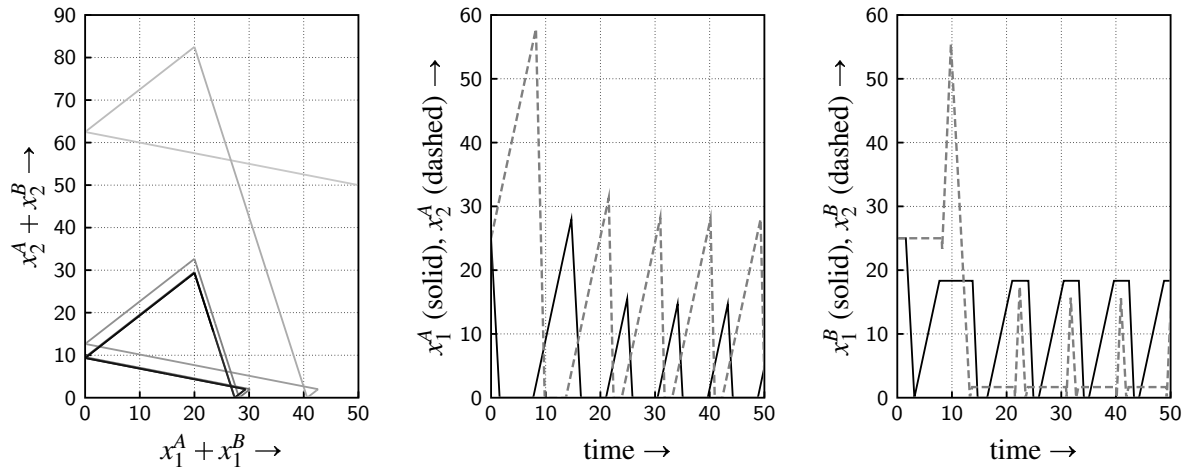


Figure 6.12: Simulation results for implementation of the state feedback controller (Proposition 6.6) on original hybrid fluid model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

discrete event simulations to the desired trajectories.

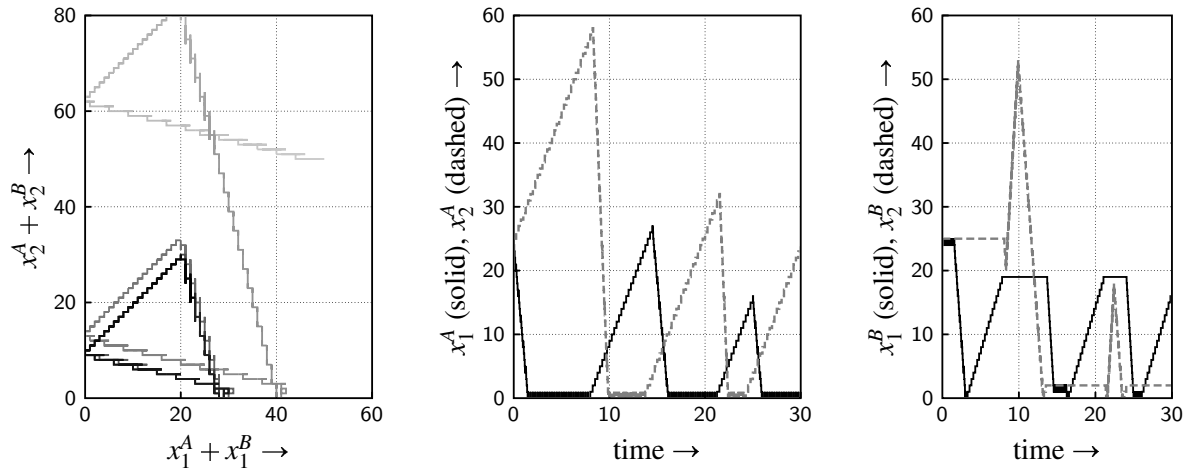


Figure 6.13: Simulation results for implementation of the state feedback controller (Proposition 6.6) on deterministic discrete event model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

Another control policy is implemented on the hybrid fluid model. If the workstations do not know the state information of each other, local information must result in control actions. Assume that only local information is available about the mode of the server and the buffer levels of that server. Both workstations can implement a clearing policy with double threshold switching levels. The switching levels can be read from Figure 6.12 (or from the numerical simulation data). These levels seem to give optimal system behavior, but does it also work with local controllers? The threshold levels are set to: $x_1^{A\#} = 10\frac{2}{3}$, $x_2^{A\#} = 26$, $x_1^{B\#} = 18\frac{1}{3}$ and $x_2^{B\#} = 1\frac{2}{3}$. Simulation results are shown in Figure 6.15. The same initial conditions were used as in the first

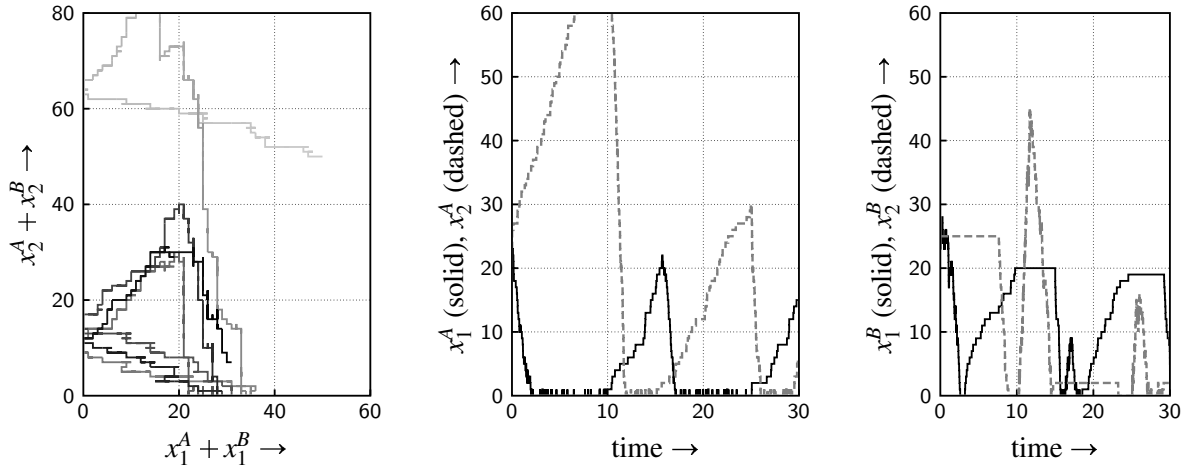


Figure 6.14: Simulation results for implementation of the state feedback controller (Proposition 6.6) on stochastic discrete event model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

simulation. As can be seen, the trajectory does not settle down to the optimal steady state periodic orbit. The synchronization between the workstations is crucial when steering the system to its optimal steady state orbit. Note that different initial conditions lead to the same periodic orbit as shown in Figure 6.15.

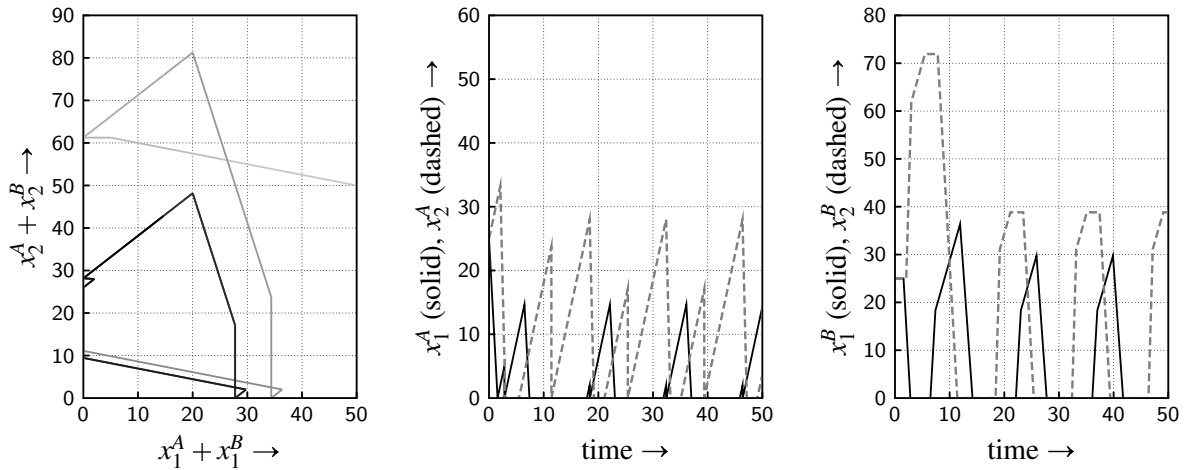


Figure 6.15: Simulation results for implementation of local clearing with double threshold controller on original hybrid fluid model. Left: periodic orbit, center: buffer levels workstation A, right: buffer levels workstation B.

All four different simulations result in mean wip levels and flow times of lots in the system. In Table 6.4 the numerical results are presented. For the discrete event simulations, a transient of 30 process cycles was taken before the measurements of flow times start. Flow times were computed during 100 process cycles, resulting in mean flow times per lot type. For the stochas-

tic simulation, this procedure has been repeated 20 times. The mean and standard deviation over those 20 simulations are shown in the table. It is striking that although the controller which uses only local information does not settle down to the desired trajectories, the mean flow times of all lots and total wip level do not differ much from the stochastic simulation results. For the lot types separately big differences exist, but overall the difference is not that big.

Table 6.4: Performance comparison for different simulations with a switching server flow line, processing two lot types. Control objective is to make the flow line with workstations A and B behave as if it were workstation B in isolation.

simulation	lot type	mean wip level	mean flow time
hybrid fluid model	1	$14\frac{2}{3}$	$3\frac{2}{3}$
	2	$14\frac{2}{3}$	$3\frac{2}{3}$
	1+2	$29\frac{1}{3}$	$3\frac{2}{3}$
deterministic discrete event	1	15.25	3.81
	2	14.80	3.70
	1+2	30.05	3.76
stochastic discrete event (20 simulations over 100 cycles)	1	16.82 ± 0.20	4.21 ± 0.051
	2	17.64 ± 0.26	4.41 ± 0.066
	1+2	34.46 ± 0.42	4.31 ± 0.053
clearing with double threshold levels (local controllers)	1	9.94	2.49
	2	25.27	6.32
	1+2	35.21	4.40

6.2.3 Special situation: identical workstations

In the previous sections two different classes of manufacturing flow lines have been studied. When can one workstation make the other workstation perform at its optimal stand alone process cycle in such a way that the corresponding mean wip level is achieved for the complete flow line? First the situation was studied in which the second workstation could keep its buffers empty, letting the first workstation process its optimal cycle. Then the counterpart was investigated: what are the conditions on the first workstation to make the flow line achieve the mean wip level of the second workstation as if it were stand alone? One situation exists that is part of both classes: a flow line consisting of identical workstations: $\mu_1^A = \mu_1^B$, $\mu_2^A = \mu_2^B$, $\sigma_{12}^A = \sigma_{12}^B$ and $\sigma_{21}^A = \sigma_{21}^B$.

In short, the conditions on the workstations in a two-workstation flow line were as follows:

- Workstation B makes the flow line behave as workstation A stand alone with respect to work in process levels. Conditions: $\mu_1^B \geq \mu_1^A$, $\mu_2^B \geq \mu_2^A$, $\sigma_{12}^B \leq \sigma_{12}^A$ and $\sigma_{21}^B \leq \sigma_{21}^A$.
- Workstation A makes the flow line behave as workstation B stand alone with respect to work in process levels. Conditions:

$$\begin{aligned} R_2 \left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A &\geq 0 \\ R_1 \left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A &\geq 0. \end{aligned}$$

It is immediately clear that identical workstation fall within the first category: all inequalities hold with equality then. The conditions of the second class of flow lines reduce to the following inequalities for identical workstations:

$$\rho_2^B \left[\tau_1^{\mu B} + \tau_2^{\lambda B} - T + \rho_1^B(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} \geq 0 \quad (6.10a)$$

$$\rho_1^B \left[\tau_2^{\mu B} + \tau_1^{\lambda B} - T + \rho_2^B(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} \geq 0. \quad (6.10b)$$

Recall that mass conservation requirement (6.8) stated:

$$\tau_i^{\mu B} + \rho_i^B(\tau_i^{\lambda B} - T) = 0.$$

This equality can be used twice in each of the inequalities (6.10). This results in $0 \geq 0$ which always holds. Thus, identical workstations also fall in the second category.

The steady state periodic process cycle of the identical workstations results in empty buffers in workstation B . The workstations mimic each other with respect to actual process rates.

As for the feedback controllers that were proposed for both classes of flow lines: they reduce to the same controller when applied to a flow line of identical workstations. The easiest way to verify this is to regard the informal representation of the controller of Proposition 6.6. Recall that $x_1^{B\#} = 0$ and $x_2^{B\#} = 0$ in case of identical workstations. With this insight, the informal representation of Proposition 6.3 is obtained.

6.3 General problem of two workstations flow line

For two special cases of a flow line of two switching servers, optimal system behavior with respect to mean wip levels has been defined so far. The cases implied imposing the optimal mean wip level of a single switching server on the entire flow line. Conditions under which this is possible were derived and after having determined optimal process cycles, feedback controllers were proposed that steer the system to the desired trajectories. For the particular case of identical workstations, it was shown that the aforementioned conditions always hold and that both feedback controllers reduce to the same controller.

The general problem of minimizing the work in process level for a flow line of two switching servers is an open problem. However, insights obtained in Chapter 5 and this chapter lead to the following lemmas which must hold for the solution of the general optimization problem.

Lemma 6.7. *In an optimal process cycle, each buffer is emptied at least once.*

Proof. Assume that an optimal cycle has a minimal buffer level of $\varepsilon > 0$ lots. An alternative process cycle with the same process rate profiles can be realized in which the particular buffer level reaches zero. The mean wip level of that particular buffer is decreased with ε now, so the claimed optimal cycle was not optimal. \square

Lemma 6.8. *Workstation A (the upstream workstation) only moves lots from A to B, so without loss of generality, workstation A can always work at the highest possible actual process rate (which might be the arrival rate).*

Proof. If lots are processed by A as quickly as possible, then lots arrive at workstation B as early as possible and therefore can leave the flow line earlier, since they can be processed earlier by B. Any other strategy in A does not result in lower wip levels, so without loss of generality, A can always process lots at the highest possible rate (which might be the arrival rate). \square

Lemma 6.9. *Workstation B removes lots from the system, so B always has to process lots at the actual highest possible rate (which might be the arrival rate).*

Proof. Assume that in an optimal cycle B does not process at the highest possible rate. In an alternative cycle, B processes at highest possible rate and stays in each mode equally long as in the optimal cycle, processing an equal number of lots as in the optimal cycle. In the alternative cycle, lots leave the system earlier than in the optimal cycle, meaning that the presumed optimal cycle was not optimal. Therefore, B always has to process lots at the highest possible rate. \square

Lemma 6.10. *At the start of τ_i^B , buffer x_i^B contains:*

- *The number of lots that arrive at workstation B outside the τ_i^B interval.*
- *The number of lots that arrive during τ_i^B that cannot be processed anymore during τ_i^B .*

Proof. Outside τ_i^B , no lots of type i are processed, so jobs arriving at workstation B reside in the buffer until processing of type i lots starts over again. Apart from these lots, it is possible that lots depart from A at a higher rate than μ_i^B at the end of τ_i^B . Not all lots can be processed then during τ_i^B . The remaining lots reside in the buffer until the next start of processing type i lots. In a steady state situation, these lots can definitely be processed in the next cycle. \square

Lemma 6.11. *Without loss of generality, one can assume that in workstation B a mode never ends with idling.*

Proof. Suppose that the optimal process cycle idles at the end of mode i . Consider an alternative cycle which is equal to the optimal cycle, apart from the fact that instead of idling, it switches to mode j . Jobs of type i are processed in the same way in the alternative cycle as in the presumed optimal cycle. However, jobs of type j might be processed earlier now and leave the system earlier. Costs of type j are therefore not higher in the alternative cycle than in the presumed optimal cycle. (Most probably the costs are even lower in the alternative cycle.) Without loss of generality, one can state that an optimal strategy never idles at the end of a mode. \square

These lemmas contribute to finding an optimal process cycle for the flow line. However, manufacturers might be satisfied with a suboptimal process cycle, as explained in the previous chapter. As long as a new process cycle is better (measured in euros) than the current one, the new cycle will do. The process of finding a ‘better’ process cycle can be guided by Lemmas 6.7–6.11 and the resulting cycle at least shares some properties with an optimal cycle. Once an optimal or desired process cycle has been found, deriving a state feedback controller is a straightforward exercise using the theory and methods presented in [73].

Unequal period lengths of the workstations

An important assumption in this chapter is that the period lengths of the process cycles in all workstations are equal. This assumption made the analysis tractable. But are equal period lengths also required? One can imagine that in situations in which the workstation’s characteristics are of the same order, the period lengths are likely to equal each other, due to the overlapping slow-mode requirements. However, the equal period assumption is not a requirement, as is shown in the following example.

Example 6.12. Consider the manufacturing flow line consisting of two workstations, serving two lot types, presented in Figure 6.16. The system characteristics and constant arrival rates are indicated in the figure. It is investigated whether it is possible to make the flow line behave as if it were only one workstation stand-alone with respect to work in process levels. For this purpose, the conditions on the workstations as presented in this chapter are checked:

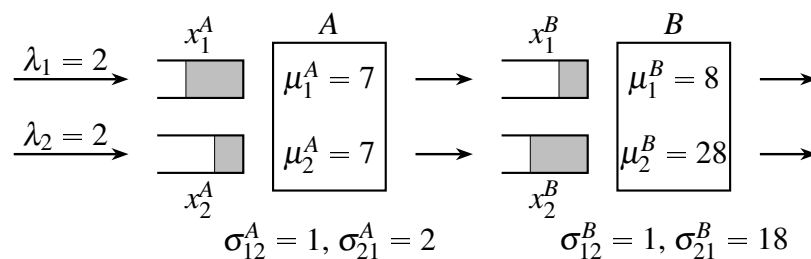


Figure 6.16: Example of flow line for which A can make the flow line behave like B stand-alone with respect to work in process levels, but with unequal period lengths

- Is it possible to make the flow line behave as if it were only workstation A? Then workstation B must not be slower than A. This holds for the maximum process rates, but not

for the setup times.

- Is it possible to make the flow line behave as if it were only workstation B ? Conditions (6.9) are checked and do not hold.

If the assumption is lifted that the periods of the process cycles in the workstations must be equal, a solution exists which makes the flow line behave as if it were only workstation B , with respect to work in process levels. The solution is presented in Figure 6.17. The left hand side graph shows the optimal periodic orbit of workstation B , derived with the theory from Chapter 5. This periodic orbit has become the orbit for the lumped buffer levels (cf. Figure 6.9). This is achieved by having workstation A perform four process cycles, while workstation B performs only one process cycle. The buffer levels of the workstations are shown in the center and right hand side graph of Figure 6.17. Note that in this solution workstation A also processes at its optimal process cycle (pure bow tie curve) as derived in Chapter 5. This means that once the flow line processes in these cycles, no room exists anymore to reduce the wip level if needed.

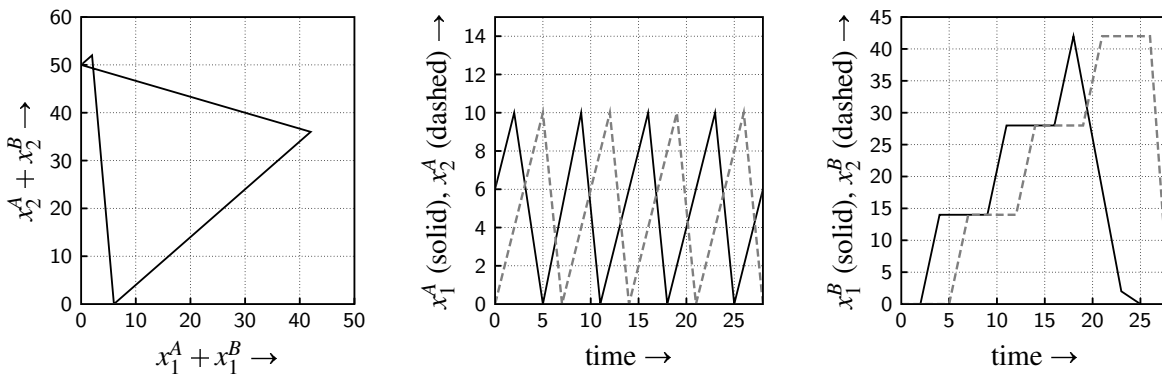


Figure 6.17: Optimal mean work in process level for the flow line is achieved with unequal period lengths of workstations A and B .

Example 6.12 shows that it is possible to make a flow line behave as one workstation with respect to wip levels when the equal period assumption is lifted. One can intuitively check that Lemmas 6.7–6.11 also hold when this assumption is lifted, making them applicable to a larger class of optimization problems of finding optimal process cycles for manufacturing flow lines and networks.

6.4 Larger flow lines

In the previous sections, optimal system behavior for certain classes of flow lines consisting of two workstations has been defined. The general idea was to have one of the workstations perform its stand-alone optimal process cycle. The other workstation should make this possible by either keeping its buffers empty (Section 6.2.1) or providing enough work at the right times

in such a way that the server performing its optimal cycle can actually do so (Section 6.2.2). In this section, similar issues are addressed for larger flow lines.

Are the results for flow lines of two workstations applicable to larger flow lines? Suppose that it is desired to achieve the optimal mean wip level of one workstation for a large flow line. Three situations are possible:

- the first workstation of the flow line is to perform its optimal cycle and the remainder of the flow line must keep its buffers empty;
- the last workstation of the flow line is to perform its optimal cycle and the rest of the flow line must make this happen;
- a combination of the first two situations: an arbitrary workstation is to perform at its optimal process cycle. Upstream workstations should make this happen and downstream workstations should keep their buffers empty.

Identification of the ‘bottleneck’ workstation can be done by determining the optimal mean wip level for each workstation with the theory of Chapter 5. The highest individual mean wip level determines the bottleneck workstation. The downstream workstations are to perform according to the first mentioned situation and the upstream workstations are to perform according to the second mentioned situation. These situations are elaborated on below.

The first situation is relatively easy: in order to keep buffers empty, the workstation should not be slower than the preceding one, as explained in Section 6.2.1. For larger flow lines these conditions hold, otherwise the downstream workstation would not be able to keep up with the upstream one. Or in other words, for a flow line consisting of $N \in \mathbb{N}$ workstations:

$$\sigma_{12}^n \leq \sigma_{12}^{n-1}, \quad \sigma_{21}^n \leq \sigma_{21}^{n-1}, \quad \mu_1^n \geq \mu_1^{n-1}, \quad \mu_2^n \geq \mu_2^{n-1} \quad \text{for } n \in \{2, 3, \dots, N\}.$$

The second situation is more complex. Upstream workstations should provide a succeeding workstation with work in such a way that the succeeding workstation is able to process lots according to a certain rate profile. In Section 6.2.1 necessary and sufficient conditions have been derived for the flow line with two workstations. For larger flow lines, the results are applicable, but the conditions are only sufficient, not necessary anymore, as explained below.

For a flow line consisting of two workstations, the conditions on the first workstation to make the flow line behave as a single server with respect to mean wip levels, left some freedom in the process cycle of the first workstation (see Figure A.21 in Appendix A.8, which shows the feasible area for θ_1^+ and θ_2^+ , the extensions of the slow-modes in workstation A in each mode). When a specific choice for θ_1^+ and θ_2^+ has been made, the process cycle of A is fixed. When an additional upstream workstation is added, the conditions on the workstations can be shifted upstream: the determined process cycle of A needs to be achieved by choosing a suitable process cycle for the newly added workstation. However, different choices in the process cycle of A might lead to different process cycles of the new workstation. Or worse: specific choices on the process cycle of A gives infeasible results for the added upstream workstation, while other choices for A lead to feasible process cycles for the added upstream workstation. It would be

better to solve the problem for multiple workstations all at once. For larger flow lines, process cycles can be obtained by solving a linear matrix inequality, as explained in the following.

Consider a flow line consisting of $N \in \mathbb{N}$ workstations, each serving two lot types, see Figure 6.18. Each server has its own processing characteristics μ_1^n, μ_2^n and setup times σ_{12}^n and σ_{21}^n , with $n \in \{1, 2, \dots, N\}$. Note that the server identification in the superscripts is a number now rather than a capital. Server N is the most downstream server which has to perform its optimal process cycle (cf. Chapter 5). All upstream workstation should behave in a way that the corresponding optimal mean work in process level can be achieved for the complete flow line.

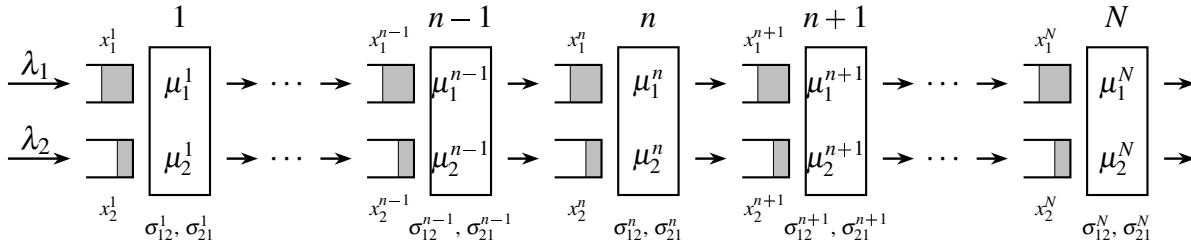


Figure 6.18: Flow line of N switching servers with two product types.

From Figure 6.10 and the observations which led to this figure it is known that the process cycle of the upstream workstation has a similar shape as the process cycle of the downstream workstation, but with possibly different lengths of all intervals. When this shape of process cycles is required for all upstream workstations, Figure 6.19 emerges. The dashed vertical lines indicate that the slow-modes should overlap each other completely when looking upstream. Slow-mode extensions θ_i^n are measured from the start and ending of the slow-mode of workstation $n+1$, as indicated graphically in the figure.

The observations for the two workstation flow line, as explained in Section 6.2.2, can be applied to the larger flow line:

- Overlapping slow-modes:

$$\theta_i^{-,n} + \tau_i^{\lambda,n+1} + \theta_i^{+,n} = \tau_i^{\lambda,n}, \quad i \in \{1, 2\}, n \in \{1, 2, \dots, N-1\} \quad (6.11a)$$

$$\theta_i^{+,n} \geq 0, \quad i \in \{1, 2\}, n \in \{1, 2, \dots, N-1\} \quad (6.11b)$$

$$\theta_i^{-,n} \geq 0, \quad i \in \{1, 2\}, n \in \{1, 2, \dots, N-1\}. \quad (6.11c)$$

The slow-mode extensions are defined with respect to the succeeding workstation, see Figure 6.19. By definition, $\theta_i^{+,N} = 0$ and $\theta_i^{-,N} = 0$.

- The period of the process cycles is equally long in all workstations (assumption).
- The equal period lengths and overlapping slow-modes lead to the following equal interval lengths:

$$\theta_2^{+,n} + \sigma_{21}^n + \tau_1^{\mu,n} + \theta_1^{-,n} = \sigma_{21}^{n+1} + \tau_1^{\mu,n+1}, \quad n \in \{1, 2, \dots, N-1\} \quad (6.12a)$$

$$\theta_1^{+,n} + \sigma_{12}^n + \tau_2^{\mu,n} + \theta_2^{-,n} = \sigma_{12}^{n+1} + \tau_2^{\mu,n+1}, \quad n \in \{1, 2, \dots, N-1\}. \quad (6.12b)$$

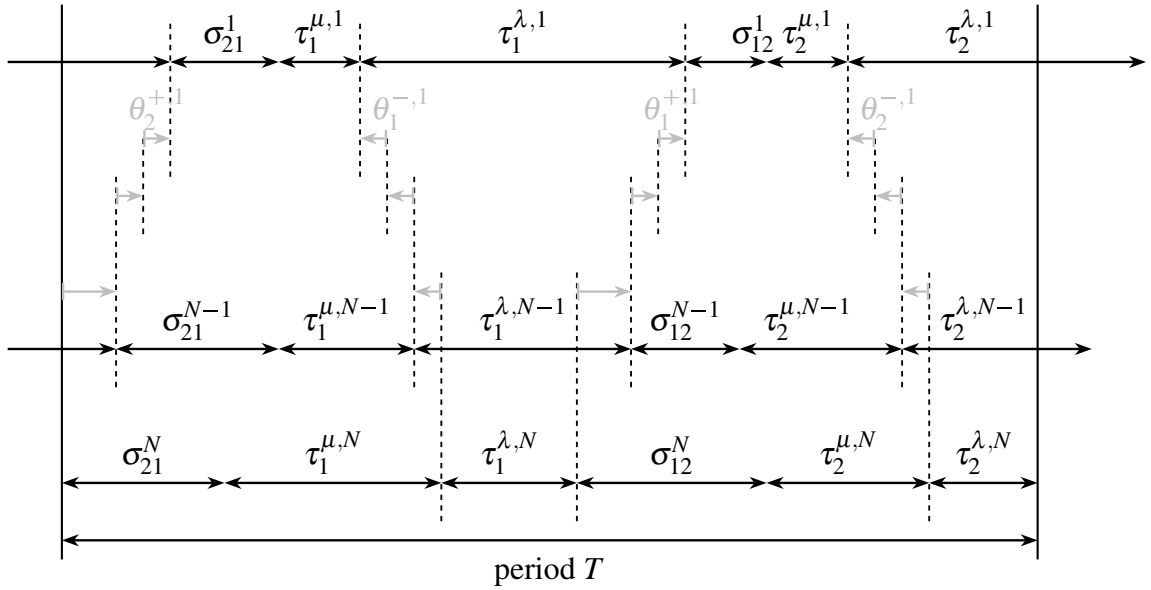


Figure 6.19: Time lines of multiple workstations, achieving optimal system behavior of most downstream workstation m stand-alone.

- Buffer levels are not allowed to become negative:

$$\mu_i^{n+1} \left(\tau_i^{\mu,n+1} - \theta_i^{-,n} - \tau_i^{\mu,n} \right) \leq \lambda_i \theta_i^{+,n}, \quad i \in \{1, 2\}, n \in \{1, 2, \dots, N-1\}. \quad (6.13)$$

- Mass conservation holds in each workstation:

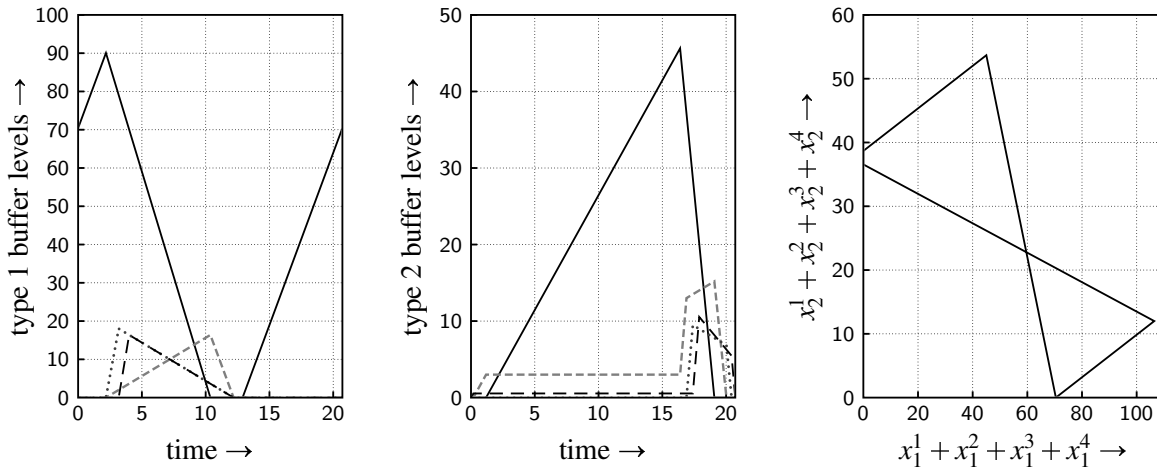
$$\mu_i^n \tau_i^{\mu,n} + \lambda_i \tau_i^{\lambda,n} = \lambda_i T, \quad i \in \{1, 2\}, n \in \{1, 2, \dots, N\}. \quad (6.14)$$

All equations and inequalities (6.11)–(6.14) are linear in all τ and θ variables, thus a linear matrix inequality (LMI) is obtained (see [100] for details on LMI theory). LMIs can be solved efficiently with numerical tools. If one is only interested in the existence of a feasible solution, a linear program solver can also be used. In fact, the linear program solver is (mis)used here to check the existence of feasible solutions. The coefficients of the linear objective function do not influence the feasibility question. Only if some freedom is left in the process cycles of the workstations, the linear objective function coefficients of the LP force the solution to a certain direction in the feasible area.

Example 6.13. Consider a flow line consisting of four workstations, each serving two product types, see Figure 6.18. The characteristics of the flow line are given in Table 6.5. Is it possible to make this flow line behave with respect to work in process levels as if it were only the most downstream workstation (workstation 4)? This problem is solved by means of constructing a linear programming problem as described above. First of all, an optimal process cycle is determined for workstation 4, with the insights obtained in Chapter 5. A slow-mode occurs for type 1 lots. The time intervals for an optimal process cycle are as follows: $\tau_1^{\mu,4} = 8.19$, $\tau_1^{\lambda,4} = 0.71$, $\tau_2^{\mu,4} = 2.83$ and $\tau_2^{\lambda,4} = 0$ hours. The period length of the steady state process cycle is 20.72 hours and the buffer values at points of switching are $x_1^\# \approx 70.4$ and $x_2^\# \approx 38.7$ lots.

Table 6.5: System parameters of flow line case study with four workstations.

λ_1 : 9 lots/hr.	μ_1^1 : 20 lots/hr.	μ_1^2 : 18 lots/hr.	μ_1^3 : 20 lots/hr.	μ_1^4 : 22 lots/hr.
λ_2 : 3 lots/hr.	μ_2^1 : 20 lots/hr.	μ_2^2 : 19 lots/hr.	μ_2^3 : 20 lots/hr.	μ_2^4 : 22 lots/hr.
	σ_{12}^1 : $3\frac{1}{2}$ hrs.	σ_{12}^2 : 4 hrs.	σ_{12}^3 : $4\frac{1}{2}$ hrs.	σ_{12}^4 : 5 hrs.
	σ_{21}^1 : 1 hr.	σ_{21}^2 : 2 hrs.	σ_{21}^3 : 3 hrs.	σ_{21}^4 : 4 hrs.

**Figure 6.20:** Buffer levels (left and center) and periodic orbit (right) of the steady state cycle of the flow line example. Buffer level graphs: workstation 1 (solid), workstation 2 (dashed gray), workstation 3 (dotted) and workstation 4 (dashed black).

For the linear programming problem, constraints (6.11)–(6.14) have been formulated for this example. The choice of the objective function parameters is arbitrary and does not influence the feasibility issue. For a certain (unspecified) choice of the objective function parameters, the resulting feasible steady state process cycle is shown in Figures 6.20 and 6.21. The latter shows the time lines of the cycles, the former shows the buffer levels over time and the periodic orbit of the lumped buffer levels of each lot type. This periodic orbit equals the periodic orbit of workstation 4 stand-alone (cf. $x_1^\#$ and $x_2^\#$ of the stand-alone curve and the corresponding values in the graph). For developing a state feedback controller that steers a trajectory of the system to the desired optimal trajectory, the reader is referred to [73].

The linear program that has been obtained by propagating conditions (6.9) over the upstream workstations results in feasible process cycles for the flow line of switching servers. However, using this method might be too restrictive. The conditions are based on the assumption that lots can only be processed at rate μ of the specific workstation or at arrival rate λ . For the flow line consisting of two workstations, this was a valid assumption, which was not over-restrictive. For a flow line consisting of multiple workstations (more than two), lots can also be processed in a slow-mode at the maximum process rate of one of the upstream workstations. In addition, it is imaginable that idling takes place in one of the workstations. This was not included in the linear

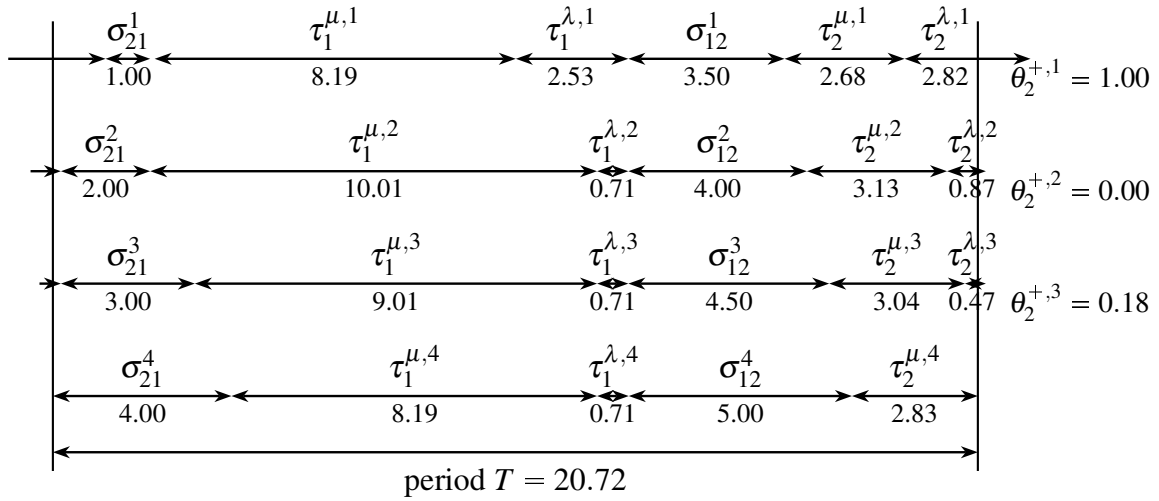


Figure 6.21: Time lines of flow line example. Numbers represent the duration of the intervals.

program that was constructed above, so infeasible LP problems do not imply that no process cycle exists that makes the flow line behave as if were only the most downstream workstation, with respect to wip levels. Conditions (6.9) are then sufficient, not necessary anymore.

6.5 Summary

This chapter elaborated on the results of Chapter 5, where optimal process cycles with respect to wip levels were derived for single switching servers with two lot types. In this chapter, flow lines of switching servers were considered. In a flow line, the number of lots in process is determined by the arrival pattern and the most downstream workstation in line. All other workstations only move lots from the one server to the other. When the arrival pattern cannot be influenced, the work in process levels of the system are only determined by the process cycle of the most downstream workstation. Is it possible to achieve the optimal wip level of a single switching server in a flow line? What are the conditions on the individual workstations? The idea here is to have one of the workstations in a flow line process at its optimal process cycle and investigate conditions on other workstations. Two classes of flow lines have been investigated:

- Flow lines for which it is possible to have the first workstation process at its optimal cycle while keeping the buffers in all other workstations empty. The optimal wip level of the stand alone first workstation is then achieved for the complete flow line. Conditions on the downstream workstation were derived. Loosely speaking, in order to keep its buffers empty, a workstation must be ‘not slower’ than its immediate upstream supplier. Both the setup times must be shorter and the process rates must be larger for each lot type.
- Flow lines for which it is possible to have the most downstream workstation process at its optimal cycle. For a flow line of two workstations, conditions on the upstream one have been derived.

For both classes of flow lines, state feedback controllers were proposed that steer the buffer levels of the workstations to the desired trajectories. Convergence to these steady state trajectories has been proven. The controllers were implemented successfully in case studies, where they were also compared to other control policies. A general conclusion from these case studies is that the workstations need a synchronization mechanism in order to reach the desired trajectories. Synchronization takes place in the proposed controllers which use full state information. When local controllers are applied (i.e. only local information of a specific workstation is available), synchronization does not take place automatically. In that case the desired steady state process cycles with corresponding work in process levels are not reached. Flow lines consisting of identical switching servers fall in both aforementioned classes and the proposed controllers reduce to the same feedback controller.

The general problem of finding an optimal process cycle for a flow line of two workstations is (still) difficult to solve. Similar remarks as in Chapter 5 can be given here: it is questionable whether one should be interested in finding an *optimal* process cycle. Apart from the mathematical challenge, suboptimal process cycles might handle disturbances or parameter changes better, as explained in the introductory chapter. Robust process cycles facilitate *smooth* operations, which can be more valuable to manufacturers than finding strictly optimal cycles.

The current graphical representation of process cycles (in the form of straight time lines, cf. Figure 6.10) is a convenient representation, but when systems become larger, the time lines become rather cluttered (cf. Figure 6.19). Especially the periodic boundary conditions make the time lines quite intransparent. A way to overcome this problem is by using rotary time lines. Figure 6.10 has been redrawn in rotary time lines in Figure 6.22. The time lines are followed in clockwise direction. The process interval lengths are no longer denoted by the lengths of the arrows, but by the angles in the figure. One complete process cycle period is 360 degrees. In addition, if it is possible to make the period of one process cycle dimensionless, the optimization procedure becomes easier, because only the area underneath the wip-time curve needs to be minimized then (cumulative costs). Lan and Olsen [69] already perform a similar analysis: they compute the relative number of setups and proportion of time processing a specific lot type per hour. Khmelnitsky and Caramanis [64] also optimize cumulative costs instead of mean costs, but do not use a normalized period length.

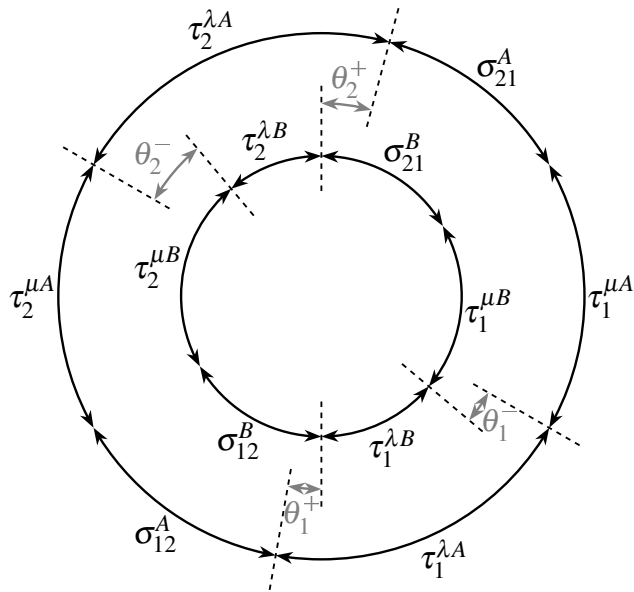


Figure 6.22: Time lines of Figure 6.10 in radial representation (for periodic boundary conditions).

Analyses in this chapter have been carried out for a flow line of two workstations. The results are applicable to larger flow lines. In an example with four workstations, it was shown that a linear programming problem or LMI can be solved to obtain a feasible process cycle that makes the flow line behave as if it were only the most downstream workstation, with respect to work in process levels. However, propagating the conditions for two workstations to conditions for a flow line of more workstations might be too restrictive, since processing lots at other rates than arrival rate or local maximum process rate is possible in larger flow lines.

Conclusions and recommendations

This chapter summarizes the conclusions from this thesis. In addition, the main contributions are recalled. Finally, some recommendations for further research are given.

7.1 Conclusions

With the increasing complexity of products and manufacturing processes, combined with ever increasing market demands, the need for advanced control strategies for manufacturing systems becomes stronger and stronger. In order to understand the dynamic aspects of manufacturing systems, models are made of the physical industrial system. The models embody only the most important phenomena that occur within the system. In this research, models have been made of manufacturing systems to perform analysis, to predict future behavior and to test and validate control methods. Based on the models, feedback controllers have been proposed that make the system behave in a pre-determined desired, possibly optimal, manner. One should keep in mind that proper working of the controller on a model in a simulation is only an indication for good results in an actual industrial implementation.

Manufacturing systems can be modelled in many different ways. Almost all models that have been treated in this thesis incorporate the aspect of time: manufacturing of products takes time. A distinction between three classes of models has been made: discrete event models, continuous models and hybrid models. Discrete event models ‘live’ in the event domain (i.e. the dynamics is driven by events). Max-plus algebraic models, min-plus algebraic models and timed process algebra have been treated. The three discrete event modelling paradigms allow for elegant and compact modelling, but have the disadvantage that most performance related measures, like

throughput, flow time, etc. are time related.

Continuous models generally ‘live’ in the time domain (discrete time domain is possible). The models that have been presented in this thesis are fluid models (based on ODEs) and flow models (based on PDEs). The fluid models lack the concept of time delay: no matter the length of the flow line, once lots are fed to the system, they immediately start to depart from the final workstation. Possible solutions to this time delay problem are sampling (resulting in discrete time models) or using approximations for the time delay, e.g. Padé approximations. Flow models are based on partial differential equations and also overcome the time delay issue. A difficulty of PDE models of manufacturing systems is that it is hard to choose the right equation describing the dynamical phenomena in manufacturing systems.

Hybrid models consist of both discrete event and continuous dynamics. The model types that have been treated in this thesis are discrete hybrid automata (DHA), hybrid fluid models (a subclass of DHA, but treated separately due to the extensive use of hybrid fluid models in this thesis) and hybrid process algebra, in the form of formalism χ .

A state space representation for a manufacturing system has been introduced. The state consists of both discrete and continuous variables. The number of lots that reside in buffers and on machines are part of this state (the discrete state elements), together with variables indicating the remaining process times of the lots that are currently on the machines (the continuous state elements). This state is finite dimensional and can be measured as if taking a snapshot from the system (collecting full state information does not take time). Moreover, the state does not contain any information about production or control policy (e.g. information that needs to be stored as a memory for the production policy). Finally, the newly developed state space representation scales up proportionally with the size of the manufacturing system.

In Chapter 3 a method has been presented to couple model types. For the max-plus, min-plus and hybrid χ model of a workstation, maps have been presented that translate the state or signals from the one representation into the other representation. In this way, analysis techniques from different domains (time or event) or paradigms can be coupled to facilitate a ‘take the best of both worlds’ idea. For one workstation, explicit maps have been presented including proofs that indeed the coupling has been established in a two-way manner. The state of the hybrid χ model has been characterized by the hybrid state as introduced earlier.

With the newly developed state space representation of a manufacturing system, it is possible to control such systems in a state feedback approach. Based on state measurements of the manufacturing system under control, new actions can be determined. In Chapter 4 a continuous time receding horizon control method has been developed to determine optimal production schedules for a fixed number of lots ahead. The feedback control law is completely determined offline using multi-parametric linear programming techniques. For a class of manufacturing systems (the class in which all product routes, recipes and orders are fixed), an optimal production schedule is available at any point in time, for all possible system states. When a disturbance occurs, a new optimal schedule is immediately available, so the controller can deal with disturbances. In Chapter 4 a framework has been developed to determine state feedback laws for manufactur-

ing flow lines or arbitrary length, with arbitrary buffer capacities at each workstation and with arbitrary control horizon. The method is applicable under certain assumptions, such as fixed product recipes and routings and first-in-first-out buffers.

In Chapters 5 and 6 switching servers have been treated. A switching server is a machine that processes multiple lot types, with switchover times between the lot types. A hybrid fluid model has been developed to describe the dynamic behavior of such servers. Optimal system behavior with respect to minimal weighted time averaged work in process levels has been derived for a switching server with two lot types that arrive with a constant inter-arrival time and both finite and infinite buffer storage capacity. The *slow-mode* concept has been introduced in Chapter 5. A slow-mode occurs when a server is processing lots at their arrival rate while the buffer of that lot type is and remains empty. In some situations (which have been made explicit) it is better to include a slow-mode in a process cycle of a switching server than to switch immediately to another lot type when the buffer is empty. This somewhat counterintuitive idea can be explained by realizing that a trade-off exists between either losing capacity due to processing lots in a slow-mode, or losing capacity due to relatively often switching between lot types. The concept of slow-mode plays an important role in the derivation of optimal process cycles for (networks of) switching servers. For a switching server with two lot types that arrive in a piecewise constant manner (e.g. a workstation that is fed by another switching server), optimal process cycles have been defined. An important conclusion from this analysis is that the number of optimization problems that need to be solved grows rapidly as the complexity of the problem increases.

State feedback controllers have been proposed (and proven to converge) that steer a switching server with constant arrival rates to its desired (optimal) trajectory from arbitrary starting point and keep it there as good as possible, even in a disturbed environment (stochastic behavior in the arrival process and process times of the server). The controllers have been validated on both the original hybrid fluid model and discrete event models. The performance of the controllers has also been compared to other controllers that are known from literature. The proposed feedback controllers outperform the standard clearing policy and a feed forward stabilizing controller.

The minimal mean wip level for a single switching server is an absolute lower bound for the mean wip level of a switching server flow line. In Chapter 6 it has been investigated under which conditions this lower bound can actually be achieved for a flow line of two switching servers with two lot types. For both a restrictive upstream workstation and a restrictive downstream workstation, conditions on the flow line have been derived that need to be fulfilled to make the flow line behave as if it were a single switching server, with respect to mean wip levels. Optimal process cycles have been derived and state feedback controllers have been proposed (and proven mathematically to converge) that steer a system's trajectory to the desired optimal one. An important observation here is that synchronization is necessary to achieve the optimal wip level, i.e. only local controllers (in which only information of one workstation is available) do not suffice to obtain optimal system behavior.

The conditions and results for flow lines consisting of two workstations have been extended to larger flow lines, which resulted in linear matrix inequalities that need to be solved to check existence of a feasible process cycle.

Research has been performed on switching servers processing more than two lot types or servers where the original hybrid fluid model has been replaced by a discrete event model. Conclusion of this study is that the derivation of an *optimal* process cycle quickly becomes too difficult, if existing at all. On the other hand, manufacturers might not always be interested in an optimal solution. Often, a *better* solution than the current one suffices, or a solution that is easy to implement, or a production process that remains stable in the presence of disturbances. The engineering aspect of the research gets more important then.

Contributions

The main contributions of this thesis are:

- A state space representation of a manufacturing workstation. This state is finite dimensional and consists of three scalars: the number of lots in the buffer, the number of lots on the machine and the remaining process time of the lot that is currently being processed. This state characterization does *not* contain any information about production or control policy. Moreover, the dimension of the state grows proportionally with the size of the manufacturing system.
- The introduced state space representation can be mapped onto other representations. In this thesis, maps to max-plus and min-plus algebraic representations were presented, but other maps are imaginable as well.
- Using multi-parametric linear programming, a continuous time receding horizon feedback control method was developed to determine optimal production schedules for a finite number of lots ahead. The method is suitable for a flow line with an arbitrary finite number of workstations, arbitrary finite number of buffer places in each workstation and arbitrary finite control horizon. The state feedback control method is based on the newly developed state space representation.
- For a switching server processing two lot types with non-zero switchover times, an optimal process cycle has been derived with respect to mean work in process levels. For a well-defined class of these switching servers, a slow-mode appears in this optimal process cycle. A slow-mode (keeping a buffer empty by processing lots at their arrival rate) occurs as the result of the trade-off between losing capacity due to processing lots under the maximum process rate and losing capacity due to relatively often switching between the lot types.
- For a flow line consisting of two switching servers as described above, necessary and sufficient conditions were derived under which it is possible to impose the minimal work in process level of a single switching server on the flow line as a whole.

7.2 Recommendations for further research

State space representation

In this thesis, the newly developed state space representation was determined for a workstation consisting of a buffer and one single-lot machine. Other manufacturing resources, such as batch machines, conveyors, (dis)assembly stations and workstations without buffer space can be characterized with a similar characterization. A batch machine fits perfectly in the newly developed state: the number of lots on the machine is then set to the batch size and allows for variable batch sizes. Extra state components might be necessary for other manufacturing resources, e.g. for a conveyor the remaining convey time of all lots residing on the conveyor are separate scalars.

If the remaining process time of lots is not known, which is quite well imaginable in industrial practice, it might be possible to use estimators for the remaining process time in the state of the manufacturing system. These estimators can be updated on-the-fly whenever new information becomes available.

Continuous time receding horizon state feedback control

The receding horizon feedback control method that has been developed in Chapter 4 can be extended in various ways. More extensive bookkeeping prevents from solving the same MPLP problem more than once, thus saving computation time. Generation of MPLP problems takes far less computation time than solving an MPLP problem, so detecting duplicates can be a great advantage, especially when larger flow lines or larger control horizons are involved.

The currently available scheduling method based on MPLP problem solutions is only valid for manufacturing systems in which the order of lots on machines is fixed and pre-determined. This is necessary to generate the precedence constraints in the linear programs. However, in more complex manufacturing networks (e.g. with re-entrant behavior or competing queues) one can imagine that dropping the fixed order assumption may lead to better system performance. The set of possible process step orders at a certain state results in more MPLP problems to solve. If the resulting value of the objective function is stored (as a look-up table with the parameter vector), then the best process step order can be selected continuously in time by choosing the schedule with the lowest costs. This is a matter of function evaluations; no optimization is involved in an implementation. The number of MPLP problems to solve increases drastically when dropping the fixed process order assumption, but with the ever increasing computational power and storage capacity of computer systems, this approach can be considered.

The current method uses the nominal process time of a machine in the generation of MPLP problems. Any variations are not taken into account. In the implementations of the controller, it was shown that due to the feedback mechanism stability for (small) disturbances has been obtained. Question remains whether the production schedules in the stochastic environment are still optimal. Introducing stochastic processes in the analysis and development of control

methods was not treated in this Ph.D. thesis, but needs study, since in a practical industrial environment constant and deterministic process times or interarrival times of lots are rare.

When stochastic behavior is added to the analysis, it is possible to link this research to recent studies on Effective Process Times (EPT), see for example [62]. The research on EPT tries to catch complex or unknown sources of variability on parameters into simplified distributions of these parameters by means of (black box) measurements on a practical industrial manufacturing system. If these distributions (or just moments of the distribution) are used in the receding horizon scheduling method, a powerful and practical engineering method emerges.

Switching servers

The currently used method for derivation of optimal process cycles is suitable for small flow lines and a limited number of lot types. So eventually, only a small class of switching server systems can be treated in the current way. For larger networks or servers with more product types, determination of optimal system behavior quickly becomes too difficult, if possible at all. As mentioned earlier, manufacturers might not be interested in the theoretically *optimal* solution of the switching server problems. Looking for optimal behavior or implementing optimal behavior simply might be too expensive. Therefore, effort should go into advanced control methods for switching server networks, that stabilize the trajectory to a *desired*, not necessarily optimal, process cycle. As long as this desired process cycle is feasible, stable, elegant and better than the current one, manufacturers will not hesitate to implement it.

In the currently proposed controllers, the transient is not designed to be optimal. In order to get optimal transient behavior, first the definition of optimal transient behavior for switching server networks must be clear.

The sensitivity of optimal process cycles (with accompanying costs) for parameter changes is worth investigating. How does an optimal process cycle change when a system parameter changes a little? This relates to the issue mentioned in the introductory chapter, where small variations in parameter settings may have a large influence on the net profit of a company. A related topic is research on the threshold levels for switching in the proposed feedback controllers. How do these parameters change in an environment with variability?

General

This Ph.D. thesis focuses on modelling and control of manufacturing *flow lines*. Many concepts, modelling techniques or control methods can be used for general manufacturing networks, traffic flow networks, communication networks, etc. as well.

The final step in the modelling and control framework (as presented in the introductory chapter), implementation of feedback controllers on physical industrial manufacturing systems, has not been treated in this thesis. Actual implementation requires studies on output measurements, state observers and signal conversions. As an intermediate step, implementation of controllers on laboratory scale manufacturing systems is useful.

Bibliography

- [1] The Sveriges Riksbank Prize in Economic Sciences in memory of Alfred Nobel, 1975. http://nobelprize.org/nobel_prizes/economics/laureates/1975/.
- [2] D. Armbruster, P. Degond, and C. Ringhofer. A model for the dynamics of large queuing networks and supply chains. *SIAM Journal on Applied Mathematics*, 66(3):896–920, 2006.
- [3] D. Armbruster, D. Marthaler, and C. Ringhofer. Kinetic and fluid model hierarchies for supply chains. *Multiscale Modeling & Simulation*, 2(1):43–61, 2003.
- [4] D. Armbruster, C. Ringhofer, and T.-C. Jo. Continuous models for production flows. In *Proceedings of the 2004 American Control Conference, Boston (MA), USA*, volume 5, pages 4589–4594, 2004.
- [5] K.J. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice Hall, 1997.
- [6] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity, An Algebra for Discrete Event Systems*. John Wiley and Sons, 1992.
- [7] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.
- [8] J.C.M. Baeten, D.A. van Beek, and J.E. Rooda. Process algebra. In P.A. Fishwick, editor, *Handbook of Dynamic System Modeling*. Chapman & Hall/CRC, 2007.
- [9] D.A. van Beek, K.L. Man, M.A. Reniers, J.E. Rooda, and R.R.H. Schiffelers. Syntax and consistent equation semantics of hybrid Chi. *Journal of Logic and Algebraic Programming*, 68(1–2):129–210, 2006.
- [10] R. Bekker, S.C. Borst, O.J. Boxma, and O. Kella. Queues with workload-dependent arrival and service rates. *Queueing Systems*, 46(3–4):537–556, 2004.

- [11] A. Bemporad. An efficient technique for translating mixed logical dynamical systems into piecewise affine systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, Las Vegas (NV), USA*, pages 1970–1975, 2002.
- [12] A. Bemporad. Efficient conversion of mixed logical dynamical systems into an equivalent piecewise affine form. *IEEE Transactions on Automatic Control*, 49(5):832–838, 2004.
- [13] A. Bemporad, F. Borrelli, and M. Morari. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on Automatic Control*, 48(9):1600–1606, 2003.
- [14] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics and constraints. *Automatica*, 35(3):407–427, 1999.
- [15] M. Boccadoro and P. Valigi. A modelling approach for the dynamic scheduling problem of manufacturing systems with non negligible setup times and finite buffers. In *Proceedings of the 42nd IEEE Conference on Decision and Control, Maui (HI), USA*, pages 5472–5477, 2003.
- [16] M. Boccadoro and P. Valigi. Optimal control of two symmetric competing queues with finite capacity and non negligible setup times. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, Lisbon, Portugal*, pages 260–266, 2003.
- [17] R.K. Boel, B. De Schutter, G. Nijssse, J.M. Schumacher, and J.H. van Schuppen. Approaches to modelling, analysis, and control of hybrid systems. *Journal A*, 40(4):16–27, 1999.
- [18] T. van den Boom and B. De Schutter. Properties of MPC for max-plus-linear systems. *European Journal of Control*, 8(5):453–462, 2002.
- [19] T.J.J. van den Boom and C.P.M. Backx. Model Predictive Control. Lecture notes, DISC: Dutch Institute of Systems and Control, 2003.
- [20] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10):1709–1721, 2005.
- [21] O.J. Boxma and D.G. Down. Dynamic server assignment in a two-queue model. *European Journal of Operational Research*, 103(3):595–609, 1997.
- [22] O.J. Boxma, H. Levy, and J.A. Westrate. Efficient visit frequencies for polling tables: Minimization of waiting cost. *Queueing systems theory and applications*, 9(1–2):133–162, 1991.
- [23] C. Buyukkoc, P. Varaiya, and J. Walrand. The $c\mu$ -rule revisited. *Advances in applied probability*, 17:237–238, 1985.

- [24] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [25] C. Chase and P.J. Ramadge. On real-time scheduling policies for flexible manufacturing systems. *IEEE transactions on automatic control*, 37(4):491–496, 1992.
- [26] R.B. Chase and N.J. Aquilano. *Production and operations management*. Irwin, 7th edition, 1995.
- [27] H. Chen and D.D. Yao. *Fundamentals of queueing networks*. Springer, 2001.
- [28] B. Cottenceau, L. Hardouin, and J.L. Biomond. Model reference control for timed event graphs in dioids. *Automatica*, 37(8):1451–1458, August 2001.
- [29] R.F. Curtain and H. Zwart. An introduction to infinite-dimensional linear systems theory. In *Number 21 in Texts in Applied Mathematics*. Springer-Verlag, Berlin, Germany, 1995.
- [30] C.F. Daganzo. Requiem for second-order fluid approximations of traffic flow. *Transportation research part B*, 29(4):277–286, 1995.
- [31] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.
- [32] G.B. Dantzig. Linear programming. *Operations Research*, 50(1):42–47, 2002.
- [33] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, July 2001.
- [34] M. Del Gaudio, F. Martinelli, and P. Valigi. A scheduling problem for two competing queues with finite capacity and non negligible setup times. In *Proceedings of the 40th IEEE Conference on Decision and Control, Orlando (FL), USA*, pages 2355–2360, 2001.
- [35] J.A.W.M. van Eekelen. Developing continuous approximation models of manufacturing systems. M.Sc. thesis, Eindhoven University of Technology, 2003. SE 420337.
- [36] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Coupling event domain and time domain models of manufacturing systems. In *Proceedings the 45th IEEE Conference on Decision and Control, San Diego (CA), USA*, pages 6068–6073, 2006.
- [37] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Feedback control of 2-product server with setups and bounded buffers. In *Proceedings of the 2006 American Control Conference, Minneapolis (MN), USA*, pages 544–549, 2006.
- [38] 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. URL <http://se.wtb.tue.nl/sereports>.
- [39] J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. State feedback control of switching server flowline with setups. In *Proceedings the 2007 American Control Conference, New York (NY), USA*, pages 3618–3623, 2007.

- [40] J.A.W.M. van Eekelen, E. Lefeber, B.J.P. Roset, H. Nijmeijer, and J.E. Rooda. Control of manufacturing systems using state feedback and linear programming. In *Proceedings of the 44th IEEE Conference on Decision and Control and 2005 European Control Conference, Seville, Spain*, pages 4652–4657, 2005.
- [41] D.D. Eisenstein. Recovering cyclic schedules using dynamic produce-up-to policies. *Operations Research*, 53(4):675–688, 2005.
- [42] G. Fábián. *A language and simulator for hybrid systems*. Ph.D. thesis, Eindhoven University of Technology, 1999. online available at <http://alexandria.tue.nl/extra2/9902279.pdf>.
- [43] S.T.J. Forschelen. Scheduling of jobs by means of explicit state feedback control. B.Sc. thesis SE 420505, Eindhoven University of Technology, 2007.
- [44] T. Gal. A "historiogramme" of parametric programming. *The Journal of the Operational Research Society*, 31(5):449–451, 1980.
- [45] T. Gal. A note on the history of parametric programming. *The Journal of the Operational Research Society*, 34(2):162–163, 1983.
- [46] G. Gallego. Scheduling the production of several items with random demands in a single facility. *Management Science*, 36(12):1579–1592, 1990.
- [47] N. Gans and G. van Ryzin. Optimal control of a multiclass, flexible queueing system. *Operations Research*, 45(5):677–693, 1997.
- [48] N. Gans and G. van Ryzin. Optimal dynamic scheduling of a general class of parallel-processing queueing systems. *Advances in Applied Probability*, 30:1130–1156, 1998.
- [49] T. Geyer, F.D. Torrisi, and M. Morari. *Hybrid Systems: Computation and Control: Proceedings of 6th International Workshop*, volume 2623/2003, chapter Efficient Mode Enumeration of Compositional Hybrid Systems, pages 216–232. Springer Berlin / Heidelberg, 2003.
- [50] R. Haijema and J. van der Wal. An MDP decomposition approach for traffic control at isolated signalized intersections. *Probability in the Engineering and Informational Sciences*, 2007. submitted.
- [51] L. Hardouin. Sur la commande de linéaire de systèmes à événements discrets dans l'algèbre (max,+). Habilitation à diriger des recherches, Université d'Angers, Istia, France, 2004.
- [52] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001.
- [53] B. Heidergott, G.J. Olsder, and J.W. van der Woude. *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press, 2006.

- [54] D. Helbing. Traffic Forum. Online community. <http://www.TrafficForum.org>.
- [55] D. Helbing. High-fidelity macroscopic traffic equations. *Physica A*, 219(3–4):391–407, 1995.
- [56] D. Helbing. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics*, 73(4):1067–1141, 2001.
- [57] D. Helbing. Micro- and macro-simulation of freeway traffic. *Mathematical and computer modelling*, 35(5–6):517–547, 2002.
- [58] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and computation*, 117(2):221–239, 1995.
- [59] A.T. Hofkamp and J.E. Rooda. *χ 1.0 reference manual*. Eindhoven University of Technology, Systems Engineering Group, 2007. <http://www.se.wtb.tue.nl/sewiki/chi>.
- [60] M. Hofri and K.W. Ross. On the optimal control of two queues with server setup times and its analysis. *SIAM Journal on computing*, 16(2):399–420, 1987.
- [61] W.J. Hopp and M.L. Spearman. *Factory physics*. McGraw-Hill International editions, 2000.
- [62] J.H. Jacobs, L.F.P. Etman, E.J.J. van Campen, and J.E. Rooda. Characterization of operational time variability using effective process times. *IEEE transactions on semiconductor manufacturing*, 16(3):511–520, 2003.
- [63] T. Jiménez and G. Koole. Scaling and comparison of fluid limits of queues applied to call centers with time-varying parameters. *OR Spectrum*, 26(3):413–422, 2004.
- [64] E. Khmelnitsky and M. Caramanis. One-machine n -part-type optimal setup scheduling: Analytical characterization of switching surfaces. *IEEE Transactions on Automatic Control*, 43(11):1584–1588, 1998.
- [65] J. Kimemia and S.B. Gershwin. An algorithm for the computer control of a flexible manufacturing system. *IIE Transactions*, 15(4):353–362, 1983.
- [66] W.W.H. de Koning. Control of systems with setup times. B.Sc. thesis SE 420464, Eindhoven University of Technology, 2006.
- [67] G. Koole and A. Mandelbaum. Queueing models of call centers, an introduction. *Annals of Operations Research*, 113(1–4):41–59, 2002.
- [68] M. Kvasnica, P. Grieder, M. Baotić, and F.J. Christophersen. *Multi-parametric toolbox (MPT)*. Institut für Automatik, ETH - Swiss Federal Institute of Technology, <http://control.ee.ethz.ch/~mpt/downloads/MPTmanual.pdf>, 2006.
- [69] W.-M. Lan and T.L. Olsen. Multiproduct systems with both setup times and costs: Fluid bounds and schedules. *Operations Research*, 54(3):505–522, 2006.

- [70] E. Lefeber. Nonlinear models for control of manufacturing systems. In *Proceedings of the 4th International Symposium on Investigations of Non-Linear Dynamic Effects in Production Systems*, 2003. Chemnitz, Germany, paper 40.
- [71] E. Lefeber. Nonlinear models for control of manufacturing systems. In *Nonlinear Dynamics of Production Systems*. Wiley-VCH, 2004.
- [72] E. Lefeber, R.A. van den Berg, and J.E. Rooda. Modelling manufacturing systems for control: A validation study. In D.H. Armbruster, K. Kaneko, and A.S. Mikhailov, editors, *Networks of interacting machines*, chapter 4, pages 101–126. World Scientific Lecture Notes, 2005.
- [73] E. Lefeber and J.E. Rooda. Controller design for switched linear systems with setups. *Physica A*, 363:48–61, 2006.
- [74] M.J. Lighthill and J.B. Whitham. On kinematic waves. I: Flow movement in long rivers. II: A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society A*, 229(1178):281–345, 1955.
- [75] J.D.C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.
- [76] J. van de Loo. Validatie van PDE modellen door middel van discrete event simulatie. B.Sc. thesis SE 420392, Eindhoven University of Technology, 2004.
- [77] T.P. Luiten. Control of a two-machine flowline with setup times. B.Sc. thesis SE 420494, Eindhoven University of Technology, 2006.
- [78] K.L. Man and R.R.H. Schiffelers. *Formal specification and analysis of hybrid systems*. Ph.D. thesis, Eindhoven University of Technology, 2006. Online available at <http://alexandria.tue.nl/extra2/200513739.pdf>.
- [79] D.M. Markowitz and L.M. Wein. Heavy traffic analysis of dynamic cyclic policies: A unified treatment of the single machine scheduling problem. *Operations Research*, 49(2):246–270, 2001.
- [80] F. Martinelli and P. Valigi. The impact of finite buffers on the optimal scheduling of a single-machine two-part-type manufacturing system. *IEEE transactions on Automatic Control*, 47(10):1705–1710, 2002.
- [81] A.S. Matveev and A.V. Savkin. *Qualitative theory of hybrid dynamical systems*. Birkhäuser, Boston, 2000.
- [82] G. Naumoski and W. Alberts. *A discrete-event simulator for systems engineering*. Ph.D. thesis, Eindhoven University of Technology, 1998. online available at <http://alexandria.tue.nl/extra2/9801171.pdf>.
- [83] I. Necoara. *Model predictive control for max-plus-linear and piecewise affine systems*. Ph.D. thesis, Delft University of Technology, 2006.

- [84] I. Necoara, B. De Schutter, and J. Hellendoorn. Structural properties of Helbing's traffic flow model. *Transportation Research Record*, (1883):21–30, 2004.
- [85] X. Nicollin and J. Sifakis. The algebra of timed processes ATP: Theory and application. *Information and Computation*, 114(1):131–178, 1994.
- [86] H.J. Payne. Models of freeway traffic and control. *Mathematical models of public services*, pages 51–60, 1971. Volume 1 of Simulations Councils Proceeding Series.
- [87] J.R. Perkins and P.R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Transactions on Automatic Control*, 34(2):139–148, 1989.
- [88] S. Platschorre. Modelling of manufacturing lines using higher order PDEs. M.Sc. thesis, Eindhoven University of Technology, 2004. SE 420413.
- [89] S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.
- [90] P. Rapisarda and J.C. Willems. State maps for linear systems. *SIAM Journal on Control and Optimization*, 35:1053–1091, 1997.
- [91] P.I. Richards. Shockwaves on the highway. *Operations Research*, 4(1):42–51, 1956.
- [92] A.D. Ridley, M.C. Fu, and W.A. Massey. Fluid approximations for a priority call center with time-varying arrivals. In *Proceedings of the 2003 Winter Simulation Conference*, volume 2, pages 1817–1823, 2003.
- [93] J.E. Rooda and J. Vervoort. Analysis of manufacturing systems using χ 1.0. Lecture notes, Eindhoven University of Technology, 2007.
- [94] B.J.P. Roset. *Manufacturing systems considered as time domain control systems: Receding horizon control and observers*. Ph.D. thesis, Eindhoven University of Technology, 2007. Available online: <http://alexandria.tue.nl/extra2/200711691.pdf>.
- [95] B.J.P. Roset, H. Nijmeijer, J.A.W.M. van Eekelen, E. Lefeber, and J.E. Rooda. Event driven manufacturing systems as time domain control systems. In *Proceedings of the 44th IEEE Conference on Decision and Control and 2005 European Control Conference, Seville, Spain*, pages 446–451, 2005.
- [96] G.J. Samson. Event-driven model predictive control of a manufacturing line. M.Sc. thesis, Eindhoven University of Technology, 2005. SE 420431.
- [97] A.V. Savkin. Regularizability of complex switched server queueing networks modelled as hybrid dynamical systems. *Systems & Control letters*, 35:291–299, 1998.
- [98] A.V. Savkin. Optimal distributed real-time scheduling of flexible manufacturing networks modeled as hybrid dynamical systems. In *Proceedings of the 42th IEEE Conference on Decision and Control, Maui (HI), USA*, pages 5468–5471, 2003.

- [99] A.V. Savkin and A.S. Matveev. A switched server system of order n with all its trajectories converging to $(n - 1)!$ limit cycles. *Automatica*, 37(2):303–306, 2001.
- [100] C.W. Scherer and S. Weiland. Linear matrix inequalities in control. Lecture notes, Dutch Institute of Systems and Control, 2005.
- [101] A.C.J. Smolders. Feedback control of a 2-product workstation with setups and one piecewise constant arrival rate. M.Sc. thesis, Eindhoven University of Technology, 2007. SE 420509.
- [102] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.
- [103] F.D. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and D. Mignone. *HYSDEL 2.0.5 – User Manual*, 2002.
- [104] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2001. Available online: <http://www.princeton.edu/~rvdb/LPbook/onlinebook.pdf>.
- [105] F.D. Vargas-Villamil and D.E. Rivera. Multilayer optimization and scheduling using model predictive control: Application to reentrant semiconductor manufacturing lines. *Computers and Chemical Engineering*, 24(8):2009–2021, 2000.
- [106] F.D. Vargas-Villamil, D.E. Rivera, and K.G. Kempf. A hierarchical approach to production control of reentrant semiconductor manufacturing lines. *IEEE Transactions on control systems technology*, 11(4):578–587, 2003.
- [107] S.L.H. Verhoeven. Feedback control of 2-product server with setups and bounded buffers. B.Sc. thesis SE 420470, Eindhoven University of Technology, 2006.
- [108] G. Weiss. A simplex based algorithm to solve separated continuous linear programs. *submitted to Mathematical Programming*, 2001. Available online: <http://stat.haifa.ac.il/~gweiss/publications/SCLP.pdf>.
- [109] E.W. Weisstein. Padé approximant. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/PadeApproximant.html>.
- [110] D. Wetjens. Discrete event system analysis using the max-plus algebra. M.Sc. thesis, Eindhoven University of Technology, 2004. SE 420388.
- [111] J.C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on automatic control*, 36(3):259–294, March 1991.
- [112] D.A.J. van Zwieten. Optimal process cycle and feedback control of a 3-product workstation with setups. B.Sc. thesis SE 420512, Eindhoven University of Technology, 2007.

Appendix A

Proofs

This appendix contains proofs of several lemmas, theorems and propositions that have been presented in this thesis. As a guide, the proofs are listed below, together with the referring page number and the page number of the proof.

Appendix	Proof of	Referring page	Proof page
A.1	Proposition 3.7	63	188
A.2	Theorem 5.10	102	193
A.3	Proposition 5.13	105	196
A.4	Proposition 5.15 and Lemma 5.14	109	198
A.5	Proposition 5.19	128	200
A.6	Remark 5.20	131	221
A.7	Proposition 6.3	145	225
A.8	Theorem 6.5	154	227
A.9	Proposition 6.6	155	230

$$\mathcal{M}_{\Theta \rightarrow \chi} : \begin{cases} x_{\chi_1}(t) = \max(0, x_{\mathcal{T}_1}(t)(0) - x_{\mathcal{T}_2}(t)(0) - 1) \\ x_{\chi_2}(t) = \min(x_{\mathcal{T}_1}(t)(0) - x_{\mathcal{T}_2}(t)(0), 1) \\ x_{\chi_3}(t) = \begin{cases} 0 & \text{if } x_{\mathcal{T}_1}(t)(0) = x_{\mathcal{T}_2}(t)(0) \\ \dots\dots\dots \\ \max(\inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_2}(t)(\tau) = x_{\mathcal{T}_2}(t)(0)\} + d, 0) \\ \quad \text{if } x_{\mathcal{T}_1}(t)(-\Delta) \geq x_{\mathcal{T}_2}(t)(0) + 1 \\ \dots\dots\dots \\ \max \left(\begin{array}{l} \inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_2}(t)(\tau) = x_{\mathcal{T}_2}(t)(0)\} + d \\ \inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_1}(t)(\tau) = x_{\mathcal{T}_2}(t)(0) + 1\} + d \\ 0 \end{array} \right) \\ \quad \text{if } x_{\mathcal{T}_1}(t)(-\Delta) < x_{\mathcal{T}_2}(t)(0) + 1 \\ \quad \text{and } x_{\mathcal{T}_1}(t)(0) > x_{\mathcal{T}_2}(t)(0) \end{cases} \\ x_{\chi_4}(t) = x_{\mathcal{T}_2}(t)(0). \end{cases} \quad (\text{A.3})$$

Map $\mathcal{M}_{\chi \rightarrow \ominus} : \mathbb{N}^2 \times \mathbb{R} \times \mathbb{Z} \rightarrow ([-\Delta, 0] \rightarrow \mathbb{Z})^2$ maps a right continuous hybrid χ state onto a set of min-plus states:

$$\mathcal{M}_{\chi \rightarrow \ominus} : \left\{ \begin{array}{l} \{\mathbf{x}_T(t) \mid \exists \tilde{\mathbf{x}}_T(t) : [-\Delta - d, d] \rightarrow \mathbb{Z}^2 \text{ with } \Delta \geq d \text{ such that:} \\ \dots\dots\dots \\ \textcircled{1} : \tilde{x}_{T_1}(t)(\tau) \leq \tilde{x}_{T_2}(t)(\tau) + N_1 + 1 \\ \textcircled{2} : \tilde{x}_{T_2}(t)(\tau) = \min(\tilde{x}_{T_1}(t)(\tau - d), \tilde{x}_{T_2}(t)(\tau - d) + 1) \text{ for } \tau \in [-\Delta, d] \\ \dots\dots\dots \\ \textcircled{3} : \tilde{x}_{T_1}(t)(0) = x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) \\ \tilde{x}_{T_2}(t)(\tau) = \begin{cases} \textcircled{4} : x_{\chi_4}(t) & \text{for } \tau \in [x_{\chi_3}(t) - d, 0] \text{ if } x_{\chi_2}(t) > 0 \\ \textcircled{5} : x_{\chi_4}(t) & \text{for } \tau = 0 \text{ if } x_{\chi_2}(t) = 0 \end{cases} \\ \textcircled{6} : \tilde{x}_{T_1}(t)(x_{\chi_3}(t) - d) \geq x_{\chi_4}(t) + 1 \text{ if } x_{\chi_2}(t) > 0 \\ \textcircled{7} : \tilde{x}_{T_2}(t)(\tau) = x_{\chi_4}(t) \text{ for } 0 \leq \tau < x_{\chi_3}(t) \text{ if } x_{\chi_2}(t) > 0 \\ \dots\dots\dots \\ \textcircled{8} : \forall \varepsilon > 0, \forall \tau \in [-\Delta - d + \varepsilon, d] : \tilde{x}_{T_i}(t)(\tau - \varepsilon) \leq \tilde{x}_{T_i}(t)(\tau), i \in \{1, 2\} \\ \dots\dots\dots \\ \forall \tau \in [-\Delta, 0] \text{ and } i \in \{1, 2\} : x_{T_i}(t)(\tau) = \tilde{x}_{T_i}(t)(\tau) \}. \end{array} \right. \quad (\text{A.4})$$

The encircled numbers ①–⑧ are used to indicate the respective (in)equalities in the map. The proof consists of two parts. First, (A.1) is proven, after which (A.2) is proven.

The first part of the proof concerns expression (A.1). Starting with right continuous hybrid χ state $\mathbf{x}_\chi(t) \in \mathcal{B}_{T_s}^\chi$, it is to be shown that when mapped to a min-plus state $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$ and back again to a right continuous hybrid χ state, the original state $\mathbf{x}_\chi(t)$ is obtained.

Suppose that $\mathbf{x}_\chi(t) = [x_{\chi_1}(t) \ x_{\chi_2}(t) \ x_{\chi_3}(t) \ x_{\chi_4}(t)]^\top \in \mathcal{B}_{T_s}^\chi$ is given. Map $\mathcal{M}_{\chi \rightarrow \ominus}$ yields a min-plus state $\mathbf{x}_T(t)$.

Let an $\mathbf{x}_T(t) \in \mathcal{M}_{\chi \rightarrow \ominus}(\mathbf{x}_\chi(t))$ be given. Then an auxiliary state $\tilde{\mathbf{x}}_T(t)$ exists such that all equalities and inequalities in (A.4) hold. To be proven: $\mathcal{M}_{\ominus \rightarrow \chi}(\tilde{\mathbf{x}}_T(t)|_{[-\Delta, 0]}) = \mathbf{x}_\chi(t)$.

Let $\mathbf{v}_\chi(t) = [v_{\chi_1}(t) \ v_{\chi_2}(t) \ v_{\chi_3}(t) \ v_{\chi_4}(t)]^\top = \mathcal{M}_{\ominus \rightarrow \chi}(\tilde{\mathbf{x}}_T(t)|_{[-\Delta, 0]})$.

Using map (A.3), each component of $\mathbf{v}_\chi(t)$ is determined:

$$\begin{aligned} v_{\chi_1}(t) &= \max(0, \tilde{x}_{T_1}(t)(0) - \tilde{x}_{T_2}(t)(0) - 1) \\ &= \max(0, x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) - x_{\chi_4}(t) - 1) \\ &= \max(0, x_{\chi_1}(t) + x_{\chi_2}(t) - 1) \\ &= x_{\chi_1}(t) \quad (\text{Lemma A.1}) \\ v_{\chi_2}(t) &= \min(\tilde{x}_{T_1}(t)(0) - \tilde{x}_{T_2}(t)(0), 1) \\ &= \min(x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) - x_{\chi_4}(t), 1) \\ &= \min(x_{\chi_1}(t) + x_{\chi_2}(t), 1) \\ &= x_{\chi_2}(t) \quad (\text{Lemma A.2}) \\ v_{\chi_4}(t) &= \tilde{x}_{T_2}(t)(0) = x_{\chi_4}(t). \end{aligned}$$

The expression for $x_{\chi_3}(t)$ in (A.3) consists of three parts, each with its own conditions:

- $v_{\chi_3}(t) = 0$ if $\tilde{x}_{\mathcal{T}_1}(t)(0) = \tilde{x}_{\mathcal{T}_2}(t)(0)$. Suppose that this condition holds. Using (A.4), the following result is obtained:

$$\begin{aligned}\tilde{x}_{\mathcal{T}_1}(t)(0) &= \tilde{x}_{\mathcal{T}_2}(t)(0) \\ x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) &= x_{\chi_4}(t) \\ x_{\chi_2}(t) + x_{\chi_1}(t) &= 0.\end{aligned}$$

Property (3.14d) states that if $x_{\chi_2}(t) = 0$, the remaining process time $x_{\chi_3}(t) = 0$ as well.

- Suppose that $\tilde{x}_{\mathcal{T}_1}(t)(-\Delta) \geq \tilde{x}_{\mathcal{T}_2}(t)(0) + 1$. State $\tilde{\mathbf{x}}_T(t)$ obeys inequality ⑧ (monotonicity), so $\tilde{x}_{\mathcal{T}_1}(t)(0) \geq \tilde{x}_{\mathcal{T}_2}(t)(0) + 1$. As a result, $x_{\chi_2}(t) = \min(\underbrace{\tilde{x}_{\mathcal{T}_1}(t)(0) - \tilde{x}_{\mathcal{T}_2}(t)(0)}_{\geq 1}, 1) = 1$.

Because $x_{\chi_2}(t) = 1$, it follows that $x_{\chi_3}(t) > 0$ and for $0 < \varepsilon \leq x_{\chi_3}(t)$ the following equality holds: $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - \varepsilon) = x_{\chi_4}(t)$.

In addition, the following always holds (①):

$$\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - \varepsilon) = \min(\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon), \tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) + 1). \quad (\text{A.5})$$

Because $\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon) \geq \tilde{x}_{\mathcal{T}_1}(t)(-\Delta) \geq \tilde{x}_{\mathcal{T}_2}(t)(0) + 1 \geq \tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) + 1$, it follows that $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - \varepsilon) = \tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) + 1$.

This results in $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) = x_{\chi_4}(t) - 1$ for $\varepsilon \leq x_{\chi_3}(t)$ and thus $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) < x_{\chi_4}(t)$ for all $\varepsilon > 0$.

Therefore, the following result is obtained:

$$\begin{aligned}v_{\chi_3}(t) &= \max(\inf \tau \in [-\Delta, 0] | \tilde{x}_{\mathcal{T}_2}(t)(\tau) = \tilde{x}_{\mathcal{T}_2}(t)(0) + d, 0) \\ &= \max(x_{\chi_3}(t) - d + d, 0) = x_{\chi_3}(t).\end{aligned}$$

- Suppose that $\tilde{x}_{\mathcal{T}_1}(t)(-\Delta) < \tilde{x}_{\mathcal{T}_2}(t)(0) + 1$ and $\tilde{x}_{\mathcal{T}_1}(t)(0) > \tilde{x}_{\mathcal{T}_2}(t)(0)$. Again, it follows that $x_{\chi_2}(t) = \min(\underbrace{\tilde{x}_{\mathcal{T}_1}(t)(0) - \tilde{x}_{\mathcal{T}_2}(t)(0)}_{\geq 1}, 1) = 1$. Now it follows that:

$$v_{\chi_3}(t) = \max \left(\begin{array}{l} 0 \\ \inf\{\tau \in [-\Delta, 0] | \tilde{x}_{\mathcal{T}_2}(t)(\tau) = \tilde{x}_{\mathcal{T}_2}(t)(0)\} + d \\ \inf\{\tau \in [-\Delta, 0] | \tilde{x}_{\mathcal{T}_1}(t)(\tau) = \tilde{x}_{\mathcal{T}_2}(t)(0) + 1\} + d \end{array} \right).$$

Note that all three terms in this max-expression are $\leq x_{\chi_3}(t)$ and thus $v_{\chi_3}(t) \leq x_{\chi_3}(t)$.

It is known that (A.5) always holds. Two options are left then:

- Either $\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon) \geq \tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) + 1$:
Then it follows that $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) < x_{\chi_4}(t)$ for all $\varepsilon > 0$ (see above) and thus $\inf\{\tau \in [-\Delta, 0] | \tilde{x}_{\mathcal{T}_2}(t)(\tau) = \tilde{x}_{\mathcal{T}_2}(t)(0)\} + d = x_{\chi_3}(t)$.
- Or: $\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon) < \tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - d - \varepsilon) + 1$:
Then it follows that $\tilde{x}_{\mathcal{T}_2}(t)(x_{\chi_3}(t) - \varepsilon) = \tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon)$ (cf. (A.5)).
Due to ④, one can conclude that $\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon) = x_{\chi_4}(t)$ and thus $\tilde{x}_{\mathcal{T}_1}(t)(x_{\chi_3}(t) - d - \varepsilon) < x_{\chi_4}(t) + 1$ for $\varepsilon \leq x_{\chi_3}(t)$ and also for all $\varepsilon > 0$, due to ⑧.
Thus, $\inf\{\tau \in [-\Delta, 0] | \tilde{x}_{\mathcal{T}_1}(t)(\tau) = \underbrace{\tilde{x}_{\mathcal{T}_2}(t)(0) + 1}_{x_{\chi_4}(t)+1}\} + d = x_{\chi_3}(t)$.

This completes the first part of the proof, in which it has been shown that $\forall \mathbf{x}_\chi(t) \in \mathcal{B}_{T_s}^\chi$, the original state $\mathbf{x}_\chi(t)$ is obtained back again when mapped to a min-plus state $\mathbf{x}_T(t)$ and back again, i.e. $\mathbf{x}_\chi(t) = \mathcal{M}_{\ominus \rightarrow \chi}(\mathcal{M}_{\chi \rightarrow \ominus}(\mathbf{x}_\chi(t)))$.

The second part of the proof concerns (A.2). Given a state $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$, it is to be shown that when mapped to a right continuous hybrid χ state $\mathbf{x}_\chi(t) \in \mathcal{B}_{T_s}^\chi$ and back again, the resulting min-plus state $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$.

Suppose $\mathbf{x}_T(t) = \begin{bmatrix} x_{T_1}(t) & x_{T_2}(t) \end{bmatrix}^\top$ is given. From (A.3) follows:

$$\mathbf{x}_\chi(t) = \begin{bmatrix} x_{\chi_1}(t) & x_{\chi_2}(t) & x_{\chi_3}(t) & x_{\chi_4}(t) \end{bmatrix}^\top = \mathcal{M}_{\ominus \rightarrow \chi}(\mathbf{x}_T(t)).$$

To be proven: $\mathbf{x}_T(t) \in \mathcal{M}_{\chi \rightarrow \ominus}(\mathbf{x}_\chi(t))$, or in other words: construct an $\tilde{\mathbf{x}}_T(t) \in [-\Delta - d, d]$ such that all (in)equalities in (A.4) are fulfilled and $\mathbf{x}_\chi(t) = \mathcal{M}_{\ominus \rightarrow \chi}(\tilde{\mathbf{x}}_T(t))$.

For $\tilde{\mathbf{x}}_T(t) \in [-\Delta - d, d]$, the following expressions are taken:

$$\tilde{x}_{T_1}(t)(\tau) = \begin{cases} \star & \text{for } \tau \in [-\Delta - d, -\Delta) \\ x_{T_1}(t)(\tau) & \text{for } \tau \in [-\Delta, 0] \\ x_{T_1}(t)(0) & \text{for } \tau \in [0, d] \end{cases} \quad (\text{A.6})$$

$$\tilde{x}_{T_2}(t)(\tau) = \begin{cases} x_{T_2}(t)(-\Delta) & \text{for } \tau \in [-\Delta - d, -\Delta) \\ x_{T_2}(t)(\tau) & \text{for } \tau \in [-\Delta, 0] \\ \min(x_{T_1}(t)(\tau - d), x_{T_2}(t)(\tau - d) + 1) & \text{for } \tau \in [0, d]. \end{cases} \quad (\text{A.7})$$

Signals $\tilde{\mathbf{x}}_T(t)$ are a copy of $\mathbf{x}_T(t)$ on the interval $[-\Delta, 0]$. For the interval $[0, d]$ the expressions can be interpreted in words as: no additional lots arrive at the workstation ($x_{T_1}(t)\tau$ remains constant after $\tau = 0$) and lots keep coming out of the system until the workstation is empty. For the \star expression, two situations are distinguished:

situation 1: $x_{T_2}(t)(-\Delta) < x_{T_2}(t)(-\Delta + d)$

Let $\vartheta = \inf \{ \tau \in [-\Delta, -\Delta + d] \mid x_{T_2}(t)(\tau) = x_{T_2}(t)(-\Delta + d) \}$

$$\tilde{x}_{T_1}(t)(\tau) = \begin{cases} x_{T_2}(t)(-\Delta) & \text{for } \tau \in [-\Delta - d, \vartheta - d) \\ x_{T_1}(t)(-\Delta) & \text{for } \tau \in [\vartheta - d, -\Delta]. \end{cases}$$

situation 2: $x_{T_2}(t)(-\Delta) = x_{T_2}(t)(-\Delta + d)$

Then $x_{T_1}(t)(-\Delta) = x_{T_2}(t)(-\Delta)$, since $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$ ((3.6b) with $\tau = -\Delta + d$)

$$\tilde{x}_{T_1}(t)(\tau) = x_{T_2}(t)(-\Delta) \text{ for } \tau \in [-\Delta - d, -\Delta].$$

In Figure A.1 two examples are given of the construction of the signals $\tilde{\mathbf{x}}_T(t)$. First monotonicity requirement ⑧ of (A.4) is checked. Since $\tilde{\mathbf{x}}_T(t)$ is a copy of the original $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$, one can immediately conclude that monotonicity is guaranteed on the interval $[-\Delta, 0]$. For the extensions of the domain at both sides, the monotonicity of the signals has to be checked. $\tilde{x}_{T_1}(t)(\tau) = x_{T_1}(t)(0)$ for $\tau \in [0, d]$ and $\tilde{x}_{T_2}(t)(\tau) = x_{T_2}(t)(-\Delta)$ for $\tau \in [-\Delta - d, -\Delta]$, which are both constant and equal at their respective boundaries, so for those subdomains, monotonicity has been proven. The \star expression for situation 2 follows a similar reasoning. For situation 1,

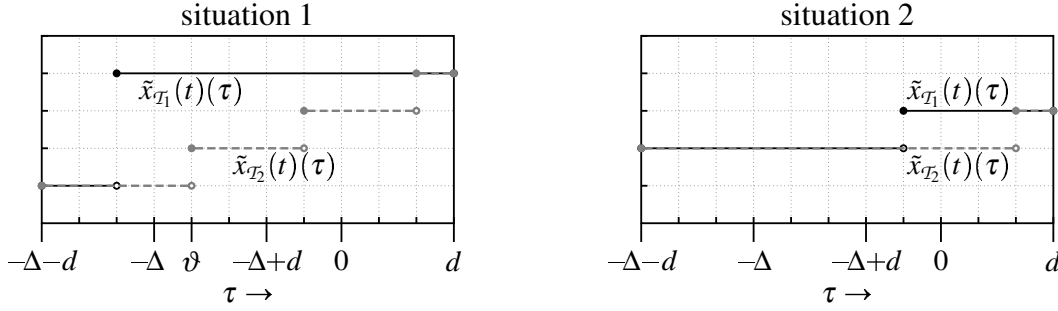


Figure A.1: Examples of construction of $\tilde{\mathbf{x}}_T(t)$.

$\tilde{x}_{T_1}(t)(-\Delta) = x_{T_1}(t)(-\Delta)$ and keeps that value on the subdomain $[\vartheta - d, -\Delta]$.

For $\tau \in [-\Delta - d, \vartheta - d]$, $\tilde{x}_{T_1}(t)(\tau) = x_{T_2}(t)(-\Delta) \leq x_{T_1}(t)(-\Delta)$ (cf. (3.6b)).

Because the original $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$, $\mathbf{x}_T(t)$ is monotonously non-decreasing on its domain, and $\min(x_{T_1}(t)(\tau - d), x_{T_2}(t)(\tau - d) + 1)$ is a non-decreasing function over $\tau \in [0, d]$, one can conclude that the constructed $\tilde{\mathbf{x}}_T(t)$ fulfills the monotonicity requirement.

Inequality ① can easily be verified considering that:

$$\begin{aligned} \tilde{x}_{T_1}(t)(\tau) - \tilde{x}_{T_2}(t)(\tau) &\leq \tilde{x}_{T_1}(t)(-\Delta) - \tilde{x}_{T_2}(t)(-\Delta) && \text{for all } \tau \in [-\Delta - d, -\Delta] \\ \tilde{x}_{T_1}(t)(\tau) - \tilde{x}_{T_2}(t)(\tau) &\leq \tilde{x}_{T_1}(t)(0) - \tilde{x}_{T_2}(t)(0) && \text{for all } \tau \in [0, d]. \end{aligned}$$

combined with the fact that the original $\mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\ominus$ and the constructed $\tilde{\mathbf{x}}_T(t)$ being a copy of the original $\mathbf{x}_T(t)$ on the interval $[-\Delta, 0]$.

Because the constructed $\tilde{\mathbf{x}}_T(t)$ is a copy of the original $\mathbf{x}_T(t)$ on $[-\Delta, 0]$, equality ② only needs to be checked for the interval $[0, d]$. The constructed $\tilde{x}_{T_2}(t)$ has the same expression as ②, so this equality holds.

Equality ③ is verified using the expressions from map $\mathcal{M}_{\ominus \rightarrow \chi}$:

$$\begin{aligned} \tilde{x}_{T_1}(t)(0) &= x_{\chi_4}(t) + x_{\chi_2}(t) + x_{\chi_1}(t) \\ &= x_{T_2}(t)(0) + \max(0, x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1) + \min(1, x_{T_1}(t)(0) - x_{T_2}(t)(0)) \\ &= x_{T_2}(t)(0) + \max(0, x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1) + \min(0, x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1) + 1 \\ &= x_{T_2}(t)(0) + 0 + x_{T_1}(t)(0) - x_{T_2}(t)(0) - 1 + 1 \\ &= x_{T_1}(t)(0). \end{aligned}$$

Equalities ④ and ⑤ are verified as follows:

- If $x_{T_1}(t)(0) = x_{T_2}(t)(0)$ then $x_{\chi_2}(t) = 0$ (cf. (A.3)). By definition: $\tilde{x}_{T_2}(t)(0) = x_{T_2}(t)(0)$ and (A.3) states that $x_{\chi_4}(t) = x_{T_2}(t)(0)$.
As a conclusion: $\tilde{x}_{T_2}(t)(\tau) = x_{\chi_4}(t)$ for $\tau = 0$ and $x_{\chi_2}(t) = 0$. This proves ⑤.
- If $x_{T_1}(t)(0) > x_{T_2}(t)(0)$ then $x_{\chi_2}(t) = 1$ (cf. (A.3)).
Also: $x_{\chi_3}(t) - d \geq \inf\{\tau \in [-\Delta, 0] | x_{T_2}(t)(\tau) = x_{T_2}(t)(0)\}$ (see Figure A.2).

Therefore, $x_{\mathcal{T}_2}(t)(\tau) = x_{\mathcal{T}_2}(t)(0)$ for $x_{\mathcal{X}_3}(t) - d \leq \tau \leq 0$.

Since $x_{\mathcal{T}_2}(t)(0) = x_{\mathcal{X}_4}(t)$ according to (A.3), this yields:

$\tilde{x}_{\mathcal{T}_2}(t)(\tau) = x_{\mathcal{X}_4}(t)$ for all $\tau \in [x_{\mathcal{X}_3}(t) - d, 0]$, which proves ④.

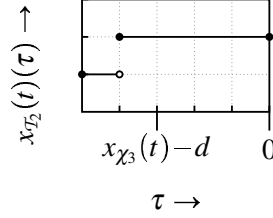


Figure A.2: Location of the infimum $< x_{\mathcal{X}_3}(t) - d$.

Inequality ⑥ is also verified in two parts:

- If $x_{\mathcal{T}_1}(t)(-\Delta) \geq x_{\mathcal{T}_2}(t)(0) + 1$ then $x_{\mathcal{T}_1}(t)(x_{\mathcal{X}_3}(t) - d) \geq x_{\mathcal{T}_1}(t)(-\Delta) \geq x_{\mathcal{T}_2}(t)(0) + 1$ and therefore $\tilde{x}_{\mathcal{T}_1}(t)(x_{\mathcal{X}_3}(t) - d) \geq x_{\mathcal{X}_4}(t) + 1$.
- If $x_{\mathcal{T}_1}(t)(-\Delta) < x_{\mathcal{T}_2}(t)(0) + 1$ and $x_{\mathcal{T}_1}(t)(0) > x_{\mathcal{T}_2}(t)(0)$ then $x_{\mathcal{X}_3}(t) - d \geq \inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_1}(t)(\tau) = x_{\mathcal{T}_2}(t)(0) + 1\}$. This results in $x_{\mathcal{T}_1}(t)(\tau) = x_{\mathcal{T}_2}(t)(0) + 1$ for $x_{\mathcal{X}_3}(t) - d \leq \tau \leq 0$ and therefore $\tilde{x}_{\mathcal{T}_1}(t)(x_{\mathcal{X}_3}(t) - d) = x_{\mathcal{X}_4}(t) + 1$.

Finally, equality ⑦ is verified. This equality is only valid for $0 \leq \tau < x_{\mathcal{X}_3}(t)$, so it is assumed that $x_{\mathcal{X}_3}(t) > 0$. Furthermore, by definition, the following holds:

$\tilde{x}_{\mathcal{T}_2}(t)(\tau) = \min(x_{\mathcal{T}_1}(t)(\tau - d), x_{\mathcal{T}_2}(t)(\tau - d) + 1)$. Two situations can occur:

- Either: $x_{\mathcal{X}_3}(t) - d = \inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_2}(t)(\tau) = x_{\mathcal{T}_2}(t)(0)\}$. Then the following holds: $x_{\mathcal{T}_2}(t)(\tau - d) \leq x_{\mathcal{T}_2}(t)(0) - 1$ for $0 \leq \tau < x_{\mathcal{X}_3}(t)$.

$$\begin{aligned} \tilde{x}_{\mathcal{T}_2}(t)(\tau) &= \min(x_{\mathcal{T}_1}(t)(\tau - d), x_{\mathcal{T}_2}(t)(\tau - d) + 1) \leq x_{\mathcal{T}_2}(t)(\tau - d) + 1 \\ &\leq x_{\mathcal{T}_2}(t)(0) - 1 + 1 = x_{\mathcal{T}_2}(t)(0) = x_{\mathcal{X}_4}(t). \end{aligned}$$

- Or: $x_{\mathcal{X}_3}(t) - d = \inf\{\tau \in [-\Delta, 0] | x_{\mathcal{T}_1}(t)(\tau) = x_{\mathcal{T}_2}(t)(0) + 1\}$. Then the following holds: $x_{\mathcal{T}_1}(t)(\tau - d) \leq x_{\mathcal{T}_2}(t)(0)$ for $0 \leq \tau < x_{\mathcal{X}_3}(t)$.

$$\tilde{x}_{\mathcal{T}_2}(t)(\tau) = \min(x_{\mathcal{T}_1}(t)(\tau - d), x_{\mathcal{T}_2}(t)(\tau - d) + 1) \leq x_{\mathcal{T}_1}(t)(\tau - d) \leq x_{\mathcal{T}_2}(t)(0) = x_{\mathcal{X}_4}(t).$$

With this, the proof that $\forall \mathbf{x}_T(t) \in \mathcal{B}_{T_s}^\Theta$: $\mathbf{x}_T(t) \in \mathcal{M}_{\chi \rightarrow \Theta}(\mathcal{M}_{\Theta \rightarrow \chi}(\mathbf{x}_T(t)))$ has been completed.

A.2 Proof of Theorem 5.10 (page 102)

The following is the proof of Theorem 5.10, which states that optimal process cycles with respect to time averaged weighted wip levels for a switching server with two different lot types and linear costs on wip levels (5.7) have a slow-mode (see Remark 5.9) for at most one lot type

(type 1). During the slow-mode, lots are processed at their arrival rate, keeping the respective buffer empty. The slow-mode occurs if and only if $c_1\lambda_1(\rho_1 + \rho_2) - (c_1\lambda_1 - c_2\lambda_2)(1 - \rho_2) < 0$.

For reasons of readability, Figure 5.5 is recalled in Figure A.3. From Corollary 5.7 the gen-

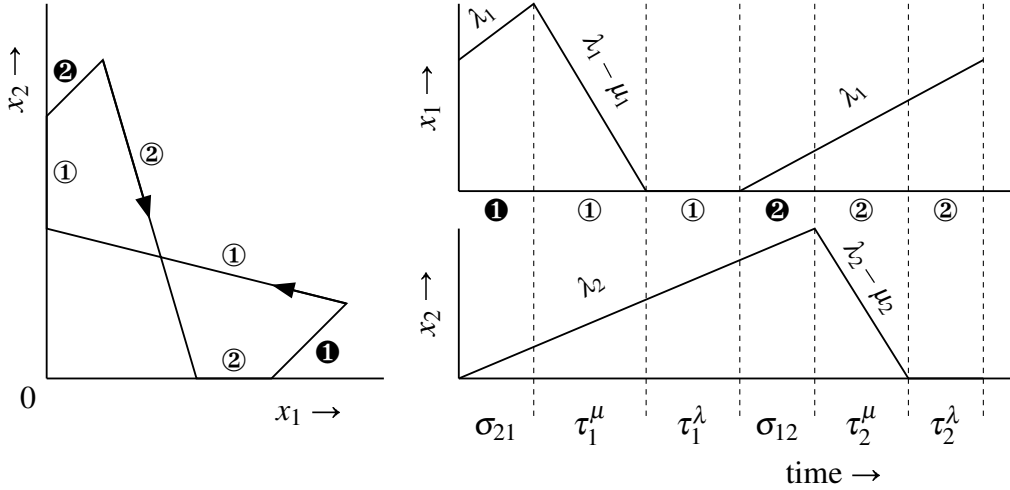


Figure A.3: General form of an optimal process cycle for a switching server processing two lot types. Left: Periodic orbit. Right: buffer levels over time, with slopes of the lines. (Recall of Figure 5.5.)

eral shape of optimal process cycles is known. Let τ_i^μ denote the duration of serving type i at maximal rate, and let τ_i^λ denote the duration of the slow-mode of type i , as indicated in Figure A.3.

In steady state, the system reaches the same situation after one complete period. During processing a lot type at full rate, as many lots are processed as arrive during setups and processing the other lot type:

$$\lambda_1(\sigma_{12} + \tau_2^\mu + \tau_2^\lambda + \sigma_{21}) = (\mu_1 - \lambda_1)\tau_1^\mu \quad (\text{A.8a})$$

$$\lambda_2(\sigma_{21} + \tau_1^\mu + \tau_1^\lambda + \sigma_{12}) = (\mu_2 - \lambda_2)\tau_2^\mu. \quad (\text{A.8b})$$

Let $\tau_1^\lambda = \alpha_1\sigma$ and $\tau_2^\lambda = \alpha_2\sigma$ with $\alpha_1 \geq 0, \alpha_2 \geq 0$ and recall $\sigma = \sigma_{12} + \sigma_{21}$. Solving (A.8) for τ_i^μ :

$$\begin{bmatrix} \sigma \\ \tau_1^\mu \\ \tau_1^\lambda \\ \tau_2^\mu \\ \tau_2^\lambda \end{bmatrix} = \frac{\sigma}{1 - \rho_1 - \rho_2} \begin{bmatrix} 1 - \rho_1 - \rho_2 \\ \alpha_1\rho_1\rho_2 + \alpha_2\rho_1(1 - \rho_2) + \rho_1 \\ \alpha_1(1 - \rho_1 - \rho_2) \\ \alpha_1\rho_2(1 - \rho_1) + \alpha_2\rho_1\rho_2 + \rho_2 \\ \alpha_2(1 - \rho_1 - \rho_2) \end{bmatrix}. \quad (\text{A.9})$$

The weighted mean wip level is computed by determining the area underneath the right hand

side graphs of Figure A.3, divided by the period length T :

$$\frac{1}{T} \int_0^T c_1 x_1(s) ds = \frac{c_1 \cdot \frac{1}{2} \cdot (\sigma + \tau_1^\mu + \tau_2^\mu + \tau_2^\lambda) \cdot (\mu_1 - \lambda_1) \tau_1^\mu}{\sigma + \tau_1^\mu + \tau_1^\lambda + \tau_2^\mu + \tau_2^\lambda} \quad (\text{A.10a})$$

$$\frac{1}{T} \int_0^T c_2 x_2(s) ds = \frac{c_2 \cdot \frac{1}{2} \cdot (\sigma + \tau_1^\mu + \tau_2^\mu + \tau_1^\lambda) \cdot (\mu_2 - \lambda_2) \tau_2^\mu}{\sigma + \tau_1^\mu + \tau_1^\lambda + \tau_2^\mu + \tau_2^\lambda}. \quad (\text{A.10b})$$

Since $\frac{1}{2}\sigma/(1 - \rho_1 - \rho_2)$ is a constant, minimizing (5.7) equals minimizing

$$\frac{c_1 \lambda_1 (1 - \rho_1) [1 + \alpha_1 \rho_2 + \alpha_2 (1 - \rho_2)]^2 + c_2 \lambda_2 (1 - \rho_2) [1 + \alpha_1 (1 - \rho_1) + \alpha_2 \rho_1]^2}{1 + \alpha_1 (1 - \rho_1) + \alpha_2 (1 - \rho_2)} \quad (\text{A.11})$$

subject to the constraints $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$.

Instead of solving this constrained optimization problem first the unconstrained problem is considered, i.e. constraints $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$ are ignored. Taking the derivatives of (A.11) with respect to α_1 and α_2 and putting them to zero gives the solution of the unconstrained problem: $\alpha_1 = \alpha_2 = -1$ (infeasible; local maximum) or

$$\alpha_1 = 1 - \frac{2c_2 \lambda_2}{c_1 \lambda_1 (1 - \rho_2) + c_2 \lambda_2 (1 - \rho_1)}, \alpha_2 = 1 - \frac{2c_1 \lambda_1}{c_1 \lambda_1 (1 - \rho_2) + c_2 \lambda_2 (1 - \rho_1)} \quad (\text{A.12})$$

which is a local minimum. For this second solution, note that

$$\alpha_1 + \alpha_2 = -2 \frac{c_1 \lambda_1 \rho_2 + c_2 \lambda_2 \rho_1}{c_1 \lambda_1 (1 - \rho_2) + c_2 \lambda_2 (1 - \rho_1)} < 0 \quad (\text{A.13})$$

which means that at least one of the two constraints is active. Suppose that $\alpha_2 = 0$. Then the following optimization problem needs to be solved:

$$\min_{\alpha_1 \geq 0} \frac{c_1 [1 + \alpha_1 \rho_2]^2 \lambda_1 (1 - \rho_1) + c_2 [1 + \alpha_1 (1 - \rho_1)]^2 \lambda_2 (1 - \rho_2)}{1 + \alpha_1 (1 - \rho_1)}. \quad (\text{A.14})$$

From $\frac{\partial J}{\partial \alpha_1} = 0$ the following equation is obtained:

$$\begin{aligned} & [c_1 \lambda_1 \rho_2^2 (1 - \rho_1) + c_2 \lambda_2 (1 - \rho_1)^2 (1 - \rho_2)] \alpha_1^2 + \dots \\ & 2 [c_1 \lambda_1 \rho_2^2 + c_2 \lambda_2 (1 - \rho_1) (1 - \rho_2)] \alpha_1 + \dots \\ & [c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2) (1 - \rho_2)] = 0. \end{aligned} \quad (\text{A.15})$$

The coefficients of α_1^2 and α_1 are both strictly positive, so this parabola in α_1 has a positive real root iff $c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2) (1 - \rho_2) < 0$. Note that this is only possible if $c_1 \lambda_1 > c_2 \lambda_2$. The constrained optimization problem thus has as a solution:

$$\alpha_1 = \begin{cases} 0 & \text{if } c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2) (1 - \rho_2) \geq 0 \\ \text{positive real root of (A.15)} & \text{otherwise.} \end{cases} \quad (\text{A.16})$$

For reasons of symmetry, if the other constraint would be active ($\alpha_1 = 0$), the following α_2 would minimize (A.11):

$$\alpha_2 = \begin{cases} 0 & \text{if } c_2 \lambda_2 (\rho_1 + \rho_2) - (c_2 \lambda_2 - c_1 \lambda_1) (1 - \rho_1) \geq 0 \\ \text{positive real root of second order polynomial} & \text{otherwise.} \end{cases} \quad (\text{A.17})$$

Since it was assumed that $c_1\lambda_1 \geq c_2\lambda_2$, (A.17) yields $\alpha_2 = 0$ and α_1 is given by (A.16). With this result, an optimal process cycle for a switching server with two product types, setup times and linear costs on buffer levels has completely been determined. Under certain conditions, as stated in Theorem 5.10, a slow-mode occurs in one of the lot types.

A.3 Proof of Proposition 5.13 (page 105)

The formal representation of the feedback control law of Proposition 5.13 is:

$$(u_0, u_1, u_2) = \begin{cases} (\textcircled{1}, \mu_1, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0 \\ (\textcircled{1}, \lambda_1, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, x_2 < x_2^\# \\ (\textcircled{2}, 0, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, x_2 \geq x_2^\# \\ (\textcircled{2}, 0, 0) & \text{if } m = 2, x_0 > 0 \\ (\textcircled{2}, 0, \mu_2) & \text{if } m = 2, x_0 = 0, x_2 > 0 \\ (\textcircled{2}, 0, \lambda_2) & \text{if } m = 2, x_0 = 0, x_2 = 0, x_1 < x_1^\# \\ (\textcircled{1}, 0, 0) & \text{if } m = 2, x_0 = 0, x_2 = 0, x_1 \geq x_1^\# \\ (\textcircled{1}, 0, 0) & \text{if } m = 1, x_0 > 0 \end{cases} \quad (\text{A.18})$$

with $x_1^\#$ and $x_2^\#$ given by (5.10).

The controller loops through Modes 1–6 of the informal feedback law description (page 105). Given an initial state, the controller starts in one of its modes. Eventually, the system reaches the same mode again. How has the state changed after one complete cycle? Based on the way the state has changed, convergence is proven.

Assume the n^{th} start of $\textcircled{2}$ (start of controller Mode 3) is from coordinate $(0, x_2^{(n)})$. From which coordinate does the $(n+1)^{\text{st}}$ start of $\textcircled{2}$ take place? Suppose that $x_2^{(n)}$ is relatively large compared to $x_2^\#$. In that case, the duration of both controller Mode 2 and Mode 5 is zero. In one cycle, the coordinates then become:

$$\begin{aligned} (0, x_2^{(n)}) &\xrightarrow{\textcircled{2}} (\lambda_1 \sigma_{12}, x_2^{(n)} + \lambda_2 \sigma_{12}) \\ &\xrightarrow{\textcircled{2}} \left(\lambda_1 \left(\sigma_{12} + \frac{x_2^{(n)} + \lambda_2 \sigma_{12}}{\mu_2 - \lambda_2} \right), 0 \right) \\ &\xrightarrow{\textcircled{1}} \left(\lambda_1 \left(\sigma + \frac{x_2^{(n)} + \lambda_2 \sigma_{12}}{\mu_2 - \lambda_2} \right), \lambda_2 \sigma_{21} \right) \\ &\xrightarrow{\textcircled{1}} \left(0, \lambda_2 \left(\sigma_{21} + \frac{\lambda_1 \left(\sigma + \frac{x_2^{(n)} + \lambda_2 \sigma_{12}}{\mu_2 - \lambda_2} \right)}{\mu_1 - \lambda_1} \right) \right) \end{aligned} \quad (\text{A.19})$$

which results in:

$$\begin{aligned} x_2^{(n+1)} &= \lambda_2 \left(\sigma_{21} + \frac{\lambda_1 \left(\sigma + \frac{x_2^{(n)} + \lambda_2 \sigma_{12}}{\mu_2 - \lambda_2} \right)}{\mu_1 - \lambda_1} \right) \\ &= \frac{\rho_1 \rho_2}{(1 - \rho_1)(1 - \rho_2)} (x_2^{(n)} - x_2^*) + x_2^*. \end{aligned} \quad (\text{A.20})$$

Note that this expression is valid for ‘large’ $x_2^{(n)}$. In case $x_2^{(n)}$ is not large compared to x_2^\sharp , either controller Mode 2 or Mode 5 has a strictly positive duration and the trajectory ends up on the optimal curve, getting:

$$x_2^{(n+1)} = x_2^\sharp. \quad (\text{A.21})$$

Combining (A.20) and (A.21) results in the difference equation:

$$x_2^{(n+1)} = \max \left(\frac{\rho_1 \rho_2}{(1 - \rho_1)(1 - \rho_2)} (x_2^{(n)} - x_2^*) + x_2^*, x_2^\sharp \right), \quad x_2^{(0)} = x_2^0 \quad (\text{A.22})$$

which has as a solution

$$x_2^{(n)} = \max \left(\left[\frac{\rho_1 \rho_2}{(1 - \rho_1)(1 - \rho_2)} \right]^n (x_2^0 - x_2^*) + x_2^*, x_2^\sharp \right). \quad (\text{A.23})$$

Since

$$0 < \frac{\rho_1 \rho_2}{(1 - \rho_1)(1 - \rho_2)} = 1 - \frac{1 - \rho_1 - \rho_2}{(1 - \rho_1)(1 - \rho_2)} < 1 \quad (\text{A.24})$$

the following result is obtained:

$$\lim_{n \rightarrow \infty} x_2^{(n)} = \max(x_2^*, x_2^\sharp) = x_2^\sharp. \quad (\text{A.25})$$

Remark A.3. In case either $x_2^* < x_2^\sharp$ (i.e. when a slow-mode occurs, $\alpha_1 > 0$) or $x_2^{(0)} < x_2^\sharp$, convergence is reached in finite time, because the trajectory reaches x_2^\sharp due to a slow-mode then. Otherwise, convergence is reached asymptotically.

A.4 Proof of Proposition 5.15 and completion of proof of Lemma 5.14 (page 109)

A formal representation of the feedback control law for a switching server with finite buffer capacities is:

$$(u_0, u_1, u_2) = \begin{cases} (\textcircled{1}, \mu_1, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0, x_2 < x_2^{\max} - \lambda_2 \sigma_{12} \\ (\textcircled{2}, 0, 0) & \text{if } m = 1, x_0 = 0, x_1 > 0, x_2 \geq x_2^{\max} - \lambda_2 \sigma_{12} \\ (\textcircled{1}, \lambda_1, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, x_2 < \bar{x}_2^\# \\ (\textcircled{2}, 0, 0) & \text{if } m = 1, x_0 = 0, x_1 = 0, x_2 \geq \bar{x}_2^\# \\ (\textcircled{2}, 0, 0) & \text{if } m = 2, x_0 > 0 \\ (\textcircled{2}, 0, \mu_2) & \text{if } m = 2, x_0 = 0, x_2 > 0, x_1 < x_1^{\max} - \lambda_1 \sigma_{21} \\ (\textcircled{1}, 0, 0) & \text{if } m = 2, x_0 = 0, x_2 > 0, x_1 \geq x_1^{\max} - \lambda_1 \sigma_{21} \\ (\textcircled{2}, 0, \lambda_2) & \text{if } m = 2, x_0 = 0, x_2 = 0, x_1 < \bar{x}_1^\# \\ (\textcircled{1}, 0, 0) & \text{if } m = 2, x_0 = 0, x_2 = 0, x_1 \geq \bar{x}_1^\# \\ (\textcircled{1}, 0, 0) & \text{if } m = 1, x_0 > 0. \end{cases} \quad (\text{A.26})$$

An informal description of this feedback law has been presented in Proposition 5.15 (page 109).

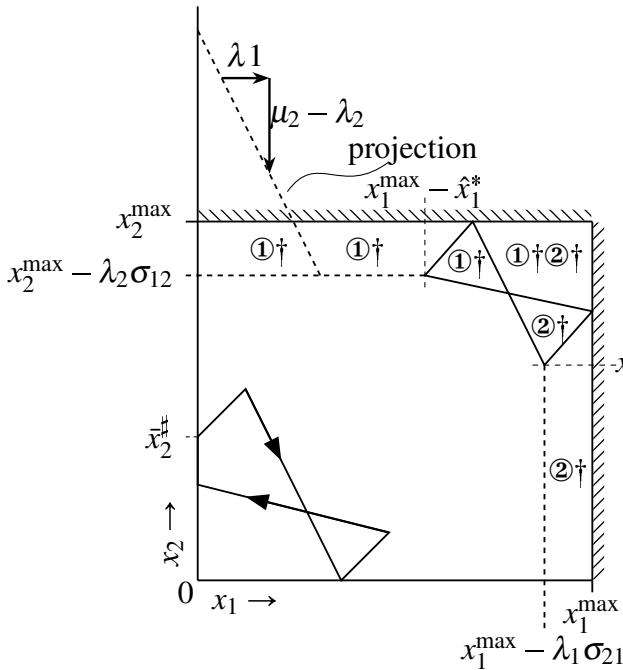


Figure A.4: Feasible and infeasible regions for trajectories, subject to buffer level constraints. (recall of Figure 5.10).

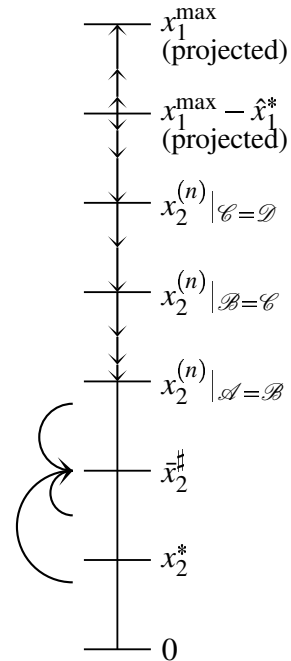


Figure A.5: Evolution of $x_2^{(n)}$ at start of $\textcircled{2}$.

Similar to the proof of the case with infinite buffer capacities, the coordinates of the $(n+1)^{\text{st}}$ start of $\textcircled{2}$ are computed given from which coordinate the n^{th} start of $\textcircled{2}$ took place. Setup for

type 2 lots can start at different places: on the axis between $(0, 0)$ and $(0, x_2^{\max} - \lambda_2 \sigma_{12})$ or on the line between $(0, x_2^{\max} - \lambda_2 \sigma_{12})$ and $(x_1^{\max}, x_2^{\max} - \lambda_2 \sigma_{12})$. The latter set of starting points of \bullet is projected onto the vertical axis, in a way that the trajectory follows the same path in the feasible (x_1, x_2) -plane for both the unconstrained and constrained situation. The projection is shown in Figure A.4. With a point $(x_1, x_2^{\max} - \lambda_2 \sigma_{12})$ the point $(0, x_2)$ is associated where x_2 is given by:

$$x_2 = \frac{\mu_2 - \lambda_2}{\lambda_1} x_1 + (x_2^{\max} - \lambda_2 \sigma_{12}) \quad (\text{A.27})$$

and vice versa. Given $x_2^{(n)}$, the next coordinate $x_2^{(n+1)}$ is looked for. In case $x_2^{(n)}$ is chosen in such a way that the trajectory is not influenced by the buffer constraints, (A.22) is obtained again. However, in case one or two buffer constraints become active, the resulting $x_2^{(n+1)}$ is larger, since switching earlier makes the system move along a line which is located higher in the (x_1, x_2) -plane, cf. Figure A.4. For the case one constraint becomes active, auxiliary variable Z is introduced, whose value depends on which constraint is active:

$$Z = \min \left(\underbrace{\frac{\mu_2 - \lambda_2}{\lambda_1} (x_1^{\max} - \lambda_1 \sigma) - \hat{x}_2^*}_{\text{is the smallest if } x_1^{\max} \text{ active}}, \underbrace{\frac{\mu_1 - \lambda_1}{\lambda_2} (x_2^{\max} - \lambda_2 \sigma) - \hat{x}_1^*}_{\text{is the smallest if } x_2^{\max} \text{ active}} \right). \quad (\text{A.28})$$

Four situations can be distinguished:

- \mathcal{A} : no active constraints, iteration as in (A.21);
- \mathcal{B} : no active constraints, iteration as in (A.19);
- \mathcal{C} : one active buffer constraint during iteration;
- \mathcal{D} : two active buffer constraints during iteration.

The endpoint of one iteration using the feedback law of Proposition 5.15 now becomes

$$x_2^{(n+1)} = \max \left(\begin{array}{ll} \bar{x}_2^\# & (\mathcal{A}) \\ \frac{\rho_1 \rho_2}{(1-\rho_1)(1-\rho_2)} (x_2^{(n)} - x_2^*) + x_2^* & (\mathcal{B}) \\ x_2^{(n)} - \frac{1-\rho_1-\rho_2}{(1-\rho_1)(1-\rho_2)} \cdot Z & (\mathcal{C}) \\ \frac{(1-\rho_1)(1-\rho_2)}{\rho_1 \rho_2} (x_2^{(n)} - [x_1^{\max} - \hat{x}_1^*]) + [x_1^{\max} - \hat{x}_1^*] & (\mathcal{D}) \end{array} \right) \quad (\text{A.29})$$

where the calligraphic capital refers to one of the four situations. The evolution of an arbitrary point $(0, x_2^{(n)})$ along the $x_1 = 0$ axis where \bullet starts can now be visualized, see Figure A.5. The arrows indicate the direction in which $x_2^{(n)}$ evolves. The distance between the arrows is a measure for the rate of the evolution.

First consider residing in region \mathcal{D} . Note that for $x_2^{(n)} > x_1^{\max} - \hat{x}_1^*$ (projected), i.e. the trajectory starts right from the upper right bow tie (see Figure A.4), the trajectory diverges, because it is known that $\frac{(1-\rho_1)(1-\rho_2)}{\rho_1 \rho_2} > 1$, cf. (A.24). When starting exactly on the upper right bow tie, the trajectory stays on it (if it is decided to switch earlier than planned by this curve, the trajectory becomes infeasible, which can be seen on geometrical grounds in Figure A.4). This completes the proof of Lemma 5.14. For $x_2^{(n)} < x_1^{\max} - \hat{x}_1^*$ the trajectory moves away from the upper right

bow tie in the correct direction, i.e. towards the bottom left in Figure A.4, and region \mathcal{D} is left after a finite number of steps. If the trajectory is in \mathcal{C} , it also moves in the desired direction with equidistant jumps and therefore after a finite number of steps leaves region \mathcal{C} . So after a finite number of steps the trajectory is either in region \mathcal{A} or in region \mathcal{B} . From the proof of Proposition 5.13, convergence to $\bar{x}_2^\#$ follows.

A.5 Proof of Proposition 5.19 (page 128)

The following sections give a detailed derivation of the expressions for the mean wip levels in a steady state process cycle for the workstation with piecewise constant arrival rates, as specified in Section 5.8. The derivation is split up into four parts:

- first type 1 lots are considered for the situation where during time span $\phi_1 P$, lots arrive at rate $\hat{\lambda}_1$, which is higher than the maximum process rate μ_1 ;
- then type 1 lots are considered for the situation where the maximum process rate is higher than the maximum arrival rate of lots;
- afterwards, expressions are derived for type 2 lots where they arrive quicker than the machine can process;
- and finally, mean wip level expressions are derived for the situation where type 2 lots can be processed quicker than they arrive.

Mean work in process level for type 1 lots, $\hat{\lambda}_1 \geq \mu_1$

The incoming lot flow starts at $t = 0$ and ends at $t = \phi_1 P$, with $0 < \phi_1 \leq 1$. Processing type 1 starts at $t = t_1$. Different situations can occur, due to overlap, as can be seen in Figure A.6. The capital letters in the figure are auxiliary variables, representing buffer levels or time instants at important points. For each situation, the mean wip level equals the area underneath the (dotted) buffer level curve, divided by the period length P . The area underneath the buffer level curve is computed by splitting up the curve into triangles, rectangles and trapezoids and summing up the individual areas. The expressions of these basic shapes can easily be recognized in the mean wip level expressions. The time span during which lots are processed is denoted by τ_1 . Since $\hat{\lambda}_1 \geq \mu_1$, lots can always be processed at maximum rate μ . Processing $\hat{\lambda}_1 \phi_1 P$ lots at this rate μ_1 takes $\bar{\rho}_1 P$ time units. For all situations I–IV, the process interval length $\tau_1 = \bar{\rho}_1 P$.

- Situation I: Domain: $0 \leq t_1 \leq \phi_1 P$. Auxiliary variables $A = t_1 \hat{\lambda}_1$ and $B = A + (\phi_1 P - t_1)(\hat{\lambda}_1 - \mu_1)$. The mean wip level \bar{w}_1 becomes:

$$\begin{aligned} \bar{w}_1(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot t_1 \cdot A + (\phi_1 P - t_1) \cdot \frac{A+B}{2} + \frac{1}{2} \cdot (t_1 + \tau_1 - \phi_1 P) \cdot B \right] \\ &= \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 \bar{\rho}_1 t_1. \end{aligned}$$

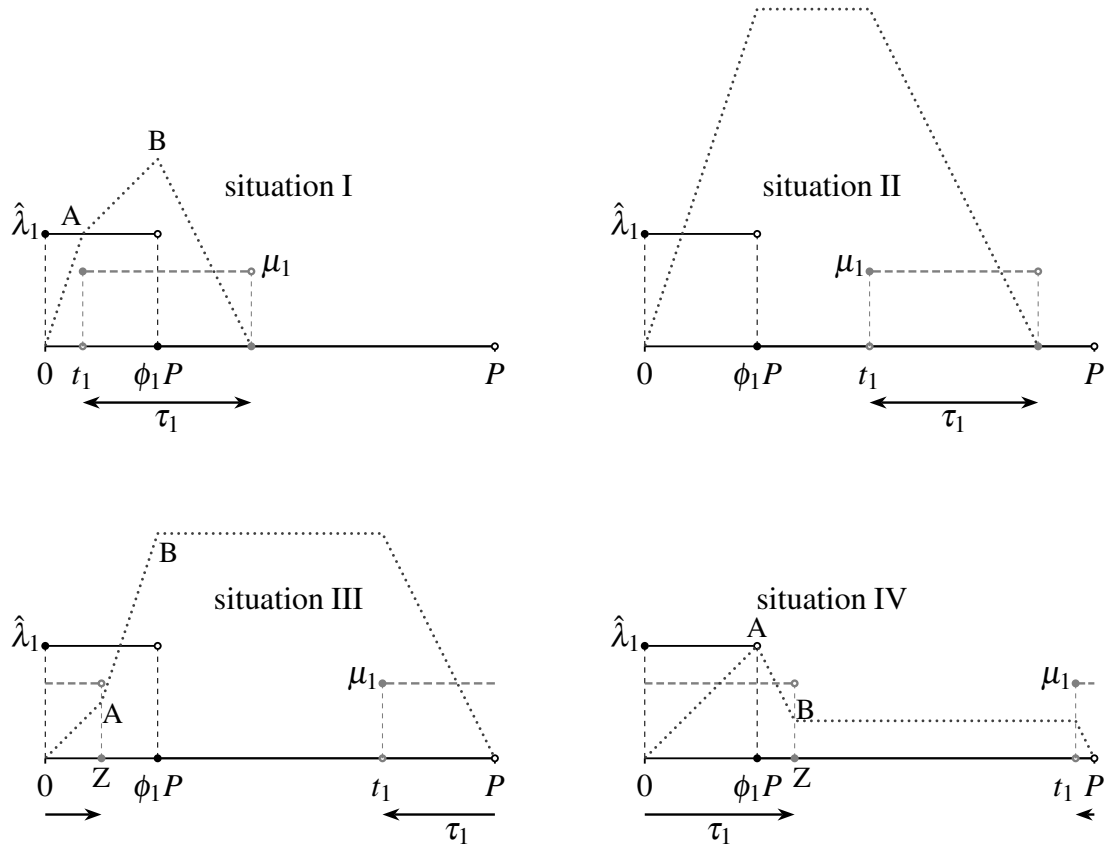


Figure A.6: Different situations for type 1 and $\hat{\lambda}_1 \geq \mu_1$. Input rate profile (solid), process rate profile (dashed gray) and buffer level curve (dotted).

Note that the first term (constant) in the mean wip level expression is positive ($\bar{\rho}_1 \geq \phi_1$) since $\hat{\lambda}_1 \geq \mu_1$ and $\bar{\rho}_1 = \phi_1 \hat{\lambda}_1 / \mu_1$.

- Situation II: Domain: $\phi_1 P \leq t_1 \leq (1 - \bar{\rho}_1)P$. The mean wip level \bar{w}_1 becomes:

$$\begin{aligned} \bar{w}_1(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot \phi_1 P \cdot \hat{\lambda}_1 \phi_1 P + (t_1 - \phi_1 P) \cdot \phi_1 P \hat{\lambda}_1 + \frac{1}{2} \cdot \tau_1 \cdot \phi_1 P \hat{\lambda}_1 \right] \\ &= \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 \bar{\rho}_1 t_1. \end{aligned}$$

- Situation III: Domain: $(1 - \bar{\rho}_1)P \leq t_1 \leq (1 + \phi_1 - \bar{\rho}_1)P$.
Auxiliary variables $Z = (\bar{\rho}_1 - 1)P + t_1$, $A = Z(\hat{\lambda}_1 - \mu_1)$ and $B = \phi_1 P \hat{\lambda}_1 - Z\mu_1$. Note that A and B are amounts of lots and Z is a time instant. The mean wip level \bar{w}_1 becomes:

$$\begin{aligned} \bar{w}_1(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot Z \cdot A + (\phi_1 P - Z) \cdot \frac{A+B}{2} + (t_1 - \phi_1 P) \cdot B + \frac{1}{2} \cdot (P - t_1) \cdot B \right] \\ &= \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 (1 - \bar{\rho}_1) (P - t_1). \end{aligned}$$

- Situation IV: Domain: $(1 + \phi_1 - \bar{\rho}_1)P \leq t_1 \leq P$.
Auxiliary variables $A = \phi_1 P \hat{\lambda}_1$, $Z = (\bar{\rho}_1 - 1)P + t_1$ and $B = A - (Z - \phi_1 P)\mu_1$. Again, Z

represents a time instant, whereas A and B represent amounts of jobs. The mean wip level \bar{w}_1 becomes:

$$\begin{aligned}\bar{w}_1(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot \phi_1 P \cdot A + (Z - \phi_1 P) \cdot \frac{A+B}{2} + (t_1 - Z) \cdot B + \frac{1}{2} \cdot (P - t_1) \cdot B \right] \\ &= \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 (1 - \bar{\rho}_1) (P - t_1).\end{aligned}$$

It can easily be verified that at the boundaries of adjacent intervals, the mean wip level values are equal. Situations I and II have the same expressions for the mean wip level and process interval τ_1 . Situations III and IV also have the same expressions for mean wip level and process interval. The domains of these situations can therefore be unified.

Summarizing, the mean wip level and process length interval expressions for type 1 lots and maximum arrival rate $\hat{\lambda}_1 \geq \mu_1$ are:

$$\bar{w}_1(t_1) = \begin{cases} \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 \bar{\rho}_1 t_1 & \text{for } 0 \leq t_1 \leq (1 - \bar{\rho}_1)P \\ \frac{1}{2} \mu_1 \bar{\rho}_1 P (\bar{\rho}_1 - \phi_1) + \mu_1 (1 - \bar{\rho}_1) (P - t_1) & \text{for } (1 - \bar{\rho}_1)P \leq t_1 \leq P \end{cases}$$

$$\tau_1 = \bar{\rho}_1 P.$$

Remark A.4. It is possible to stay in mode 1 longer than $\sigma_{21} + \bar{\rho}_1 P$. The workstation is then idling, processing no jobs. This is discussed when optimization takes place over both lot types. Instead of requiring that the sum of the setup times and process interval lengths equals period length P , it is required that this sum is less than or equal to the period length P . This inequality facilitates the possibility of idling in a mode after having completed the jobs.

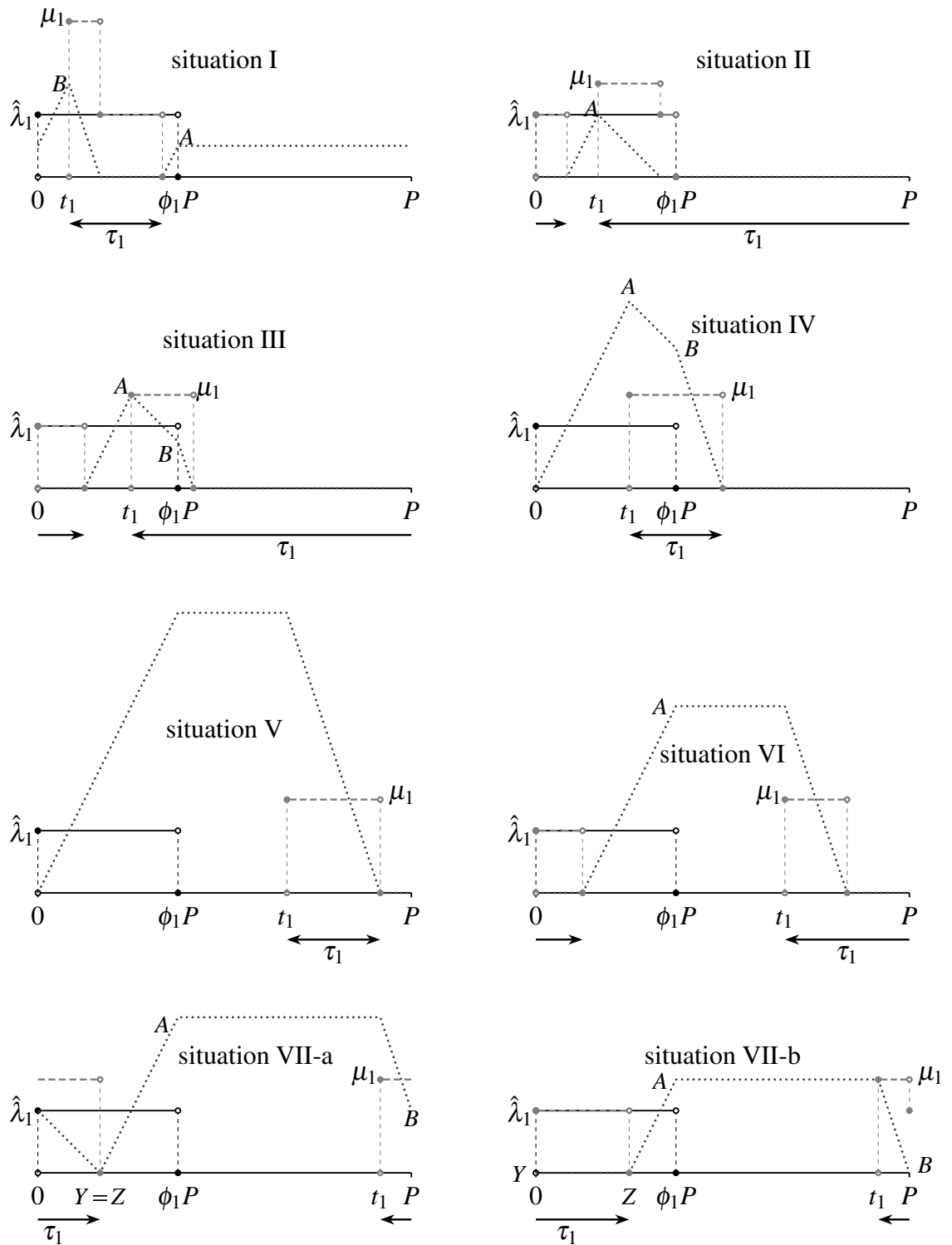


Figure A.7: Different situations for type 1 and $\hat{\lambda}_1 < \mu_1$. Solid line: input rate profile. Gray dashed line: process rate profile. Dotted line: buffer level.

Mean work in process level for type 1 lots, $\hat{\lambda}_1 < \mu_1$

When the process rate of type 1 lots is greater than the arrival rate, a slow-mode may occur, depending on the start time of the process interval. It is even possible to make a choice at certain time instants: process at maximum rate or at arrival rate. The different situations that may occur are depicted in Figure A.7. All situations are described below. For each situation, the mean wip level has been plotted against the start time of processing type 1 jobs, t_1 , and the duration of this process interval, τ_1 . In order to make these plots, the following parameters have been used: $\hat{\lambda}_1 = 2$, $\mu_1 = 3$, $\phi_1 = \frac{3}{8}$ and $P = 12$. This results in $\bar{\rho}_1 = \frac{1}{4}$.

- Situation I: Domain: $0 \leq t_1 \leq (\phi_1 - \bar{\rho}_1)P$ and $\bar{\rho}_1 P \leq \tau_1 \leq \phi_1 P - t_1$. A slow-mode may occur. Auxiliary variable A denotes the buffer level for $t > \phi_1 P$, $A = (P - (t_1 + \tau_1))\hat{\lambda}_1$ and B represents the top buffer level, $B = A + t_1\hat{\lambda}_1$. The mean wip level can be computed:

$$\begin{aligned} \bar{w}_1^I(t_1, \tau_1) &= \frac{1}{P} \left[t_1 \cdot \frac{A+B}{2} + \frac{1}{2} \cdot \frac{B}{\mu_1 - \hat{\lambda}_1} \cdot B + \frac{1}{2} \cdot (\phi_1 P - (t_1 + \tau_1)) \cdot A + (P - \phi_1 P) \cdot A \right] \\ &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} \left[\phi_1 \tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1 \bar{\rho}_1)\tau_1 - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 \right. \\ &\quad \left. + \phi_1 P^2(2(\phi_1 - \bar{\rho}_1)(1 - \phi_1) + \phi_1^2) \right]. \end{aligned}$$

This function is investigated further to determine whether it is a convex function. Convex functions have a global optimum, which can be determined efficiently in optimization routines. The Hessian matrix of a function $f(x, y)$ is defined as:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}.$$

For this mean wip level function $\bar{w}_1^I(t_1, \tau_1)$, the Hessian matrix \mathbf{H}^I is:

$$\mathbf{H}^I = \begin{bmatrix} 0 & 0 \\ 0 & \frac{\mu_1 \bar{\rho}_1}{(\phi_1 - \bar{\rho}_1)P} \end{bmatrix}$$

which is always positive semi-definite, since $\phi_1 \geq \bar{\rho}_1$. This means that the mean wip level function is a convex function, which is an important and convenient property for the optimizations that are to be performed. The mean wip level has been plotted for the feasible domain. In Figures A.8 and A.9 a three-dimensional plot and a contourplot are shown respectively. The gray shaded area shows the feasible domain. The black lines in the contourplot are lines with equal wip level (isolines) and the color gradient shows in which direction the mean wip level decreases (darker) or increases (lighter).

- Situation II: Domain: $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \phi_1 P$ and $P - t_1 \leq \tau_1 \leq P$ and $\tau_1 \geq \bar{\rho}_1 P$. A process interval that was started during the arrivals of type 1 lots may take so long that during the

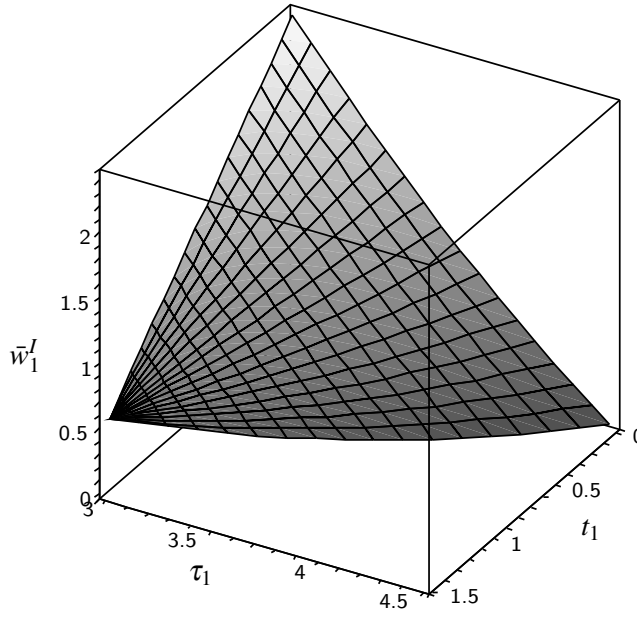


Figure A.8: Surface plot of mean wiplevel \bar{w}_1^I against t_1 and τ_1 (situation I).

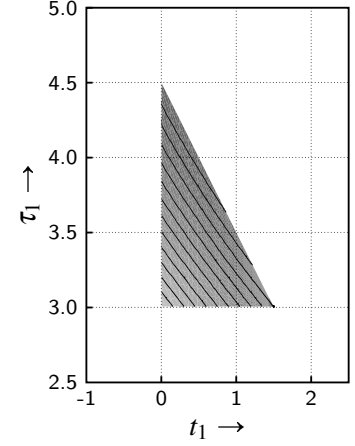


Figure A.9: Contourplot of \bar{w}_1^I in the (t_1, τ_1) -plane (situation I).

process interval, the next arrival period of type 1 jobs starts. At $t = t_1$, jobs are processed (possibly at maximum rate) and the buffer is emptied before $\phi_1 P$. A slow-mode may occur then. When the arrivals of lots come to an end, the machine idles until the arrivals start over again. From that point ($t = P$), the machine processes lots in slow-mode until the required amount of lots in one process cycle has been completed. Auxiliary variable A denotes the buffer top level, $A = (P - \tau_1)\hat{\lambda}_1$. An additional condition for this situation to guarantee that the buffer is empty before $t = \phi_1 P$ is that $(P - \tau_1)\hat{\lambda}_1 \leq (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)$. This can be rewritten as: $\tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left[1 - \phi_1\left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)\right]P$.

$$\begin{aligned}\bar{w}_1^H(\tau_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot \left(P - \tau_1 + \frac{A}{\mu_1 - \hat{\lambda}_1} \right) \cdot A \right] \\ &= \frac{\mu_1 \bar{\rho}_1 (P - \tau_1)^2}{2P(\phi_1 - \bar{\rho}_1)}.\end{aligned}$$

Note that the mean wip level is independent of starting point t_1 , within the domain for t_1 . The Hessian matrix for this mean wip level function, \mathbf{H}^H , equals the Hessian matrix of situation I, so this function is also convex. The mean wip level function has been plotted in Figures A.10 and A.11.

- Situation III: Domain: $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \phi_1 P$ and $\tau_1 \geq \bar{\rho}_1 P$. Notice the overlap in domain of t_1 with situation II. The only difference is that now the buffer cannot be emptied before time instant $t = \phi_1 P$. The number of lots that flows into the buffer when it is not processing type 1 lots, $\hat{\lambda}_1(P - \tau_1)$, must be greater than (or equal to) the number of jobs that can be processed before $t = \phi_1 P$, i.e. $(\mu_1 - \hat{\lambda}_1)(\phi_1 P - t_1)$ so this situation has the

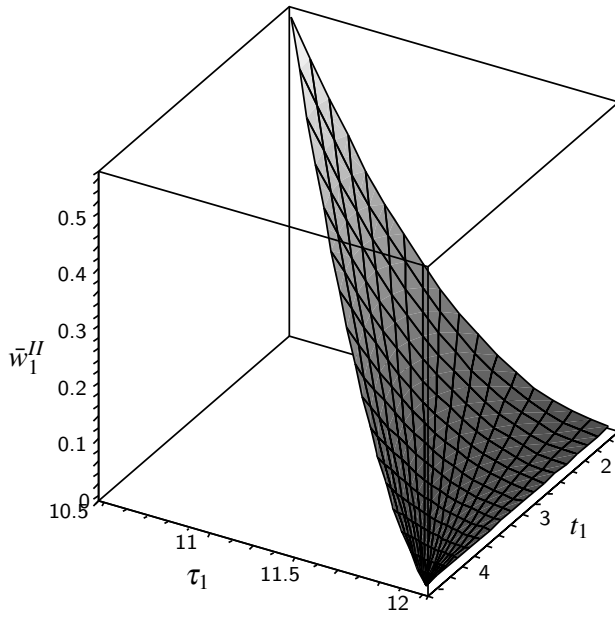


Figure A.10: Surface plot of mean wiplevel \bar{w}_1^{II} against t_1 and τ_1 (situation II).

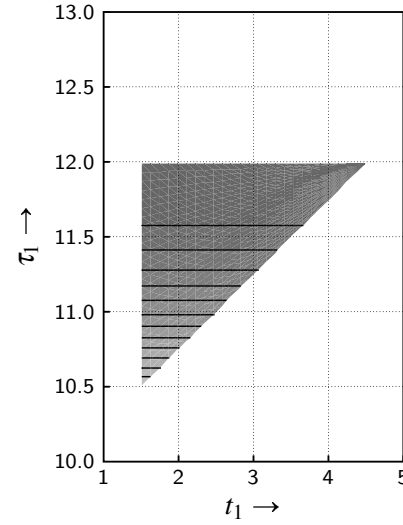


Figure A.11: Contourplot of \bar{w}_1^{II} in the (t_1, τ_1) -plane (situation II).

complementary additional constraint:

$$\tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left[1 - \phi_1 \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) \right] P.$$

Another upper bound on the process time interval exists. Before $t = \phi_1 P$, the number of lots that has been processed is $(\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)$. The total number of jobs that has to be processed during one cycle equals $\mu_1 \bar{\rho}_1 P$. The number of lots that has to be processed when the arrivals start again therefore equals $\mu_1 \bar{\rho}_1 P - (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)$. Maximum duration of processing these lots is completely in slow-mode. This takes another $(\mu_1 \bar{\rho}_1 P - (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)) / \hat{\lambda}_1$ time units. The total process interval τ_1 then has as lower bound:

$$\begin{aligned} \tau_1 &\leq P - t_1 + \frac{\mu_1 \bar{\rho}_1 P - (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)}{\hat{\lambda}_1} \\ &\leq \left(\frac{\phi_1}{\bar{\rho}_1} - 2 \right) t_1 + \left(1 + 2\phi_1 - \frac{\phi_1^2}{\bar{\rho}_1} \right) P. \end{aligned}$$

Auxiliary variable A denotes the top buffer level: $A = (P - \tau_1)\hat{\lambda}_1$ and variable B denotes the buffer level at $t = \phi_1 P$: $B = A - (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)$. In the expressions for these variables, the additional constraints for situations II and III can be recognized. The expressions for the mean wip level now becomes:

$$\begin{aligned} \bar{w}_1^{III}(t_1, \tau_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot (P - \tau_1) \cdot A + (\phi_1 P - t_1) \cdot \frac{A + B}{2} + \frac{1}{2} \cdot \frac{B}{\mu_1} \cdot B \right] \\ &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1^2 P} \left[((1 + \phi_1)P - \tau_1 - t_1) ((\phi_1 - \bar{\rho}_1)t_1 - (\phi_1 + \bar{\rho}_1)\tau_1 \right. \\ &\quad \left. + P(\bar{\rho}_1 + \phi_1 + \bar{\rho}_1\phi_1 - \phi_1^2)) \right]. \end{aligned}$$

The mean wip level function has been plotted in Figures A.12 and A.13. Further analysis of this situation (and situation VI) is carried out after the presentation of all situations.

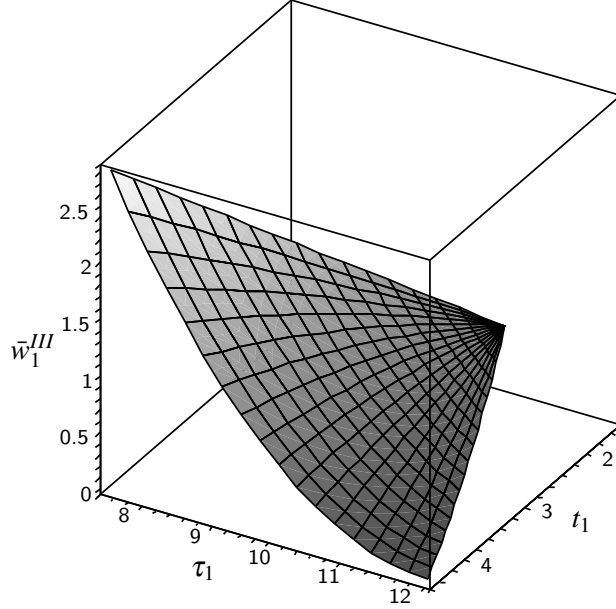


Figure A.12: Surface plot of mean wiplevel \bar{w}_1^{III} against t_1 and τ_1 (situation III).

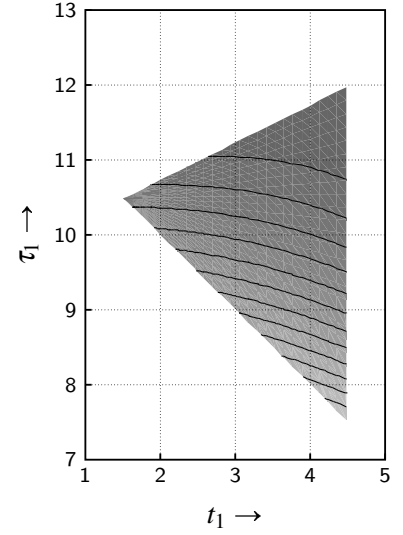


Figure A.13: Contourplot of \bar{w}_1^{III} in the (t_1, τ_1) -plane (situation III).

- Situation IV: Domain: $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \phi_1 P$. Auxiliary variable $A = t_1 \hat{\lambda}_1$ denotes the top buffer level and $B = A - (\phi_1 P - t_1)(\mu_1 - \hat{\lambda}_1)$ is the buffer level at $t = \phi_1 P$. The mean wip level is:

$$\begin{aligned} \bar{w}_1^{IV}(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot t_1 \cdot A + (\phi_1 P - t_1) \cdot \frac{A+B}{2} + \frac{1}{2} \cdot (t_1 + \bar{\rho}_1 P - \phi_1 P) \cdot B \right] \\ &= -\frac{1}{2} \mu_1 \bar{\rho}_1 P (\phi_1 - \bar{\rho}_1) + \mu_1 \bar{\rho}_1 t_1 \\ \tau_1 &= \bar{\rho}_1 P \end{aligned}$$

which is a linear (i.e. convex) function in t_1 .

- Situation V: Domain: $\phi_1 P \leq t_1 \leq (1 - \bar{\rho}_1)P$. The process interval starts after the complete arrivals interval and if processed at maximum rate for the whole process interval, it ends before the period P is over. No slow-mode occurs. The mean wip level $\bar{w}_1(t_1)$ is:

$$\begin{aligned} \bar{w}_1^V(t_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot (\phi_1 P + \bar{\rho}_1 P) \cdot \phi_1 P \hat{\lambda}_1 + (t_1 - \phi_1 P) \cdot \phi_1 P \hat{\lambda}_1 \right] \\ &= -\frac{1}{2} \mu_1 \bar{\rho}_1 P (\phi_1 - \bar{\rho}_1) + \mu_1 \bar{\rho}_1 t_1 \\ \tau_1 &= \bar{\rho}_1 P \end{aligned}$$

which is the same convex linear function as in situation IV. The mean wip level for situations IV and V is plotted in Figure A.14.

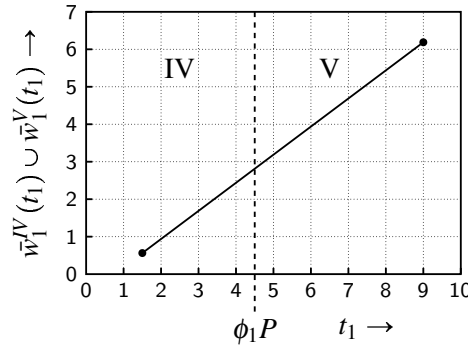


Figure A.14: Mean wip levels \bar{w}_1^{IV} and \bar{w}_1^V over their united domains.

- Situation VI: Domain: $\phi_1 P \leq t_1 \leq P$ and $P - t_1 \leq \tau_1 \leq (1 + \phi_1)P - t_1$ and $\tau_1 \geq \bar{\rho}_1 P$. Notice that an overlap exists with the domain of situation V. Another realization of the process rate profile is to use a slow-mode when the arrivals interval starts again. At $t = t_1$ a small number of lots is processed at rate μ_1 until the buffer is empty. Then the machine idles (slow-mode with arrival rate 0) until the arrival of new lots starts again. From that point ($t = P$) the machine processes lots at their arrival rate $\hat{\lambda}_1$. In a limit case, the machine starts with idling and processes all incoming lots in slow-mode. The idling period before processing of lots starts can be used by the other type, but is blocked in this situation. However, the same process rate profile is possible without the idling, in situation I. So if losing the idling period in front of the process interval results in a lower mean wip level, the evaluation of situation I reveals this. The duration of the slowmode in situation V is $\tau_1^\lambda = \tau_1 - (P - t_1)$. The length of this slow-mode has a lower bound. The time before the arrivals start again, $P - t_1$ provides a maximum number of lots that can be processed within that time span: $\mu_1(P - t_1)$. Each cycle, a total number of $\bar{\rho}_1 P$ lots has to be processed. This means the duration of the slow-mode is at least the difference of these two quantities divided by the arrival rate $\hat{\lambda}_1$. This constraint thus becomes:

$$\begin{aligned} \tau_1 &\geq P - t_1 + \frac{\mu_1(t_1 - (1 - \bar{\rho}_1)P)}{\hat{\lambda}_1} \\ &\geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1}\right)P. \end{aligned}$$

Auxiliary variable $A = (\phi_1 P - \tau_1^\lambda)\hat{\lambda}_1$ denotes the top buffer level for a given t_1 and τ_1 . The mean wip level for this situation is:

$$\begin{aligned} \bar{w}_1^{VI}(t_1, \tau_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot (\phi_1 P - \tau_1^\lambda) \cdot A + (t_1 - \phi_1 P) \cdot A + \frac{1}{2} \cdot \frac{A}{\mu_1} \cdot A \right] \\ &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1^2 P} \left[((1 + \phi_1)P - \tau_1 - t_1)((\phi_1 - \bar{\rho}_1)t_1 - (\phi_1 + \bar{\rho}_1)\tau_1 \right. \\ &\quad \left. + P(\bar{\rho}_1 + \phi_1 + \bar{\rho}_1\phi_1 - \phi_1^2)) \right]. \end{aligned}$$

This mean wip level function has been plotted in Figures A.15 and A.16. Further analysis of this situation is given after presentation of all situations.

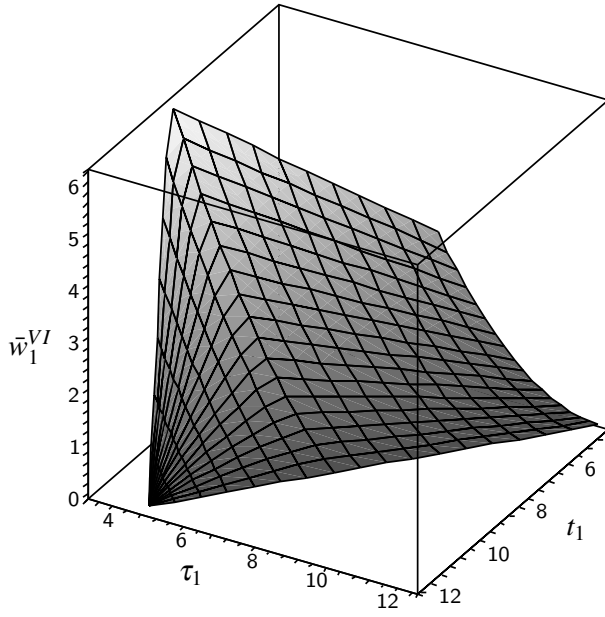


Figure A.15: Surface plot of mean wiplevel \bar{w}_1^{VI} against t_1 and τ_1 (situation VI).

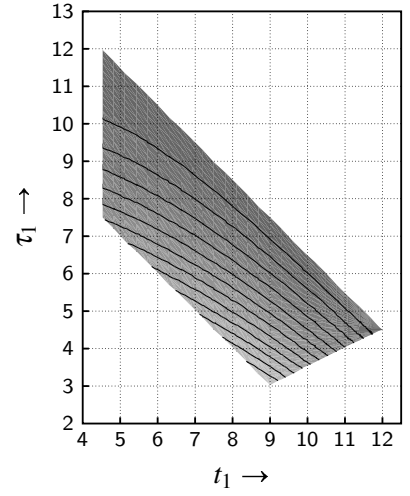


Figure A.16: Contourplot of \bar{w}_1^{VI} in the (t_1, τ_1) -plane (situation VI).

- Situation VII: Domain: $(1 - \bar{\rho}_1)P \leq t_1 < P$ and $\bar{\rho}_1 P \leq \tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1\right)t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1}\right)P$. Again notice that an overlap exists with the domain of situation VI. In situation VI, processing type 1 lots starts towards the end of period P and all lots cannot be processed before P is over. Until P is reached, lots are processed at maximum process rate. The remainder of lots has to be processed while new lots arrive. Again, a choice is possible: using a slow-mode or not using a slow-mode. Two limit realizations are shown in Figure A.7 in situations VII-a and VII-b. The former does not use a slow-mode ($\tau_1 = \bar{\rho}_1 P$), the latter uses maximum slow-mode duration ($\tau_1 = P - t_1 + (\hat{\lambda}_1 \phi_1 P - \mu_1(P - t_1))/\hat{\lambda}_1$). All other durations of slowmode between these limits are possible realizations. Auxiliary variables are: $Y = B/(\mu_1 - \hat{\lambda}_1)$ (start time of slowmode), $Z = \tau_1 - (P - t_1)$ (end time of slowmode), $A = (\phi_1 P - Z)\hat{\lambda}_1$ (top buffer level) and $B = A - (P - t_1)\mu_1$ (buffer level at $t = 0$). The mean wip level depends on the start time of processing t_1 :

$$\begin{aligned} \bar{w}_1^{VII}(t_1, \tau_1) &= \frac{1}{P} \left[\frac{1}{2} \cdot Y \cdot B + \frac{1}{2} \cdot (\phi_1 P - Z) \cdot A + (t_1 - \phi_1 P) \cdot A + (P - t_1) \cdot \frac{A + B}{2} \right] \\ &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} \left[\phi_1 \tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1 \bar{\rho}_1)\tau_1 \right. \\ &\quad \left. - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 - P^2(2\bar{\rho}_1 - 2\phi_1 + \phi_1^3 - 2\phi_1^2 \bar{\rho}_1) \right]. \end{aligned}$$

The Hessian matrix of this mean wip level function, \mathbf{H}^{VII} is equal to the Hessians of situations I and II, which is positive semi-definite. Therefore, \bar{w}_1^{VII} is a convex function.

Situations IV and V have the same expressions for the mean wip level and the process interval length, so they can be combined with domain $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \max(\phi_1 P, (1 - \bar{\rho}_1)P)$.

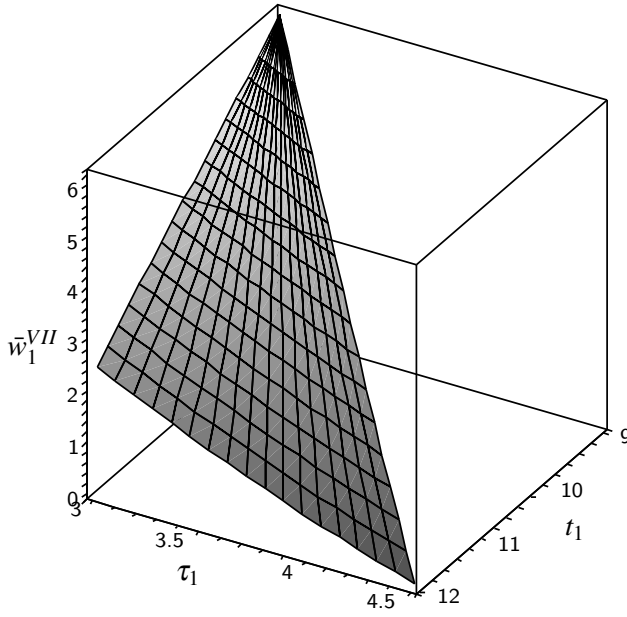


Figure A.17: Surface plot of mean wip level \bar{w}_1^{VII} against t_1 and τ_1 (situation VII).

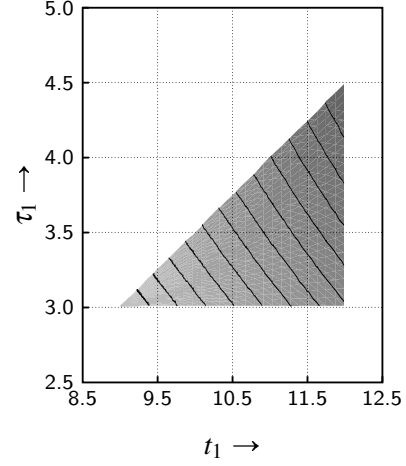


Figure A.18: Contourplot of \bar{w}_1^{VII} in the (t_1, τ_1) -plane (situation VII).

If $(1 - \bar{\rho}_1)P < \phi_1 P$ then situation IV does not occur (empty domain). Since $\phi_1 \leq 1$, $\mu_1 \geq \hat{\lambda}_1$ and $\hat{\lambda}_1 = \bar{\rho}_1 \mu_1 / \phi_1$, it is never possible that situation IV overlaps the domain of situation I, because it follows that $(\phi_1 - \bar{\rho}_1)P \leq (1 - \bar{\rho}_1)P$.

Situations III and VI also have the same expressions for the mean wip level, so it is useful to investigate a possible merge of these situations. The mean wip level expression is a hyperbolic paraboloid in t_1 and τ_1 , which has a saddle point at $t_1 = \phi_1 P$ and $\tau_1 = P$. The Hessian matrix $\mathbf{H}^{III} = \mathbf{H}^{VI}$ is given by:

$$\mathbf{H}^{III} = \mathbf{H}^{VI} = \begin{bmatrix} \frac{\mu_1 \bar{\rho}_1 (\bar{\rho}_1 - \phi_1)}{\phi_1^2 P} & \frac{\mu_1 \bar{\rho}_1^2}{\phi_1^2 P} \\ \frac{\mu_1 \bar{\rho}_1^2}{\phi_1^2 P} & \frac{\mu_1 \bar{\rho}_1 (\bar{\rho}_1 + \phi_1)}{\phi_1^2 P} \end{bmatrix}$$

which is not positive (semi)-definite. Maybe, the function is convex in its feasible domain. To determine whether this is the case, the eigenvalues e_1, e_2 and eigenvectors $\mathbf{e}_1, \mathbf{e}_2$ belonging to the mean wip level function are derived:

$$e_1 = \frac{\mu_1 \bar{\rho}_1 (\bar{\rho}_1 + \sqrt{\phi_1^2 + \bar{\rho}_1^2})}{\phi_1^2 P} \quad \mathbf{e}_1 = \begin{bmatrix} 1 & \frac{\phi_1 + \sqrt{\phi_1^2 + \bar{\rho}_1^2}}{\bar{\rho}_1} \end{bmatrix}^T$$

$$e_2 = \frac{\mu_1 \bar{\rho}_1 (\bar{\rho}_1 - \sqrt{\phi_1^2 + \bar{\rho}_1^2})}{\phi_1^2 P} \quad \mathbf{e}_2 = \begin{bmatrix} 1 & \frac{\phi_1 - \sqrt{\phi_1^2 + \bar{\rho}_1^2}}{\bar{\rho}_1} \end{bmatrix}^T.$$

Note that the eigenvectors of the mean wip level function are always perpendicular to each other: $\mathbf{e}_1 \cdot \mathbf{e}_2 = 0$. The slope of \mathbf{e}_1 is greater than 1, while the slope of vector \mathbf{e}_2 is smaller than zero. To check the convexity of the mean wip level function in its feasible domain, first the borders of these domains are examined further. The domains of t_1 can easily be merged, since

they have a common boundary point $\phi_1 P$. The constraints on τ_1 for situation III are:

$$\tau_1 \geq \bar{\rho}_1 P \quad (\text{A.30})$$

$$\tau_1 \geq P - t_1 \quad (\text{A.31})$$

$$\tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1^2}{\bar{\rho}_1} \right) P \quad (\text{A.32})$$

$$\tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 2 \right) t_1 + \left(1 + 2\phi_1 - \frac{\phi_1^2}{\bar{\rho}_1} \right) P \quad (\text{A.33})$$

$$\tau_1 \leq P. \quad (\text{A.34})$$

The constraints on τ_1 for situation VI are:

$$\tau_1 \geq \bar{\rho}_1 P \quad (\text{A.35})$$

$$\tau_1 \geq P - t_1 \quad (\text{A.36})$$

$$\tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1} \right) P \quad (\text{A.37})$$

$$\tau_1 \leq (1 + \phi_1)P - t_1 \quad (\text{A.38})$$

$$\tau_1 \leq P. \quad (\text{A.39})$$

Note that the first two and final constraints appear in both situations and that (A.37) is *not* the same as (A.32). The intersection of constraints (A.32), (A.33) and (A.34) is at $t_1 = \phi_1 P$. Since this is the border of the domain of situation III and (A.32) lies below (A.33) for $t_1 < \phi_1 P$, the consequence is that (A.33) is only active at $t = \phi_1 P$, together with (A.32). The intersection of constraints (A.38) and (A.39) with (A.32), (A.33) and (A.34) also lies at $t_1 = \phi_1 P$.

Since $\phi_1 \geq \bar{\rho}_1$, constraint (A.38) is always active on the domain of situation VI. For the lower bounds, constraints (A.30), (A.31), (A.35), (A.36) and (A.37) have a common intersection point at $t_1 = (1 - \bar{\rho}_1)P$. The slopes of these constraints indicate that the constraints are never active outside the domains of their own situation. A graphical representation of all constraints is given in Figure A.19, with the same parameter values as before. The gray shaded area represents the feasible area. An important observation is that constraint line (A.38) is a line where wip level $\bar{w}_1(t_1, \tau_1) = 0$. The two lines in the hyperbolic paraboloid for which the mean wip level is 0 are:

$$\begin{aligned} \tau_1 &= (1 + \phi_1)P - t_1 \\ \tau_1 &= \frac{\phi_1 - \bar{\rho}_1}{\phi_1 + \bar{\rho}_1} t_1 - \frac{\phi_1^2 - \phi_1 - \bar{\rho}_1 \phi_1 - \bar{\rho}_1}{\phi_1 + \bar{\rho}_1} P. \end{aligned}$$

As mentioned, the first equation is the border of constraint (A.38). The second equation of course also crosses the saddle point $(\phi_1 P, P)$ and has a slope which is always lower than the slope of constraint (A.32), which proves that the mean wip level is only zero at the border of the feasible area which is defined by constraint (A.38). The slope of eigenvector \mathbf{e}_1 is greater than the slope of constraint (A.32), which means that the mean wip level is not a convex function over the feasible area. Special attention needs to be given to this issue in the optimization of

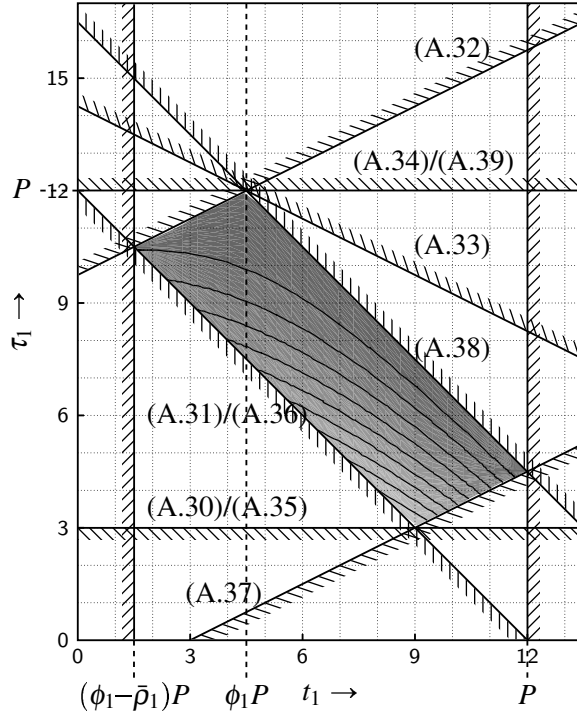


Figure A.19: Unification of situations III and VI is justified. Saddle point of the hyperbolic paraboloid is the top point of the feasible area.

the whole workstation with two lot types. Finally, it can be concluded that situations III and VI can be merged to one situation, with the extended domain for t_1 : $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq P$ and all constraints (A.30)–(A.39).

Situations I and VII have almost the same expression for the mean wip level, except for the constant term. This constant term does not influence solution of an optimization problem. Therefore, it would be possible to join situations I and VII. However, two issues have to be kept in mind then. First, depending on the solution of the optimization problem, one should compensate for the constant term to obtain the correct mean wip level and second, the domains of the two situations cannot be merged, since the constraints of both situations rule out each other. The domain of t_1 however can be shifted down one period length P to obtain a feasible area, immediately solving the issue of the constant factor difference, but then the domain of the design variable t_1 is different than for all other optimization subproblems. Since the mean wip level is a convex function which can easily be solved in polynomial time (even by hand!), the choice is made not to combine situations I and VII. The optimization problem remains transparent and insightful in this way.

For some parameter settings, it is easy to verify that the mean wip level must equal zero. In addition, at the borders of the domains of the situations, the mean wip levels must be equal, since they correspond to the same physical situation. The following properties are easy to verify (an intuitive check can be performed with Figure A.7):

$$\begin{aligned}
\bar{w}_1^I|_{t_1=0, \tau_1=\phi_1 P} &= 0 \\
\bar{w}_1^{II}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P} &= 0 \\
\bar{w}_1^{VI}|_{t_1=P, \tau_1=\phi_1 P} &= 0 \\
\bar{w}_1^{VII}|_{t_1=P, \tau_1=\phi_1 P} &= 0 \\
\bar{w}_1^I|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=\bar{\rho}_1 P} &= \bar{w}_1^{II}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P-t_1} \\
\bar{w}_1^{III}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P-t_1} &= \bar{w}_1^{II}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P-t_1} \\
\bar{w}_1^{IV}|_{t_1=(\phi_1-\bar{\rho}_1)P} &= \bar{w}_1^{III}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P-t_1} \\
\bar{w}_1^{IV}|_{t_1=(\phi_1-\bar{\rho}_1)P} &= \bar{w}_1^{II}|_{t_1=(\phi_1-\bar{\rho}_1)P, \tau_1=P-t_1} \\
\bar{w}_1^{IV}|_{t_1=\phi_1 P} &= \bar{w}_1^V|_{t_1=\phi_1 P} \\
\bar{w}_1^V|_{t_1=(1-\bar{\rho}_1)P, \tau_1=\bar{\rho}_1 P} &= \bar{w}_1^{VI}|_{t_1=(1-\bar{\rho}_1)P, \tau_1=P-t_1} \\
\bar{w}_1^{VI}|_{t_1=(1-\bar{\rho}_1)P, \tau_1=\rho_1 P} &= \bar{w}_1^{VII}|_{t_1=(1-\bar{\rho}_1)P, \tau_1=\rho_1 P}.
\end{aligned}$$

The first four properties show that the mean wip level can become zero in these cases, contrary to the situation where $\hat{\lambda} \geq \mu$, where the mean wip level in general does not equal zero.

Summarizing, for type 1 lots and $\hat{\lambda}_1 \leq \mu_1$, all possible situations and their mean wip level expressions can be stated as:

$$\begin{aligned}
\bar{w}_1^I(t_1, \tau_1) &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} \left[\phi_1 \tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1 \bar{\rho}_1) \tau_1 \right. \\
&\quad \left. - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 + \phi_1 P^2(2(\phi_1 - \bar{\rho}_1)(1 - \phi_1) + \phi_1^2) \right] \\
&\text{for } 0 \leq t_1 \leq (\phi_1 - \bar{\rho}_1)P \text{ and } \bar{\rho}_1 P \leq \tau_1 \leq \phi_1 P - t_1
\end{aligned}$$

$$\begin{aligned}
\bar{w}_1^{II}(\tau_1) &= \frac{\mu_1 \bar{\rho}_1 (P - \tau_1)^2}{2P(\phi_1 - \bar{\rho}_1)} \\
&\text{for } (\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \phi_1 P \text{ and } P - t_1 \leq \tau_1 \leq P \text{ and } \tau_1 \geq \bar{\rho}_1 P \\
&\text{and } \tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1^2}{\bar{\rho}_1} \right) P
\end{aligned}$$

$$\begin{aligned}
\bar{w}_1^{III-VI}(t_1, \tau_1) &= \frac{\mu_1 \bar{\rho}_1}{2\phi_1^2 P} \left[((1 + \phi_1)P - \tau_1 - t_1)((\phi_1 - \bar{\rho}_1)t_1 \right. \\
&\quad \left. - (\phi_1 + \bar{\rho}_1)\tau_1 + P(\bar{\rho}_1 + \phi_1 + \bar{\rho}_1 \phi_1 - \phi_1^2)) \right] \\
&\text{for } \tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1^2}{\bar{\rho}_1} \right) P \text{ and } \tau_1 \geq P - t_1 \\
&\text{and } \tau_1 \geq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1} \right) P \text{ and } \tau_1 \leq (1 + \phi_1)P - t_1
\end{aligned}$$

$$\bar{w}_1^{IV-V}(t_1) = -\frac{1}{2}\mu_1 \bar{\rho}_1 P(\phi_1 - \bar{\rho}_1) + \mu_1 \bar{\rho}_1 t_1 \text{ and } \tau_1 = \bar{\rho}_1 P$$

for $(\phi_1 - \bar{\rho}_1)P \leq t_1 \leq \max(\phi_1 P, (1 - \bar{\rho}_1)P)$

$$\bar{w}_1^{VII}(t_1, \tau_1) = \frac{\mu_1 \bar{\rho}_1}{2\phi_1(\phi_1 - \bar{\rho}_1)P} \left[\phi_1 \tau_1^2 - 2P(\phi_1 - \bar{\rho}_1 + \phi_1 \bar{\rho}_1) \tau_1 \right. \\ \left. - 2P(1 - \phi_1)(\phi_1 - \bar{\rho}_1)t_1 - P^2(2\bar{\rho}_1 - 2\phi_1 + \phi_1^3 - 2\phi_1^2 \bar{\rho}_1) \right] \\ \text{for } (1 - \bar{\rho}_1)P \leq t_1 < P \text{ and } \bar{\rho}_1 P \leq \tau_1 \leq \left(\frac{\phi_1}{\bar{\rho}_1} - 1 \right) t_1 + \left(1 + \phi_1 - \frac{\phi_1}{\bar{\rho}_1} \right) P.$$

Mean work in process level for type 2 lots, $\hat{\lambda}_2 \geq \mu_2$

The difference with type 1 lots is that the inflow does not start at $t = 0$, but at $t = s_2$. The mean wip level for $\hat{\lambda}_2 \geq \mu_2$ follows the same curve as for type 1 lots, but shifted in time. This property is used for obtaining expressions for the mean wip level for type 2 lots. In Figure A.20 the mean wip level curve for type 1 lots has been plotted as a function of the start time of the process interval (solid line). For now, assume that this line represents the mean wip curve for type 2 lots, with $s_2 = 0$. All subscripts 1 can then be regarded as subscripts 2. The dashed line represents a shifted variant for type 2 lots, where $0 < s_2 < P$ indicates the start time of type 2 arrivals. t_2 denotes the start time of processing type 2 lots. As Figure A.20 shows, two situations can occur:

- Situation I: $0 \leq s_2 < \bar{\rho}_2 P$ (left hand side graph). The expression for the mean wip level and process interval are obtained by shifting the expressions for type 1 in time:

$$\bar{w}_2(t_2) = \begin{cases} \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2(1 - \bar{\rho}_2)(s_2 - t_2) & \text{for } 0 \leq t_2 < s_2 \\ \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2\bar{\rho}_2(t_2 - s_2) & \text{for } s_2 \leq t_2 < s_2 + (1 - \bar{\rho}_2)P \\ \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2(1 - \bar{\rho}_2)(P - t_2 + s_2) & \text{for } s_2 + (1 - \bar{\rho}_2)P \leq t_2 \leq P \end{cases}$$

$$\tau_2 = \bar{\rho}_2 P.$$

- Situation II: $\bar{\rho}_2 P \leq s_2 < P$ (right hand side graph). The mean wip level and process interval become:

$$\bar{w}_2(t_2) = \begin{cases} \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2\bar{\rho}_2(P + t_2 - s_2) & \text{for } 0 \leq t_2 < s_2 - \bar{\rho}_2 P \\ \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2(1 - \bar{\rho}_2)(s_2 - t_2) & \text{for } s_2 - \bar{\rho}_2 P \leq t_2 < s_2 \\ \frac{1}{2}\mu_2\bar{\rho}_2P(\bar{\rho}_2 - \phi_2) + \mu_2\bar{\rho}_2(t_2 - s_2) & \text{for } s_2 \leq t_2 \leq P \end{cases}$$

$$\tau_2 = \bar{\rho}_2 P.$$

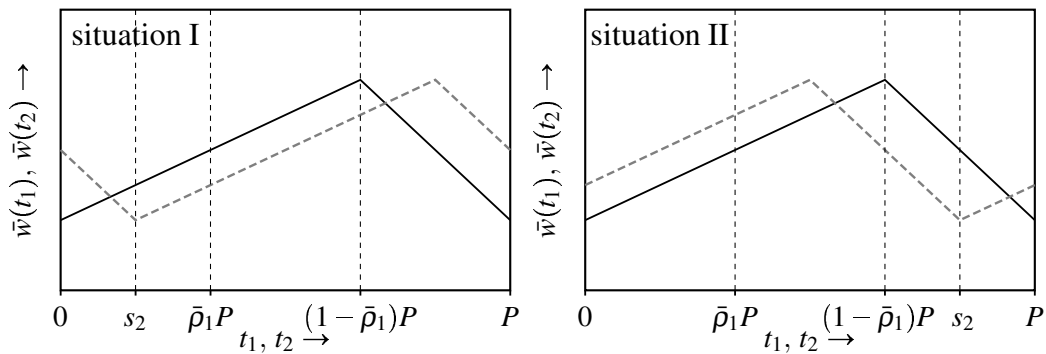


Figure A.20: Different situations for type 2 and $\hat{\lambda}_2 \geq \mu_2$. Determination of mean wip curve based on time shifting of the type 1 curve. Solid line: $s_1 = 0$. Dashed line: $s_2 > 0$.

Similar to the previous section, the mean wip level expressions for type 2 lots are the time-shifted equivalents of expressions the type 1 mean wip level expressions for $\hat{\lambda}_1 < \mu_1$. Due to the time shifting, case distinction takes place, yielding more expressions and conditions. The resulting mean wip level expressions for type 2 lots with $\hat{\lambda}_2 < \mu_2$ are as given below. The dotted lines separate the expressions with a different domain for s_2 (which is known beforehand and not a design variable).

$$\bar{w}_2^I(t_2, \tau_2) = \left\{ \begin{array}{l} \frac{\mu_2 \bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} [\phi_2 \tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2 \bar{\rho}_2) \tau_2 \\ \quad - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(t_2 - s_2) + \phi_2 P^2(2(\phi_2 - \bar{\rho}_2)(1 - \phi_2) + \phi_2^2)] \\ \quad \text{for } s_2 \leq t_2 \leq (\phi_2 - \bar{\rho}_2)P + s_2 \\ \quad \bar{\rho}_2 P \leq \tau_2 \leq \phi_2 P - t_2 + s_2 \\ \quad 0 \leq s_2 \leq (1 - [\phi_2 - \bar{\rho}_2])P \\ \\ \dots\dots\dots \\ \frac{\mu_2 \bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} [\phi_2 \tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2 \bar{\rho}_2) \tau_2 \\ \quad - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(t_2 - s_2 + P) + \phi_2 P^2(2(\phi_2 - \bar{\rho}_2)(1 - \phi_2) + \phi_2^2)] \\ \quad \text{for } 0 \leq t_2 \leq -(1 - [\phi_2 - \bar{\rho}_2])P + s_2 \\ \quad \bar{\rho}_2 P \leq \tau_2 \leq s_2 - t_2 - (1 - \phi_2)P \\ \quad (1 - [\phi_2 - \bar{\rho}_2])P \leq s_2 < P \\ \\ \frac{\mu_2 \bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} [\phi_2 \tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2 \bar{\rho}_2) \tau_2 \\ \quad - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(t_2 - s_2) + \phi_2 P^2(2(\phi_2 - \bar{\rho}_2)(1 - \phi_2) + \phi_2^2)] \\ \quad \text{for } s_2 \leq t_2 \leq P \\ \quad \bar{\rho}_2 P \leq \tau_2 \leq s_2 - t_2 + \phi_2 P \\ \quad (1 - [\phi_2 - \bar{\rho}_2])P \leq s_2 < P \end{array} \right.$$

$$\bar{w}_2^H(\tau_2) = \left\{ \begin{array}{l} \frac{\mu_2 \bar{\rho}_2 (P - \tau_2)^2}{2P(\phi_2 - \bar{\rho}_2)} \text{ for } \begin{array}{l} (\phi_1 - \bar{\rho}_2)P + s_2 \leq t_2 \leq \phi_2 P + s_2 \\ P - (t_2 - s_2) \leq \tau_2 \leq P \\ \tau_2 \geq \bar{\rho}_2 P \\ \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2} \right) P \\ 0 \leq s_2 \leq (1 - \phi_2)P \end{array} \\ \dots\dots\dots \\ \frac{\mu_2 \bar{\rho}_2 (P - \tau_2)^2}{2P(\phi_2 - \bar{\rho}_2)} \text{ for } \begin{array}{l} (\phi_1 - \bar{\rho}_2)P - P + s_2 \leq t_2 \leq -(1 - \phi_2)P + s_2 \\ s_2 - t_2 \leq \tau_2 \leq P \\ \tau_2 \geq \bar{\rho}_2 P \\ \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2} \right) P \\ (1 - \phi_2)P \leq s_2 \leq (1 - [\phi_2 - \bar{\rho}_2])P \end{array} \\ \dots\dots\dots \\ \frac{\mu_2 \bar{\rho}_2 (P - \tau_2)^2}{2P(\phi_2 - \bar{\rho}_2)} \text{ for } \begin{array}{l} (\phi_1 - \bar{\rho}_2)P + s_2 \leq t_2 \leq \phi_2 P + s_2 \\ P - (t_2 - s_2) \leq \tau_2 \leq P \\ \tau_2 \geq \bar{\rho}_2 P \\ \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2} \right) P \\ (1 - \phi_2)P \leq s_2 \leq (1 - [\phi_2 - \bar{\rho}_2])P \end{array} \\ \dots\dots\dots \\ \frac{\mu_2 \bar{\rho}_2 (P - \tau_2)^2}{2P(\phi_2 - \bar{\rho}_2)} \text{ for } \begin{array}{l} (\phi_1 - \bar{\rho}_2)P - P + s_2 \leq t_2 \leq -(1 - \phi_2)P + s_2 \\ s_2 - t_2 \leq \tau_2 \leq P \\ \tau_2 \geq \bar{\rho}_2 P \\ \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2} \right) P \\ (1 - [\phi_2 - \bar{\rho}_2])P \leq s_2 < P \end{array} \end{array} \right.$$

$$\begin{aligned}
\bar{w}_2^{III-VI}(t_2, \tau_2) = & \left\{ \begin{aligned} & \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (P + t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) \\ & \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\ & \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \geq P - (P + t_2 - s_2) \\ & \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \leq (1 + \phi_2)P - (P + t_2 - s_2) \\ & \quad 0 \leq s_2 \leq \bar{\rho}_2 P \\ \\ & \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(t_2 - s_2) \\ & \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\ & \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \geq P - (t_2 - s_2) \\ & \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \leq (1 + \phi_2)P - (t_2 - s_2) \\ & \quad 0 \leq s_2 \leq \bar{\rho}_2 P \\ \\ & \dots\dots\dots \\ & \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (P + t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) \\ & \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\ & \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \geq P - (P + t_2 - s_2) \\ & \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \leq (1 + \phi_2)P - (P + t_2 - s_2) \\ & \quad \bar{\rho}_2 P \leq s_2 \leq (1 - \phi_2)P \\ \\ & \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(t_2 - s_2) \\ & \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\ & \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \geq P - (t_2 - s_2) \\ & \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2}\right)P \\ & \quad \tau_2 \leq (1 + \phi_2)P - (t_2 - s_2) \\ & \quad \bar{\rho}_2 P \leq s_2 \leq (1 - \phi_2)P \\ \\ & \dots\dots\dots \\ & \text{(continued on next page)} \end{aligned} \right.
\end{aligned}$$

$$\begin{aligned}
& \text{(continued from previous page)} \\
& \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (P + t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) \\
& \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\
& \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \geq P - (P + t_2 - s_2) \\
& \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \leq (1 + \phi_2)P - (P + t_2 - s_2) \\
& \quad (1 - \phi_2)P \leq s_2 \leq (1 - [\phi_2 - \bar{\rho}_2])P \\
& \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(t_2 - s_2) \\
& \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\
& \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \geq P - (t_2 - s_2) \\
& \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \leq (1 + \phi_2)P - (t_2 - s_2) \\
& \quad (1 - \phi_2)P \leq s_2 \leq (1 - [\phi_2 - \bar{\rho}_2])P \\
& \dots\dots\dots \\
& \frac{\mu_2 \bar{\rho}_2}{2\phi_2^2 P} [((1 + \phi_2)P - \tau_2 - (P + t_2 - s_2)) ((\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) \\
& \quad - (\phi_2 + \bar{\rho}_2)\tau_2 + P(\bar{\rho}_2 + \phi_2 + \bar{\rho}_2\phi_2 - \phi_2^2))] \\
& \text{for } \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \geq P - (P + t_2 - s_2) \\
& \quad \tau_2 \geq \left(\frac{\phi_2}{\bar{\rho}_2} - 1\right)(P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2^2}{\bar{\rho}_2}\right)P \\
& \quad \tau_2 \leq (1 + \phi_2)P - (P + t_2 - s_2) \\
& \quad (1 - [\phi_2 - \bar{\rho}_2])P \leq s_2 < P
\end{aligned}$$

$$\begin{aligned}
\bar{w}_2^{IV-V}(t_2) &= \left\{ \begin{array}{l} -\frac{1}{2}\mu_2\bar{\rho}_2P(\phi_2 - \bar{\rho}_2) + \mu_2\bar{\rho}_2(t_2 - s_2) \text{ and } \tau_2 = \bar{\rho}_2P \\ \text{for } (\phi_2 - \bar{\rho}_2)P + s_2 \leq t_2 \leq \max(\phi_2P, (1 - \bar{\rho}_2)P) + s_2 \\ 0 \leq s_2 \leq P - \max(\phi_2P, (1 - \bar{\rho}_2)P) \\ \dots\dots\dots \\ -\frac{1}{2}\mu_2\bar{\rho}_2P(\phi_2 - \bar{\rho}_2) + \mu_2\bar{\rho}_2(P + t_2 - s_2) \text{ and } \tau_2 = \bar{\rho}_2P \\ \text{for } 0 \leq t_2 \leq \max(\phi_2P, (1 - \bar{\rho}_2)P) - P + s_2 \\ P - \max(\phi_2P, (1 - \bar{\rho}_2)P) \leq s_2 \leq (1 - (\phi_2 - \bar{\rho}_2))P \\ \dots\dots\dots \\ -\frac{1}{2}\mu_2\bar{\rho}_2P(\phi_2 - \bar{\rho}_2) + \mu_2\bar{\rho}_2(t_2 - s_2) \text{ and } \tau_2 = \bar{\rho}_2P \\ \text{for } (\phi_2 - \bar{\rho}_2)P + s_2 \leq t_2 \leq P \\ P - \max(\phi_2P, (1 - \bar{\rho}_2)P) \leq s_2 \leq (1 - (\phi_2 - \bar{\rho}_2))P \\ \dots\dots\dots \\ -\frac{1}{2}\mu_2\bar{\rho}_2P(\phi_2 - \bar{\rho}_2) + \mu_2\bar{\rho}_2(P + t_2 - s_2) \text{ and } \tau_2 = \bar{\rho}_2P \\ \text{for } -(1 - (\phi_2 - \bar{\rho}_2))P + s_2 \leq t_2 \leq \max(\phi_2P, (1 - \bar{\rho}_2)P) - P + s_2 \\ (1 - (\phi_2 - \bar{\rho}_2))P \leq s_2 < P \end{array} \right. \\
\\
\bar{w}_2^{VII}(t_2, \tau_2) &= \left\{ \begin{array}{l} \frac{\mu_2\bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} \left[\phi_2\tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2\bar{\rho}_2)\tau_2 \right. \\ \quad \left. - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) - P^2(2\bar{\rho}_2 - 2\phi_2 + \phi_2^3 - 2\phi_2^2\bar{\rho}_2) \right] \\ \text{for } -(1 - (1 - \bar{\rho}_2))P + s_2 \leq t_2 < s_2 \\ \bar{\rho}_2P \leq \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2} \right) P \\ 0 \leq s_2 \leq \bar{\rho}_2P \\ \dots\dots\dots \\ \frac{\mu_2\bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} \left[\phi_2\tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2\bar{\rho}_2)\tau_2 \right. \\ \quad \left. - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(t_2 - s_2) - P^2(2\bar{\rho}_2 - 2\phi_2 + \phi_2^3 - 2\phi_2^2\bar{\rho}_2) \right] \\ \text{for } (1 - \bar{\rho}_2)P + s_2 \leq t_2 < P + s_2 \\ \bar{\rho}_2P \leq \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2} \right) P \\ 0 \leq s_2 \leq \bar{\rho}_2P \\ \dots\dots\dots \\ \frac{\mu_2\bar{\rho}_2}{2\phi_2(\phi_2 - \bar{\rho}_2)P} \left[\phi_2\tau_2^2 - 2P(\phi_2 - \bar{\rho}_2 + \phi_2\bar{\rho}_2)\tau_2 \right. \\ \quad \left. - 2P(1 - \phi_2)(\phi_2 - \bar{\rho}_2)(P + t_2 - s_2) - P^2(2\bar{\rho}_2 - 2\phi_2 + \phi_2^3 - 2\phi_2^2\bar{\rho}_2) \right] \\ \text{for } -(1 - (1 - \bar{\rho}_2))P + s_2 \leq t_2 < s_2 \\ \bar{\rho}_2P \leq \tau_2 \leq \left(\frac{\phi_2}{\bar{\rho}_2} - 1 \right) (P + t_2 - s_2) + \left(1 + \phi_2 - \frac{\phi_2}{\bar{\rho}_2} \right) P \\ \bar{\rho}_2P \leq s_2 \leq P. \end{array} \right.
\end{aligned}$$

A.6 Proof of Remark 5.20 (page 131)

The claim is that for a single switching server with piecewise constant arrival rates, the optimization problem as formulated in Section 5.8 leads to the condition for occurrence of a slow-mode, as stated in Theorem 5.10, when the arrival time span fractions ϕ_i are set to one ($i \in \{1, 2\}$).

Physically, both arrival rates become constant when $\phi_i = 1 \forall i$, since no room is left between two successive arrival time spans. For a given period P , it means that all t_i lie between 0 and $\phi_i P$. The maximum process rate μ_i must be larger than (or equal to) the arrival rate $\hat{\lambda}_i$, to ensure stability (as explained in Section 5.1). For both types, it means that the expressions for the mean wip level for $\hat{\lambda}_i \leq \mu_i$, $i \in \{1, 2\}$ have to be used. Reviewing the possible different situations, as discussed in Appendix A.5, the expressions for the mean wip level for type 1 and type 2 lots are obtained by taking the mean wip expressions of situation I and II, substituting $\phi_i = 1$:

$$\begin{aligned}\bar{w}_1(t_1, \tau_1) &= \frac{\bar{\rho}_1 \mu_1 (P - \tau_1)^2}{2P(1 - \bar{\rho}_1)} \text{ with } \bar{\rho}_1 P \leq \tau_1 \leq P \\ \bar{w}_2(t_2, \tau_2) &= \frac{\bar{\rho}_2 \mu_2 (P - \tau_2)^2}{2P(1 - \bar{\rho}_2)} \text{ with } \bar{\rho}_2 P \leq \tau_2 \leq P\end{aligned}$$

with the constraints:

$$\begin{aligned}t_2 + \tau_2 + \sigma_{21} &\leq t_1 \\ t_1 + \tau_1 + \sigma_{12} &\leq t_2 + P\end{aligned} \tag{A.40}$$

or

$$\begin{aligned}t_1 + \tau_1 + \sigma_{12} &\leq t_2 \\ t_2 + \tau_2 + \sigma_{21} &\leq t_1 + P.\end{aligned} \tag{A.41}$$

When $\phi_i = 1 \forall i$, no fixed period exists, given by the arrival pattern. Therefore, it is also impossible to distinguish *start times of processing lots within the period length* t_1 and t_2 . The expressions for mean wip level confirm this: they are independent of t_1 and t_2 . On the other hand, the period length P has now become a design variable in the optimization problem! The problem that is to be solved reads: find the process cycle period length and individual type specific process interval lengths for the switching server which ensure minimal weighted work in process levels.

For reasons of simplicity, let $\rho_i = \bar{\rho}_i|_{\phi_i=1} = \hat{\lambda}_i/\mu_i$. With $\phi_i = 1$, the mean wip level expressions and constraints can be written as:

$$\begin{aligned}\bar{w}_1(\tau_1, P) &= \frac{\mu_1 \rho_1 (P - \tau_1)^2}{2P(1 - \rho_1)} \\ \bar{w}_2(\tau_2, P) &= \frac{\mu_2 \rho_2 (P - \tau_2)^2}{2P(1 - \rho_2)}\end{aligned}$$

with the constraints:

$$\begin{aligned}\rho_1 P &\leq \tau_1 \leq P \\ \rho_2 P &\leq \tau_2 \leq P \\ \tau_1 + \tau_2 + \sigma_{12} + \sigma_{21} &\leq P.\end{aligned}$$

The first and second constraint denote the minimal and maximum duration of the process interval lengths and the third constraint is obtained by combining either of the two constraint sets (A.40) or (A.41), losing t_1 and t_2 . Lemma 5.17 states that lots are always served at the highest currently possible rate, after which the server might idle. As long as the server is not setting up for a lot type, serving at the highest possible rate means that as long the buffers contain lots or as long lots keep arriving, the highest possible rate > 0 . A consequence for $\phi_i = 1$ is that lots keep arriving all the time, so the server never idles. The period length P obviously equals the sum of the process interval lengths and setup times then. So the final constraint must hold with equality. Rewriting the constraints thus yields:

$$\begin{aligned}\rho_1 P &\leq \tau_1 \leq P - \sigma_{12} - \sigma_{21} - \tau_2 \\ \rho_2 P &\leq \tau_2 \leq P - \sigma_{12} - \sigma_{21} - \tau_1\end{aligned}$$

which in turn can be simplified into (with $\sigma = \sigma_{12} + \sigma_{21}$):

$$\begin{aligned}\rho_1 P &\leq \tau_1 \leq (1 - \rho_2)P - \sigma \\ \rho_2 P &\leq \tau_2 \leq (1 - \rho_1)P - \sigma.\end{aligned}$$

The equality $P = \tau_1 + \tau_2 + \sigma$ allows for elimination of 1 design variable, so with $\mu_i \rho_i = \lambda_i$ the optimization problem becomes:

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

subject to:

$$\begin{aligned}\rho_1 P &\leq \tau_1 \leq (1 - \rho_2)P - \sigma \\ P &\geq \frac{\sigma}{1 - \rho_1 - \rho_2}.\end{aligned}$$

Without loss of generality, it is assumed that $c_1 \lambda_1 \geq c_2 \lambda_2$, as it had been assumed in Section 5.2. The optimization problem is solved below. Based on the parameter choices, constraints can be active (hold with equality) or inactive (hold with inequality). The optimization problem is solved for the unconstrained case, with one active constraint and with two active constraints. This yields conditions for the occurrence of possible slow-modes.

- **Unconstrained optimization**

The optimal values of P and τ_1 are obtained by solving the following set of equations:

$$\left\{ \frac{\partial J}{\partial P} = 0, \frac{\partial J}{\partial \tau_1} = 0 \right\}$$

which has as solutions:

$$\{P^* = -\sigma, \tau_1^* = -\sigma\} \text{ or } \left\{P^* = \sigma, \tau_1^* = \frac{\sigma(c_2\lambda_2(1-\rho_1) - c_1\lambda_1(1-\rho_2))}{-c_2\lambda_2(1-\rho_1) - c_1\lambda_1(1-\rho_2)}\right\}.$$

Both solutions clearly violate the constraints, so at least one constraint of the original problem is active.

- **One active constraint**

First assume that $\tau_1^* = \rho_1 P$. Substituting this in the objective function and solving $\frac{\partial J}{\partial P} = 0$ gives as solutions:

$$P^* = \frac{\sigma\sqrt{c_2\lambda_2}}{\sqrt{c_1\lambda_1(1-\rho_1)(1-\rho_2) + c_2\lambda_2\rho_1^2}} \text{ or } P^* = -\frac{\sigma\sqrt{c_2\lambda_2}}{\sqrt{c_1\lambda_1(1-\rho_1)(1-\rho_2) + c_2\lambda_2\rho_1^2}}.$$

The first solution can be a valid solution, while the second solution is negative and therefore rejected.

The other possibility for one active constraint is: $\tau_1^* = (1-\rho_2)P - \sigma$. Substituting this in the objective function and solving for P gives as solutions:

$$P^* = \frac{\sigma\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2}} \text{ or } P^* = -\frac{\sigma\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2}}.$$

Again, only the first solution can be a valid solution. Notice the symmetry with the solutions of the previous optimization, which is not surprising, when realizing that in fact the objective function is symmetric for type 1 and type 2 lots. In the first term of the objective function, $P - \tau_1$ can be replaced with $\sigma + \tau_2$ to recognize the symmetry.

- **Two active constraints**

The constraints hold with equality: $\tau_1 = \rho_1 P$ and $\tau_1 = (1-\rho_2)P - \sigma$. Automatically, the third constraint of the original problem ($P \geq \frac{\sigma}{1-\rho_1-\rho_2}$) becomes active. The set of two equalities has as a solution:

$$P^* = \frac{\sigma}{1-\rho_1-\rho_2} \text{ and } \tau_1^* = \frac{\rho_1\sigma}{1-\rho_1-\rho_2}$$

which is the solution for the pure bow tie curve (see Section 5.2). In this case, no slow-modes occur.

Now that the optimization problem has been solved (or at least the solution has been characterized), the question arises: when are the constraints active?

First, the activity of the constraint $\tau_1 = (1-\rho_2)P - \sigma$ is investigated (with the other constraint inactive). The optimal period length P is given by:

$$P^* = \frac{\sigma\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1-\rho_1)(1-\rho_2) + c_1\lambda_1\rho_2^2}}$$

so the process interval τ_1 becomes:

$$\tau_1^* = (1 - \rho_2)P - \sigma = \frac{(1 - \rho_2)\sigma\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1 - \rho_1)(1 - \rho_2) + c_1\lambda_1\rho_2^2}} - \sigma.$$

The other constraint ($\rho_1 P \leq \tau_1$) is inactive, so it holds with strict inequality: $\rho_1 P^* < \tau_1^*$:

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

Division by $\sigma > 0$ and some rewriting gives:

$$\frac{\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1 - \rho_1)(1 - \rho_2) + c_1\lambda_1\rho_2^2}} < \frac{1}{1 - \rho_1 - \rho_2}.$$

All numerators and denominators are positive, so it is allowed to write:

$$\sqrt{c_1\lambda_1}(1 - \rho_1 - \rho_2) < \sqrt{c_2\lambda_2(1 - \rho_1)(1 - \rho_2) + c_1\lambda_1\rho_2^2}$$

and after squaring and some basic algebra:

$$c_1\lambda_1(\rho_1 + \rho_2) - (c_1\lambda_1 - c_2\lambda_2)(1 - \rho_2) < 0.$$

Because of the symmetry of the optimization problem, it is possible to immediately write down the condition for the activeness of constraint $\tau_1 = \rho_1 P$ and strict inequality $\tau_1 < (1 - \rho_2)P - \sigma$:

$$c_2\lambda_2(\rho_1 + \rho_2) + (c_1\lambda_1 - c_2\lambda_2)(1 - \rho_1) < 0.$$

Since it was assumed that $c_1\lambda_1 \geq c_2\lambda_2$, the latter condition is never fulfilled, leaving the following conclusion:

- Unconstrained optimum: infeasible.
- One active constraint: $\tau_1 = (1 - \rho_2)P - \sigma$ is the only candidate. This situation occurs when $c_1\lambda_1(\rho_1 + \rho_2) - (c_1\lambda_1 - c_2\lambda_2)(1 - \rho_2) < 0$.
- Two active constraints: pure bow tie curve: no occurrence of slowmodes.

The only possible steady state process cycle which is not the bow tie curve, is the one with one or more slowmodes. If constraint $\tau_1 = (1 - \rho_2)P - \sigma$ is active, then the lengths of the period and the process intervals are:

$$\begin{aligned} P^* &= \frac{\sigma\sqrt{c_1\lambda_1}}{\sqrt{c_2\lambda_2(1 - \rho_1)(1 - \rho_2) + c_1\lambda_1\rho_2^2}} \\ \tau_1^* &= (1 - \rho_2)P - \sigma \\ \tau_2^* &= P^* - \sigma - \tau_1^*. \end{aligned}$$

For this solution, mass conservation tells which lot type(s) is/are processed in a slow-mode. Since the machine does not idle and lots are always processed at the highest possible rate, the only process rates that occur are maximum rate μ_i and arrival rate λ_i . The duration of processing at μ_i is denoted by τ_i^μ while the slow-modes take τ_i^λ . The following set of equations needs to be solved:

$$\begin{aligned}\tau_1^* &= \tau_1^\mu + \tau_1^\lambda \\ \tau_2^* &= \tau_2^\mu + \tau_2^\lambda \\ \lambda_1 P^* &= \mu_1 \tau_1^\mu + \lambda_1 \tau_1^\lambda \\ \lambda_2 P^* &= \mu_2 \tau_2^\mu + \lambda_2 \tau_2^\lambda\end{aligned}$$

which has as solution:

$$\begin{aligned}\tau_1^\mu &= \frac{\rho_1}{1 - \rho_1} (P^* - \tau_1^*) > 0 \\ \tau_1^\lambda &= \frac{\tau_1^* - \rho_1 P^*}{1 - \rho_1} > 0 \\ \tau_2^\mu &= \frac{\rho_2}{1 - \rho_2} (P^* - \tau_2^*) > 0 \\ \tau_2^\lambda &= \frac{\tau_2^* - \rho_2 P^*}{1 - \rho_2} = 0.\end{aligned}$$

From this solution one can conclude that the only possible slowmode that may occur is for type 1 lots (under the assumption that $c_1 \lambda_1 \geq c_2 \lambda_2$). A slowmode occurs in the optimal steady state cycle if $c_1 \lambda_1 (\rho_1 + \rho_2) - (c_1 \lambda_1 - c_2 \lambda_2) (1 - \rho_2) < 0$.

A.7 Proof of Proposition 6.3 (page 145)

Initially, the workstations have to get into the same mode $\mathbf{m} = (m^A, m^B)$. Without loss of generality, the mode of B is adjusted to equal the mode of A (if necessary). This is done by means of the following initial control action:

$$\begin{pmatrix} u_0^A \\ u_0^B \\ u_1^A \\ u_2^A \\ u_1^B \\ u_2^B \end{pmatrix}^T = \begin{cases} (\mathbf{1}, \mathbf{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A > 0 \\ (\mathbf{1}, \mathbf{1}, \mu_1^A, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_1^A > 0 \\ (\mathbf{1}, \mathbf{1}, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_1^A = 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A > 0 \\ (\mathbf{2}, \mathbf{2}, 0, \mu_2^A, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = 0, x_2^A > 0 \\ (\mathbf{2}, \mathbf{2}, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = 0, x_2^A = 0 \end{cases} \quad (\text{A.42})$$

Immediately after this action (if necessary), the following feedback is used:

$$\begin{pmatrix} u_0^A \\ u_0^B \\ u_1^A \\ u_2^A \\ u_1^B \\ u_2^B \end{pmatrix}^\top = \begin{cases} (\mathbf{1}, \mathbf{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B > 0 \\ (\mathbf{1}, \mathbf{1}, 0, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B = 0, x_1^B > 0 \\ (\mathbf{1}, \mathbf{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B = 0, x_1^B = 0 \\ (\mathbf{1}, \mathbf{1}, \mu_1^A, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B > 0, x_1^A > 0 \\ (\mathbf{1}, \mathbf{1}, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B > 0, x_1^A = 0 \\ (\mathbf{1}, \mathbf{1}, \mu_1^A, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A > 0, x_1^B > 0 \\ (\mathbf{1}, \mathbf{1}, \lambda_1, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_1^B > 0 \\ (\mathbf{1}, \mathbf{1}, \mu_1^A, 0, \mu_1^A, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A > 0, x_1^B = 0 \\ (\mathbf{1}, \mathbf{1}, \lambda_1, 0, \lambda_1, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_1^B = 0, x_2^A < x_2^{A\sharp} \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_1^B = 0, x_2^A \geq x_2^{A\sharp} \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B = 0, x_2^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B = 0, x_2^B = 0 \\ (\mathbf{2}, \mathbf{2}, 0, \mu_2^A, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B > 0, x_2^A > 0 \\ (\mathbf{2}, \mathbf{2}, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B > 0, x_2^A = 0 \\ (\mathbf{2}, \mathbf{2}, 0, \mu_2^A, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A > 0, x_2^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, \lambda_2, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A = 0, x_2^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, \mu_2^A, 0, \mu_2^A) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A > 0, x_2^B = 0 \\ (\mathbf{2}, \mathbf{2}, 0, \lambda_2, 0, \lambda_2) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A = 0, x_2^B = 0, x_1^A < x_1^{A\sharp} \\ (\mathbf{1}, \mathbf{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A = 0, x_2^B = 0, x_1^A \geq x_1^{A\sharp} \end{cases} \quad (\text{A.43})$$

in which $x_1^{A\sharp}$ and $x_2^{A\sharp}$ are given by the expressions in (5.10).

Recall that $\mu_i^B \geq \mu_i^A$ for $i \in \{1, 2\}$ and that setup times $\sigma_{12}^B \leq \sigma_{12}^A$ and $\sigma_{21}^B \leq \sigma_{21}^A$. This means that if workstation B always serves at the highest possible actual process rate, it is always able to keep a buffer empty once it is empty. The controller loops the lines as explained above in the informal description of the controller. After one complete loop the buffers in workstation B are empty, since switching only takes place when both buffers of that specific job type are empty. The maximum process rates of workstation B are greater than the maximum process rates of A and these are in turn greater than the arrival rates of lots, so eventually the buffers of a lot type are empty during the first cycle. After having completed one process cycle, workstation B does not influence the evolution of the trajectory anymore: it is never restrictive in the controller. Therefore, after one cycle, the controlled system behaves as if it were only the first workstation. Convergence of the controller is then exactly similar to the proof of Proposition 5.13.

A.8 Proof of Theorem 6.5 (page 154)

The following is the proof of Theorem 6.5. Goal is to show that the following equations (which are (6.9a) and (6.9b)) are necessary and sufficient conditions for workstation A to make the flow line consisting of two workstations behave as if it were the downstream workstation B stand-alone.

$$\begin{aligned} R_2 \left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A &\geq 0 \\ R_1 \left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A &\geq 0 \end{aligned}$$

with $R_1 = \max(\rho_1^A, \rho_1^B)$ and $R_2 = \max(\rho_2^A, \rho_2^B)$.

The proof consists of two parts. First, if a periodic orbit of A makes the flow line of workstations A and B behave like B stand-alone, it fulfills (6.3)–(6.8). During $\tau_i^{\mu A} + \theta_i^-$, A has to process $\mu_i^B \tau_i^{\mu B} - \lambda_i \theta_i^+ = \lambda_i(T - \theta_i^+ - \tau_i^{\lambda B})$ lots. This takes at least $\lambda_i(T - \theta_i^+ - \tau_i^{\lambda B})/\mu_i^A$ time units (if $\theta_i^- = 0$). As a consequence:

$$\tau_i^{\mu A} + \theta_i^- \geq \rho_i^A(T - \theta_i^+ - \tau_i^{\lambda B}). \quad (\text{A.44})$$

Note that the mass conservation requirement has been replaced with inequality constraints. Combining this result with (6.5) results in:

$$\sigma_{21}^B - \sigma_{21}^A + \tau_1^{\mu B} - \theta_2^+ \geq \rho_1^A(T - \theta_1^+ - \tau_1^{\lambda B}) \quad (\text{A.45})$$

which can be rewritten as:

$$\rho_1^A(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \quad (\text{A.46a})$$

and similar for the other lot type:

$$\rho_2^A(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B. \quad (\text{A.46b})$$

From (6.8) it is known that:

$$\tau_i^{\mu B} + \rho_i^B(\tau_i^{\lambda B} - T) = 0. \quad (\text{A.47})$$

Adding this to (6.7) results in:

$$\rho_1^B(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \quad (\text{A.48a})$$

$$\rho_2^B(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B \quad (\text{A.48b})$$

which looks rather similar to (A.46a) and (A.46b). Combining (A.46) and (A.48) results in:

$$\max(\rho_1^A, \rho_1^B)(\theta_1^+ + \tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \theta_2^+ \geq \sigma_{21}^A - \sigma_{21}^B \quad (\text{A.49a})$$

$$\max(\rho_2^A, \rho_2^B)(\theta_2^+ + \tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \theta_1^+ \geq \sigma_{12}^A - \sigma_{12}^B. \quad (\text{A.49b})$$

These inequalities, together with $\theta_1^+ \geq 0$ and $\theta_2^+ \geq 0$, enclose a feasible area in the (θ_2^+, θ_1^+) -plane (Figure A.21).

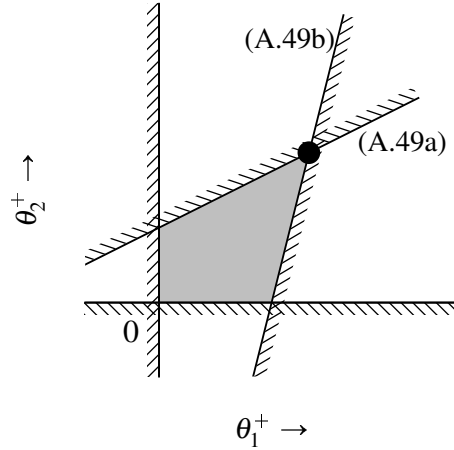


Figure A.21: (θ_2^+, θ_1^+) -plane with feasible area (gray). The diagonal lines represent constraints (A.49a) and (A.49b).

The intersection point of the two linear borders defined by (A.49) lies in the positive/positive quarter of this plane. The intersection point is given by:

$$\theta_1^+ = \frac{R_2 \left[\tau_1^{\mu B} + \tau_2^{\lambda B} + \sigma_{21}^B - \sigma_{21}^A - T + R_1(\tau_1^{\lambda B} - T) \right] + \tau_2^{\mu B} + \sigma_{12}^B - \sigma_{12}^A}{1 - R_1 R_2} \quad (\text{A.50a})$$

$$\theta_2^+ = \frac{R_1 \left[\tau_2^{\mu B} + \tau_1^{\lambda B} + \sigma_{12}^B - \sigma_{12}^A - T + R_2(\tau_2^{\lambda B} - T) \right] + \tau_1^{\mu B} + \sigma_{21}^B - \sigma_{21}^A}{1 - R_1 R_2} \quad (\text{A.50b})$$

which are only both non-negative if (6.9a) and (6.9b) are fulfilled.

The second part of the proof is to show that conditions (6.9) are sufficient to guarantee that A can make the flow line of workstations A and B behave like B stand-alone. For arbitrary θ_i^+ (the amount of time a slowmode in workstation A of type i takes after the slowmode of B has ended), the values for the processing intervals $\tau_i^{\mu A}$ and $\tau_i^{\lambda A}$ can be computed. These can be solved from (6.3), (6.5a), (6.5b) and (6.8), guaranteeing mass conservation and the requirement that the process cycle of A fits in the process cycle of B for both lot types:

$$\tau_1^{\mu A} = \frac{\rho_1^A(\sigma_{12}^B + \tau_2^{\mu B} + \tau_2^{\lambda B} - \theta_1^+ + \theta_2^+ + \sigma_{21}^A)}{1 - \rho_1^A} \quad (\text{A.51})$$

$$\tau_2^{\mu A} = \frac{\rho_2^A(\tau_1^{\mu B} + \tau_1^{\lambda B} + \sigma_{21}^B - \theta_2^+ + \theta_1^+ + \sigma_{12}^A)}{1 - \rho_2^A} \quad (\text{A.52})$$

$$\theta_1^- = \frac{\sigma_{21}^B - \theta_2^+ - \sigma_{21}^A + \tau_1^{\mu B} + \rho_1^A(\theta_1^+ - \tau_1^{\mu B} - \tau_2^{\mu B} - \tau_2^{\lambda B} - \sigma_{21}^B - \sigma_{12}^B)}{1 - \rho_1^A} \quad (\text{A.53})$$

$$\theta_2^- = \frac{\sigma_{12}^B - \theta_1^+ - \sigma_{12}^A + \tau_2^{\mu B} + \rho_2^A(\theta_2^+ - \tau_1^{\mu B} - \tau_2^{\mu B} - \tau_1^{\lambda B} - \sigma_{21}^B - \sigma_{12}^B)}{1 - \rho_2^A} \quad (\text{A.54})$$

$$\tau_1^{\lambda A} = \theta_1^- + \tau_1^{\lambda B} + \theta_1^+ \quad (\text{A.55})$$

$$\tau_2^{\lambda A} = \theta_2^- + \tau_2^{\lambda B} + \theta_2^+. \quad (\text{A.56})$$

If θ_1^+ and θ_2^+ are chosen in such a way that $\tau_1^{\mu A}$, θ_1^- , $\tau_2^{\mu A}$ and θ_2^- are non-negative and moreover, the buffer levels do not become negative ((6.7a) and (6.7b)), then a feasible solution has been found. From (A.55) and (A.56), it can easily be seen that $\tau_1^{\lambda A}$ and $\tau_2^{\lambda A}$ are also non-negative then. If (6.9a) and (6.9b) hold, then (A.50) gives feasible values for θ_1^+ and θ_2^+ , since they are non-negative. In (A.50) the period length T and the process intervals $\tau_i^{\mu B}$ are substituted with:

$$\tau_i^{\mu B} = \rho_i^B (T - \tau_i^{\lambda B}) \quad (\text{follows from (6.8)}) \quad (\text{A.57})$$

$$T = \tau_1^{\mu B} + \tau_2^{\mu B} + \tau_1^{\lambda B} + \tau_2^{\lambda B} + \sigma_{12}^B + \sigma_{21}^B. \quad (\text{A.58})$$

Substituting (A.50) in (A.51)–(A.54) gives expressions for $\tau_1^{\mu A}$, $\tau_2^{\mu A}$, θ_1^- and θ_2^- :

$$\begin{aligned} \tau_1^{\mu A} = & \frac{\rho_1^A (1 - R_1)}{(1 - \rho_1^A)(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[[1 - \rho_1^B - \rho_2^B] \sigma_{12}^A \right. \\ & + R_2 [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A + [R_2 (1 - \rho_1^B) + \rho_1^B] \sigma_{12}^B + [1 - \rho_2^B + R_2 \rho_2^B] \sigma_{21}^B \\ & \left. + [R_2 (1 - \rho_1^B) + \rho_1^B \rho_2^B (1 - R_2)] \tau_1^{\lambda B} + [\rho_1^B \rho_2^B (R_2 - 1) + 1 - \rho_2^B] \tau_2^{\lambda B} \right] \end{aligned}$$

$$\begin{aligned} \tau_2^{\mu A} = & \frac{\rho_2^A (1 - R_2)}{(1 - \rho_2^A)(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[R_1 [1 - \rho_1^B - \rho_2^B] \sigma_{12}^A \right. \\ & + [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A + [1 - \rho_1^B + R_1 \rho_1^B] \sigma_{12}^B + [R_1 (1 - \rho_2^B) + \rho_2^B] \sigma_{21}^B \\ & \left. + [\rho_1^B \rho_2^B (R_1 - 1) + 1 - \rho_1^B] \tau_1^{\lambda B} + [R_1 (1 - \rho_2^B) + \rho_1^B \rho_2^B (1 - R_1)] \tau_2^{\lambda B} \right] \end{aligned}$$

$$\begin{aligned} \theta_1^- = & \frac{R_1 - \rho_1^A}{(1 - \rho_1^A)(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[[1 - \rho_1^B - \rho_2^B] \sigma_{12}^A \right. \\ & + R_2 [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A + [\rho_1^B + R_2 (1 - \rho_1^B)] \sigma_{12}^B + [1 - \rho_2^B + R_2 \rho_2^B] \sigma_{21}^B \\ & \left. + [R_2 (1 - \rho_1^B) + \rho_1^B \rho_2^B (1 - R_2)] \tau_1^{\lambda B} + [\rho_1^B \rho_2^B (R_2 - 1) + 1 - \rho_2^B] \tau_2^{\lambda B} \right] \end{aligned}$$

$$\begin{aligned} \theta_2^- = & \frac{R_2 - \rho_2^A}{(1 - \rho_2^A)(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[R_1 [1 - \rho_1^B - \rho_2^B] \sigma_{12}^A \right. \\ & + [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A + [1 - \rho_1^B + R_1 \rho_1^B] \sigma_{12}^B + [\rho_2^B + R_1 (1 - \rho_2^B)] \sigma_{21}^B \\ & \left. + [\rho_1^B \rho_2^B (R_1 - 1) + 1 - \rho_1^B] \tau_1^{\lambda B} + [R_1 (1 - \rho_2^B) + \rho_1^B \rho_2^B (1 - R_1)] \tau_2^{\lambda B} \right]. \end{aligned}$$

Each expression above consists of six terms. Recalling that workload $0 \leq \rho_i^j < 1$, $\sum_i \rho_i^j < 1$ and that $R_i = \max(\rho_i^A, \rho_i^B)$, one can easily verify that all six terms in each expression are non-negative. Finally, (6.7a) and (6.7b) are checked, resulting in:

$$\begin{aligned} & \frac{R_1 - \rho_1^B}{(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[-[1 - \rho_1^B - \rho_2^B] \sigma_{12}^A - R_2 [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A \right. \\ & \quad - [\rho_1^B + R_2 (1 - \rho_1^B)] \sigma_{12}^B - [1 - \rho_2^B (1 - R_2)] \sigma_{21}^B \\ & \quad \left. - [\rho_1^B \rho_2^B (1 - R_2) + R_2 (1 - \rho_1^B)] \tau_1^{\lambda B} - [\rho_1^B \rho_2^B (R_2 - 1) + 1 - \rho_2^B] \tau_2^{\lambda B} \right] \leq 0 \end{aligned}$$

and

$$\begin{aligned} & \frac{R_2 - \rho_2^B}{(1 - R_1 R_2)(1 - \rho_1^B - \rho_2^B)} \left[-R_1 [1 - \rho_1^B - \rho_2^B] \sigma_{12}^A - [1 - \rho_1^B - \rho_2^B] \sigma_{21}^A \right. \\ & \quad - [1 - \rho_1^B (1 - R_1)] \sigma_{12}^B - [\rho_2^B + R_1 (1 - \rho_2^B)] \sigma_{21}^B \\ & \quad \left. - [(\rho_1^B \rho_2^B (R_1 - 1) + 1 - \rho_1^B) \tau_1^{\lambda B} - [\rho_1^B \rho_2^B (1 - R_1) + R_1 (1 - \rho_2^B)] \tau_2^{\lambda B}] \right] \leq 0. \end{aligned}$$

For these two inequalities, it can easily be verified that all terms at the left hand side are non-positive, so the inequalities hold. With these results, it has been proven that (6.9a) and (6.9b) are sufficient conditions to guarantee that workstation A can make the flow line of workstations A and B perform like B stand-alone with respect to work in process levels.

A.9 Proof of Proposition 6.6 (page 155)

The following Lemma is needed in the proof of Proposition 6.6:

Lemma A.5. *Under conditions (6.9a) and (6.9b), i.e. workstation A can make flow line of workstations A and B behave like B stand-alone, the following inequality holds: $R_1 + R_2 < 1$.*

Proof. Choose θ_i^+ as small as possible, but within the feasible area (cf. Figure A.21). The feasible area may have three different shapes, as presented in Figure A.22. The smallest values

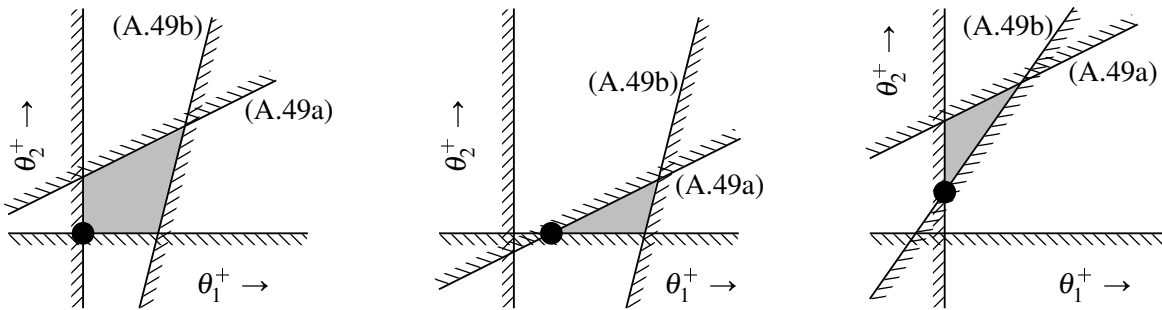


Figure A.22: Possible shapes of the feasible area in the (θ_2^+, θ_1^+) -plane.

for θ_1^+ is the intersection point of (A.49a) with the horizontal axis, or zero if the intersection point is negative. A similar reasoning goes for θ_2^+ , the vertical axis and constraint line (A.49b). This leads to the following expressions for θ_i^+ , $i \in \{1, 2\}$:

$$\theta_1^+ = \max(0, R_2(\tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \sigma_{12}^A + \sigma_{12}^B) \quad (\text{A.59a})$$

$$\theta_2^+ = \max(0, R_1(\tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \sigma_{21}^A + \sigma_{21}^B). \quad (\text{A.59b})$$

Rewriting (6.9) gives:

$$\begin{aligned} (1 - R_1 - R_2)(\tau_1^{\mu B} + \tau_2^{\mu B} + \sigma_{12}^B) - (\sigma_{21}^A + \sigma_{12}^A + R_1(\tau_2^{\lambda B} + \sigma_{21}^B) + R_2\tau_2^{\lambda B}) \geq \\ (1 - R_2)(R_1(\tau_1^{\lambda B} - T) + \tau_1^{\mu B} - \sigma_{21}^A) \quad (\text{A.60a}) \end{aligned}$$

$$(1 - R_1 - R_2)(\tau_1^{\mu B} + \tau_2^{\mu B} + \sigma_{21}^B) - (\sigma_{21}^A + \sigma_{12}^A + R_2(\tau_1^{\lambda B} + \sigma_{12}^B) + R_1 \tau_1^{\lambda B}) \geq (1 - R_1)(R_2(\tau_2^{\lambda B} - T) + \tau_2^{\mu B} - \sigma_{12}^A). \quad (\text{A.60b})$$

Suppose that $R_1 + R_2 \geq 1$, implying that the left hand sides of (A.60) are negative. Combining this with (A.59), this results in $\theta_1^+ < \sigma_{12}^B$ and $\theta_2^+ < \sigma_{21}^B$, meaning that A and B are never working at different lot types simultaneously (cf. Figure 6.10) if working at the desired process cycles. A consequence is that a *virtual station* can be put between workstations A and B , with maximum process rates $\min(\mu_1^A, \mu_i^B)$ and $\min(\mu_2^A, \mu_2^B)$, which does not influence the process cycles of A and B (the virtual station paradigm has been elaborated in [27, Ch.8]). In Figure A.23, the

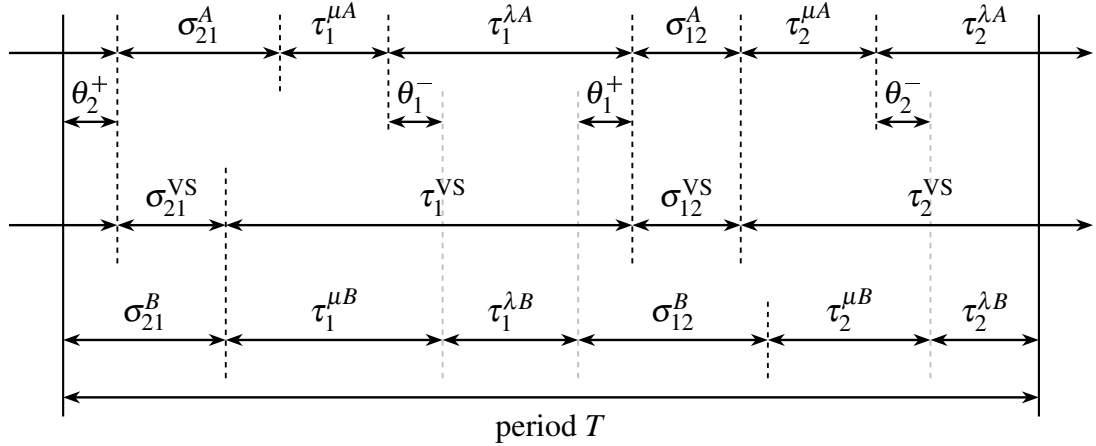


Figure A.23: Virtual station between workstations A and B .

virtual station (VS) has been put between the time lines of A and B . During τ_i^{VS} , the virtual station processes type i lots (with maximum process rate $\min(\mu_i^A, \mu_i^B)$). The process intervals τ_i^{VS} start with the first start of $\tau_i^{\mu j}$ and end at the same moment as θ_i^+ . For reasons of stability, the utilization of the virtual station must not exceed 1: $R_1 + R_2 < 1$. This is in contradiction with our assumption, so under conditions (6.9), $R_1 + R_2 < 1$. \square

Before convergence of the trajectory to the desired (optimal) trajectory is proven, first the formal feedback law of Proposition 6.6 is given.

Initially, the workstations have to get into the same mode $\mathbf{m} = (m^A, m^B)$. Without loss of generality, the mode of A is adjusted to equal the mode of B (if necessary). This is done by means of the following initial control action:

$$\begin{pmatrix} u_0^A \\ u_0^B \\ u_1^A \\ u_2^A \\ u_1^B \\ u_2^B \end{pmatrix}^T = \begin{cases} (\mathbf{1}, \mathbf{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^B > 0 \\ (\mathbf{1}, \mathbf{1}, 0, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (2, 1), x_0^B = 0, x_1^B > 0 \\ (\mathbf{1}, \mathbf{1}, 0, 0, \lambda_1, 0) & \text{if } \mathbf{m} = (2, 1), x_0^B = 0, x_1^B = 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, \mu_2^B) & \text{if } \mathbf{m} = (1, 2), x_0^B = 0, x_2^B > 0 \\ (\mathbf{2}, \mathbf{2}, 0, 0, 0, \lambda_2) & \text{if } \mathbf{m} = (1, 2), x_0^B = 0, x_2^B = 0. \end{cases} \quad (\text{A.61})$$

Immediately after this initial action (if necessary), the following feedback is used:

$$\begin{pmatrix} u_0^A \\ u_0^B \\ u_1^A \\ u_2^A \\ u_1^B \\ u_2^B \end{pmatrix}^T = \begin{cases} (\textcircled{1}, \textcircled{1}, \mu_1^A, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A > 0, x_1^B > 0 \\ (\textcircled{1}, \textcircled{1}, \mu_1^A, 0, \min(\mu_1^A, \mu_1^B), 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A > 0, x_1^B = 0 \\ (\textcircled{1}, \textcircled{1}, \lambda_1, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_1^B > 0 \\ (\textcircled{1}, \textcircled{1}, \lambda_1, 0, \lambda_1, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = x_0^B = x_1^A = x_1^B = 0, x_2^A < x_2^{B\#} \\ (\textcircled{1}, \textcircled{2}, \lambda_1, 0, 0, \mu_2^B) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_2^B > 0 \\ (\textcircled{1}, \textcircled{2}, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B = 0, x_1^A = 0, x_2^B = 0 \\ (\textcircled{1}, \bullet, \mu_1^A, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B > 0, x_1^A > 0 \\ (\textcircled{1}, \bullet, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = 0, x_0^B > 0, x_1^A = 0 \\ (\textcircled{1}, \bullet, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B > 0, x_1^A = 0, x_1^B < x_1^{B\#} \\ (\textcircled{1}, \bullet, \lambda_1, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A = x_0^B = x_1^A = x_1^B = 0, x_2^A \geq x_2^{A\#} \\ (\textcircled{2}, \textcircled{1}, 0, \lambda_2, \mu_1^B, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = x_0^B = x_2^A = 0, x_1^B > 0, x_2^B < x_2^{B\#} \\ (\textcircled{2}, \textcircled{1}, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = x_0^B = x_2^A = x_1^B = 0, x_2^B < x_2^{B\#} \\ (\textcircled{2}, \textcircled{2}, 0, \mu_2^A, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A > 0, x_2^B > 0 \\ (\textcircled{2}, \textcircled{2}, 0, \mu_2^A, 0, \min(\mu_2^A, \mu_2^B)) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A > 0, x_2^B = 0 \\ (\textcircled{2}, \textcircled{2}, 0, \lambda_2, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A = 0, x_0^B = 0, x_2^A = 0, x_2^B > 0 \\ (\textcircled{2}, \textcircled{2}, 0, \lambda_2, 0, \lambda_2) & \text{if } \mathbf{m} = (2, 2), x_0^A = x_0^B = 0, x_2^A = x_2^B = 0, x_1^A < x_1^{A\#} \\ (\textcircled{2}, \bullet, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A = x_0^B = 0, x_2^A = x_2^B = 0, x_1^A \geq x_1^{A\#} \\ (\textcircled{2}, \bullet, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = 0, x_0^B > 0, x_2^A = 0, x_2^B < x_2^{B\#} \\ (\textcircled{2}, \bullet, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^B > 0, x_2^A > 0 \\ (\textcircled{2}, \bullet, 0, \lambda_2, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^B > 0, x_2^A = 0 \\ (\bullet, \textcircled{1}, 0, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B = 0, x_1^B > 0 \\ (\bullet, \textcircled{1}, 0, 0, \mu_1^B, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = x_0^B = x_2^A = 0, x_1^B > 0, x_2^B \geq x_2^{B\#} \\ (\bullet, \textcircled{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B = 0, x_1^B = 0 \\ (\bullet, \textcircled{1}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = 0, x_0^B = 0, x_1^B = 0, x_2^B \geq x_2^{B\#} \\ (\bullet, \bullet, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 1), x_0^A > 0, x_0^B > 0 \\ (\bullet, \bullet, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 1), x_0^A = 0, x_0^B > 0, x_2^B \geq x_2^{B\#} \\ (\bullet, \textcircled{2}, 0, 0, 0, \mu_2^B) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B = 0, x_2^B > 0 \\ (\bullet, \textcircled{2}, 0, 0, 0, \mu_2^B) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B = 0, x_2^B > 0, x_1^B \geq x_1^{B\#} \\ (\bullet, \textcircled{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B = 0, x_2^B = 0 \\ (\bullet, \textcircled{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B = 0, x_2^B = 0, x_1^B \geq x_1^{B\#} \\ (\bullet, \textcircled{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (2, 2), x_0^A > 0, x_0^B > 0 \\ (\bullet, \textcircled{2}, 0, 0, 0, 0) & \text{if } \mathbf{m} = (1, 2), x_0^A = 0, x_0^B > 0, x_1^A = 0, x_1^B \geq x_1^{B\#} \end{cases} \quad (\text{A.62})$$

in which $x_1^{B\#} = \lambda_1 \theta_1^+$, $x_2^{B\#} = \lambda_2 \theta_2^+$, $x_1^{A\#} = \lambda_1 (\sigma_{12}^A + \tau_2^{\mu A} + \tau_2^{\lambda A} - \theta_2^+)$ and $x_2^{A\#} = \lambda_2 (\sigma_{21}^A + \tau_1^{\mu A} + \tau_1^{\lambda A} - \theta_1^+)$.

In both the desired (optimal) trajectory and the transient, the system loops the following modes $\mathbf{m} = (m^A, m^B)$: $(1, 1) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (2, 1) \rightarrow (1, 1) \rightarrow \dots$. For the desired trajectory, the buffer levels after leaving modes (m^A, m^B) are:

$$\begin{aligned}
 \text{After mode } (1, 1) : \quad & \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ x_2^{A\#} \\ x_2^{B\#} \end{bmatrix} \\
 \text{After mode } (1, 2) : \quad & \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} 0 \\ x_1^{B\#} \\ x_2^{A\#} + \lambda_2 \theta_1^+ \\ x_2^{B\#} - \mu_2^B \max(\theta_1^+ - \sigma_{12}^B, 0) \end{bmatrix} \\
 \text{After mode } (2, 2) : \quad & \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\#} \\ x_1^{B\#} \\ 0 \\ 0 \end{bmatrix} \\
 \text{After mode } (2, 1) : \quad & \begin{bmatrix} x_1^A \\ x_1^B \\ x_2^A \\ x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\#} + \lambda_1 \theta_2^+ \\ x_1^{B\#} - \mu_1^B \max(\theta_2^+ - \sigma_{21}^B, 0) \\ 0 \\ x_2^{B\#} \end{bmatrix}.
 \end{aligned} \tag{A.63}$$

The duration of mode $\mathbf{m} = (1, 1)$ is $x_2^{A\#}/\lambda_2$, whereas the duration of mode $(2, 2)$ equals $x_1^{A\#}/\lambda_1$. Furthermore, mode $(1, 2)$ always takes θ_1^+ and mode $(2, 1)$ always takes θ_2^+ (this follows directly from the controller description).

Suppose that mode $(1, 2)$ is entered in the transient for the n^{th} time ($n > 1$). The buffer levels are at this point:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} 0 & 0 & x_2^{A\#} + X^{(n)} & x_2^{B\#} \end{bmatrix} \tag{A.64}$$

where $X^{(n)} \geq 0$ represents the additional buffer content with respect to the steady state value, when starting mode $(1, 2)$ for the n^{th} time. What are the buffer levels after mode $(2, 2)$, and consequently after mode $(1, 1)$? Basically, the map $X^{(n+1)} = f(X^{(n)})$ is looked for. With this map, convergence might be proven. However, instead of deriving the map f explicitly, an easy to find upper bound for $X^{(n+1)}$ is determined here by means of an alternative control strategy.

Consider the alternative control strategy that first goes through mode $(1, 2)$ during θ_1^+ and then stays in mode $(2, 2)$ during $x_1^{A\#}/\lambda_1$, as if it were on the desired orbit. The resulting buffer levels

are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} 0 & 0 & X^{(n)} & x_2^{B\#} \end{bmatrix}. \quad (\text{A.65})$$

Assume that A and B both process type 2 lots at rate $\min(\mu_2^A, \mu_2^B)$ to empty buffer x_2^A . This takes another $X^{(n)} / (\min(\mu_2^A, \mu_2^B) - \lambda_2)$ time units. The resulting buffer levels after this step are then:

$$\begin{bmatrix} x_1^A & x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{A\#} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)} & x_1^{B\#} & 0 & 0 \end{bmatrix}. \quad (\text{A.66})$$

With the original controller of Proposition 6.6, different values for the buffer levels are obtained. However, the original controller processes at least as much lots as the alternative controller, at each time instant, because the original controller always processes lots at the highest possible rate (cf. Lemma 5.2). For this reason, it is known that for the real controller at the end of mode $(2, 2)$, $\begin{bmatrix} x_1^B & x_2^A & x_2^B \end{bmatrix} = \begin{bmatrix} x_1^{B\#} & 0 & 0 \end{bmatrix}$ and for x_1^A :

$$x_1^{A\#} \leq x_1^A \leq x_1^{A\#} + \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} X^{(n)}. \quad (\text{A.67})$$

Completing the controller cycle for modes $(2, 1)$ and $(1, 1)$, similar reasoning leads to the following result:

$$0 \leq X^{(n+1)} \leq \frac{\lambda_1}{\min(\mu_2^A, \mu_2^B) - \lambda_2} \cdot \frac{\lambda_2}{\min(\mu_1^A, \mu_1^B) - \lambda_1} \cdot X^{(n)} \quad (\text{A.68})$$

or rewritten:

$$0 \leq X^{(n+1)} \leq \frac{R_1}{1 - R_1} \cdot \frac{R_2}{1 - R_2} \cdot X^{(n)}. \quad (\text{A.69})$$

Since $R_1 + R_2 < 1$ (result of Lemma A.5), the following can be concluded:

$$\lim_{n \rightarrow \infty} X^{(n)} = 0 \quad (\text{A.70})$$

which means that the system converges to the desired (optimal) periodic orbit (cf. (A.64) with the result of (A.70)).

Matlab and χ models

B.1 Chapter 4: Receding horizon control using (multi-parametric) linear programming

MPLP problem formulation for flow line of multiple workstations

The following MATLAB script computes the feedback control law for a flow line consisting of multiple workstations. Each workstation consists of a buffer with finite capacity and a single-lot machine. The code has been commented shortly. A more elaborate explanation is given here.

- In lines 1–6 the parameters need to be supplied by the user. In vector \mathbf{N} the number of buffer places in each workstation is supplied, from upstream (first workstation) to downstream (last workstation). Vector \mathbf{m} contains the process times of the machines in the workstations. Variable hor represents the control horizon (the number of lots that are optimized).
- In lines 8–21 all subproblems are constructed. Each column in the resulting matrix `subproblems` represents a subproblem, containing an instantiation of the numbers of lots in the buffers and on the machines.
- Lines 23–228 deal with constructing all constraint matrices for each subproblem and solving the MPLP problem.
- In lines 27–38 the dimensions of vectors \mathbf{u} , \mathbf{v} , \mathbf{w} and \mathbf{y} are computed for each workstation, based on the subproblem configuration (numbers of lots in buffers and on machines) and the control horizon hor , which is the dimension of vector \mathbf{z} .
- In lines 40–177 the constraints are constructed which are present in each workstation. The

constraints are divided into groups, starting with a capital letter. The small letter behind it indicates on which design variable the submatrix has effect. For example, submatrix Bw contains the elements for the **w** vector of a workstation, regarding the constraints that say that jobs can only start after authorization. In several occasions, constraint submatrices are grouped, e.g. in submatrix Jvwy, which contains the elements of constraints that affect **v**, **w** and **y**.

- In lines 120–176 the submatrices are concatenated to each other. All if constructs ensure that no errors occur due to bad concatenation (concatenation of empty matrices).
- Lines 178–202 construct the remaining constraints that affect the flow line as a whole: bounds on the exploration space and constraints on dummy variables in vector **z**, that deal with the absolute value function in the objective function.
- In lines 204–226 the big matrices are constructed that form the inequality $\mathbf{GU} \leq \mathbf{W} + \mathbf{EX}$. Also the objective function vectors and additional constraint matrices that bound the exploration space are constructed: $\mathbf{AX} \leq \mathbf{b}$.
- In line 227 the MPLP problem solver is invoked.
- Lines 228–229 are used to store the results of the solver, in order to have them available in an implementation of the controller.

```

1  clear
   lambda1=0.001; % Weighing factor for first subordinate control goal
   lambda2=0.0001; % Weighing factor for second subordinate control goal
   C=[2 4 2]; % Buffer capacities in workstations ( >=1 ).
5  d=[4 3 2]; % Process times of machines. Upstream -> downstream
   hor=4; % Control horizon (>= 1)
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   nws=length(C); % number of workstations
   % Generate all subproblems.
10  nrsp=2^nws*prod(C+1); % number of subproblems
   machinepart= repmat(de2bi(0:2^nws-1),prod(C+1),1);
   bufferpart=[];
   for temp=1:nws
       step=prod(C(temp+1:end)+1)*2^nws;
15       column=[];
       for j=0:C(temp)
           column=[column;j*ones(step,1)];
       end
       bufferpart=[bufferpart repmat(column,nrsp/length(column),1)];
20  end
   subproblems=[bufferpart machinepart]';
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Solve the subproblems
   xregionsset=[]; PN=[]; FI=[]; GI=[]; % initialization lumped solution matrices.
25  % X: due dates r (dim hor), time (dim 1) and remaining process times x3 (dim nws)
   for p=subproblems
       % Compute dimensions of u v w y z vectors
       x1=p(1:nws); % number of jobs in buffers from upstream to downstream
       x2=p(nws+1:end); % number of jobs on machines from upstream to downstream
30      dimu(nws+1)=hor;
       for temp=nws:-1:1
           dimy(temp)=dimu(temp+1);
           dimv(temp)=max(dimy(temp)-x2(temp),0);

```

```

        dimw(temp)=dimv(temp);
35      dimu(temp)=max(dimw(temp)-x1(temp),0);
    end
    dimu=dimu(1:nws);
    dimz=hor;
    G={}; W={}; E={}; % initialization for each subproblem constraint matrices.
40  for ws=1:nws
        % now construct the submatrices that apply to each workstation
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % w>=u (jobs can only start after their arrival)
        rA=dimu(ws);
45      Au{ws}=[eye(rA)];          Av{ws}=zeros(rA,dimv(ws));
        Aw{ws}=-diag(ones(1,rA),x1(ws)); Aw{ws}=Aw{ws}(1:rA,1:dimw(ws));
        Ay{ws}=zeros(rA,dimv(ws));    Ap{ws}=zeros(rA,1);
        Artx3{ws}=zeros(rA,hor+1+nws);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50      % w>=v (jobs can only start after authorization)
        rB=dimv(ws);
        Bu{ws}=zeros(rB,dimu(ws));    Bv{ws}=eye(rB);
        Bw{ws}=-eye(rB);              By{ws}=zeros(rB,dimv(ws));
        Bp{ws}=zeros(rB,1);           Brtx3{ws}=zeros(rB,hor+1+nws);
55      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % y>=w+d (jobs can only leave after processing has finished)
        rC=max(dimv(ws)-x2(ws),0);
        Cuv{ws}=zeros(rC,dimu(ws)+dimv(ws)); Cw{ws}=eye(rC);
        Cy{ws}=-diag(ones(1,rC),x2(ws));    Cy{ws}=Cy{ws}(1:rC,:);
60      Cp{ws}=-d(ws)*ones(rC,1);           Crtx3{ws}=zeros(rC,hor+1+nws);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % w>=y (a new job can only start once the previous job has left)
        rD=max(dimv(ws)-1,0);
        Duv{ws}=zeros(rD,dimu(ws)+dimv(ws)); Dw{ws}=-diag(ones(1,rD),1-x2(ws));
65      Dw{ws}=Dw{ws}(1:rD,:);              Dy{ws}=[eye(rD) zeros(rD,1)];
        Dp{ws}=zeros(rD,1);                 Drtx3{ws}=zeros(rD,hor+1+nws);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % u>=w (incorporate buffer level constraints)
        rE=max(dimv(ws)-C(ws)-x2(ws),0);
70      Eu{ws}=[zeros(rE,dimv(ws)-rE) -eye(rE)]; Ev{ws}=zeros(rE,dimv(ws));
        Ew{ws}=[eye(rE) zeros(rE,dimw(ws)-rE)]; Ey{ws}=zeros(rE,dimv(ws));
        Ep{ws}=zeros(rE,1);                 Ertx3{ws}=zeros(rE,hor+1+nws);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
75      % y1>=x3+t (if x2==1, first lot can leave after remaining process time)
        Hu={}; Hv={}; Hw={}; Hy={}; Hp={}; Hr={}; Ht={}; Hx3={};
        if x2(ws)==1 && dimv(ws)>0
            rH=1;
            Hu{ws}=zeros(rH,dimv(ws));      Hv{ws}=zeros(rH,dimv(ws));
            Hw{ws}=zeros(rH,dimw(ws));      Hy{ws}=zeros(rH,dimv(ws));
80          Hp{ws}=zeros(rH,1);              Hr{ws}=zeros(rH,hor);
            Ht{ws}=zeros(rH,1);              Hx3{ws}=zeros(rH,nws);
            Hy{ws}(1,1)=-1;                  Ht{ws}(1,1)=-1;
            Hx3{ws}(1,ws)=-1;
        end
85      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % u>=t v>=t w>=t y>=t (we cannot change the past)
        rI=dimu(ws)+dimv(ws)+dimw(ws)+dimv(ws);
        Iuvwy{ws}=-eye(rI);                Ip{ws}=zeros(rI,1);
        Ir{ws}=zeros(rI,hor);               It{ws}=-1*ones(rI,1);
90      Ix3{ws}=zeros(rI,nws);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% u_{i+1} >= u_i (jobs arrive in chronological order)
rJ=max(dimu(ws)-1,0);
Ju{ws}=[eye(rJ) zeros(rJ,1)]+[zeros(rJ,1) -eye(rJ)];
95 Jvwy{ws}=zeros(rJ,dimv(ws)+dimw(ws)+dimy(ws));
Jp{ws}=zeros(rJ,1); Jrtx3{ws}=zeros(rJ,hor+nws+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% v_{i+1} >= v_i (jobs are authorized in chronological order)
rK=max(dimv(ws)-1,0);
100 Ku{ws}=zeros(rK,dimv(ws));
Kv{ws}=[eye(rK) zeros(rK,1)]+[zeros(rK,1) -eye(rK)];
Kwy{ws}=zeros(rK,dimw(ws)+dimy(ws));
Kp{ws}=zeros(rK,1);
Krtx3{ws}=zeros(rK,hor+nws+1);
105 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% w_{i+1} >= w_i (jobs are started in chronological order)
rL=max(dimw(ws)-1,0);
Luv{ws}=zeros(rL,dimv(ws)+dimw(ws));
Lw{ws}=[eye(rL) zeros(rL,1)]+[zeros(rL,1) -eye(rL)];
110 Ly{ws}=zeros(rL,dimv(ws));
Lp{ws}=zeros(rL,1);
Lrtx3{ws}=zeros(rL,hor+nws+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% y_{i+1} >= y_i (jobs leave in chronological order)
115 rM=max(dimy(ws)-1,0);
Muvw{ws}=zeros(rM,dimv(ws)+dimw(ws)+dimw(ws));
My{ws}=[eye(rM) zeros(rM,1)]+[zeros(rM,1) -eye(rM)];
Mp{ws}=zeros(rM,1);
Mrtx3{ws}=zeros(rM,hor+nws+1);
120 % construct the full matrices and vectors. If's to prevent errors.
%G{ws}={}; W{ws}={}; E{ws}={};
if ~isempty([Iuvwy{ws}])
    G{ws}=Iuvwy{ws};
    W{ws}=Ip{ws};
125 E{ws}=[Ir{ws} It{ws} Ix3{ws}];
end
if ~isempty([Au{ws} Av{ws} Aw{ws} Ay{ws}])
    G{ws}=[G{ws}; Au{ws} Av{ws} Aw{ws} Ay{ws}];
    W{ws}=[W{ws}; Ap{ws}];
130 E{ws}=[E{ws}; Artx3{ws}];
end
if ~isempty([Bu{ws} Bv{ws} Bw{ws} By{ws}])
    G{ws}=[G{ws}; Bu{ws} Bv{ws} Bw{ws} By{ws}];
    W{ws}=[W{ws}; Bp{ws}];
135 E{ws}=[E{ws}; Brtx3{ws}];
end
if ~isempty([Cuv{ws} Cw{ws} Cy{ws}])
    G{ws}=[G{ws}; Cuv{ws} Cw{ws} Cy{ws}];
    W{ws}=[W{ws}; Cp{ws}];
140 E{ws}=[E{ws}; Crtx3{ws}];
end
if ~isempty([Duv{ws} Dw{ws} Dy{ws}])
    G{ws}=[G{ws}; Duv{ws} Dw{ws} Dy{ws}];
    W{ws}=[W{ws}; Dp{ws}];
145 E{ws}=[E{ws}; Drtx3{ws}];
end
if ~isempty([Eu{ws} Ev{ws} Ew{ws} Ey{ws}])

```



```

        G{ws}=[G{ws};Eu{ws} Ev{ws} Ew{ws} Ey{ws}];
        W{ws}=[W{ws};Ep{ws}];
150      E{ws}=[E{ws};Ertx3{ws}];
      end
      if x2(ws)==1 && dimy(ws)>0
        G{ws}=[G{ws};Hu{ws} Hv{ws} Hw{ws} Hy{ws}];
        W{ws}=[W{ws};Hp{ws}];
155      E{ws}=[E{ws};Hr{ws} Ht{ws} Hx3{ws}];
      end
      if ~isempty([Ju{ws} Jvwy{ws}])
        G{ws}=[G{ws};Ju{ws} Jvwy{ws}];
        W{ws}=[W{ws};Jp{ws}];
160      E{ws}=[E{ws};Jrtx3{ws}];
      end
      if ~isempty([Ku{ws} Kv{ws} Kwy{ws}])
        G{ws}=[G{ws};Ku{ws} Kv{ws} Kwy{ws}];
        W{ws}=[W{ws};Kp{ws}];
165      E{ws}=[E{ws};Krtx3{ws}];
      end
      if ~isempty([Luv{ws} Lw{ws} Ly{ws}])
        G{ws}=[G{ws};Luv{ws} Lw{ws} Ly{ws}];
        W{ws}=[W{ws};Lp{ws}];
170      E{ws}=[E{ws};Lrtx3{ws}];
      end
      if ~isempty([Muvw{ws} My{ws}])
        G{ws}=[G{ws};Muvw{ws} My{ws}];
        W{ws}=[W{ws};Mp{ws}];
175      E{ws}=[E{ws};Mrtx3{ws}];
      end
      end
      end
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % Construct the remaining constraints %
180      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % z>=y-r (dummy variable z to deal with |y-r| in objective function)
      rF=hor;
      Fy=eye(rF);          Fz=-eye(rF);          Fp=zeros(rF,1);
      Fr=eye(rF);          Ftx3=zeros(rF,1+nws);
185      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % z>=r-y (dummy variable z to deal with |y-r| in objective function)
      rG=hor;
      Gy=-eye(rG);          Gz=-eye(rG);          Gp=zeros(rG,1);
      Gr=-eye(rG);          Gtx3=zeros(rG,1+nws);
190      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % r_{i+1} >= r_i (due dates have to be in chronological order)
      rN=max(hor-1,0);
      Np=zeros(rN,1);      Nr=[eye(rN) zeros(rN,1)]+[zeros(rN,1) -eye(rN)];
      Nt=zeros(rN,1);      Nx3=zeros(rN,nws);
195      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % x3<=d && x3>=0 (remaining proc. times >=0 and have an upper bound)
      r0=2*nws;            % In Op d 10* enlarged to handle stochastic implementation
      Op=[10*d';zeros(nws,1)];      Or=zeros(r0,hor);
      Ot=zeros(r0,1);          Ox3=[eye(nws); -eye(nws)];
200      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % r>=0, t>=0 (only positive times are considered)
      Prt=-eye(hor+1);      Pp=zeros(hor+1,1);      Px3=zeros(hor+1,nws);
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      % Construct the big matrices for the GG*U<=WW+EE*X form.

```

```

205 % G matrices are put under each other while the y and u variables overlap
% to make them equal to each other. W and E matrices are concatenated.
GG=G{1};WW=W{1};EE=E{1};
for temp=2:nws
    zerosleftlower=zeros(size(G{temp},1),size(GG,2)-dimu(temp));
210    zerosrightupper=zeros(size(GG,1), dimv(temp)+dimw(temp)+dimy(temp));
    GG=[GG zerosrightupper; zerosleftlower G{temp}];
    WW=[WW; W{temp}]; EE=[EE; E{temp}];
end
% Add constraints on dummy variables z
215 GG=[GG zeros(size(GG,1),hor); zeros(2*hor,size(GG,2)-dimy(end)) [Fy Fz;Gy Gz]];
    WW=[WW; Fp; Gp];
    EE=[EE; Fr Ftx3; Gr Gtx3];
    HH=-lambda1*ones(1,dimv(1)+dimw(1));
for temp=2:nws
220    HH=[HH lambda2*ones(1,dimv(temp)) -lambda1*ones(1,dimv(temp)+dimw(temp))];
end
    HH=[HH zeros(1,dimy(end)) ones(1,dimz)];
    FF=[zeros(1,hor) 0 zeros(1,nws)]; % objective function does not depend on X
    Matrices.H=HH; Matrices.F=FF; Matrices.G=GG; Matrices.W=WW; Matrices.E=EE;
225 Matrices.bndA=[Nr Nt Nx3; Or Ot Ox3; Prt Px3];
    Matrices.bndb=[Np; Op; Pp];
    [Pn,Fi,Gi,actCon,Phard,details] = mpt_mplp(Matrices, struct('mplpver', 6));
    xregionsset=[xregionsset; x1' x2' details.nRegions];
    PN=[PN Pn]; FI=[FI Fi]; GI=[GI Gi];
230 end

```

In order to compute control action vector \mathbf{U} for arbitrary state, due dates and time, the following MATLAB function can be used:

```

1 function U=computecontrolaction(xregionsset, PN, FI, GI, xsel, x)
% xsel: selects the solution of one of the MPLP problems based on [x1s x2s]
% x: parameter vector which is fed to function getOptimizer (from the MPT toolbox)
for j=1:size(xregionsset,1)
5     if xregionsset(j,1:end-1)==xsel
        nr=j; break
    end
end
if nr>1
10     beforenr=sum(xregionsset(1:nr-1,end));
    else
        beforenr=0;
    end
    tempPN=PN(beforenr+1:beforenr+xregionsset(nr,end));
15    tempFI=FI(beforenr+1:beforenr+xregionsset(nr,end));
    tempGI=GI(beforenr+1:beforenr+xregionsset(nr,end));
    U=mpt_getOptimizer(tempPN,tempFI,tempGI,x);

```

As an example, consider a flow line consisting of three workstations. At current time $t = 0$, the numbers of lots in the buffers are (from upstream to downstream) 1, 4 and 2 respectively. Machines 1 and 3 are empty and machine 2 is processing a product, with a remaining process time of 2.5 hours. Suppose that the control horizon was 4 lots and the due dates are 10, 11, 12 and 13. The control action vector \mathbf{U} can be computed with the following commands:

```

1  x1s=[1 4 2];
   x2s=[0 1 0];
   xsel=[x1s x2s];
   duedates=[10 11 12 13];
5  time=0;
   x3s=[0 2.5 0];
   x=[duedates time x3s];
   U=computecontrolaction(xregionsset, PN, FI, GI, xsel, x)

```

B.2 Chapter 5: A switching server

Hybrid fluid model in Matlab

The following MATLAB script is an implementation of the hybrid fluid model that has been developed in Chapter 5. It uses an internal discrete event simulation to compute the time instants and state changes of all events that can take place given the current state and time. Then it takes the earliest occurring event and updates the clock and state. A more detailed description of the script is given first, followed by the code itself.

First all necessary system parameters are provided (lines 3–9). Then some auxiliary variables are computed, like the partial relative workloads ρ_i (lines 11–15). The initial condition of the system (buffer levels, mode and remaining setup time) are also needed before the simulation can start (lines 17–25). The occurrence of a slowmode is investigated, based on the condition in Theorem 5.10 (lines 26–34). As a final step before the actual simulation starts, the buffer levels at which a setup to the other lot type has to take place are computed, taking into account the maximum buffer capacities (lines 36–44). The simulator repeats the following steps:

- Based on the current state, the process rates of the lot types are determined, as specified in (5.1), taking into account that lots are always served at the highest possible rate (Lemma 5.2). Lines 50–68.
- The process rates and current state determine which events can take place. The time spans until the occurrence of these events are placed in the event matrix, together with the values of the state elements in case this event would take place at that particular time instant. Every time the simulator enters this step, the event matrix is emptied before all possible new events are computed. Lines 69–103.
- The earliest occurring event is taken from the event matrix. Line 106.
- The state and time are updated according to the occurring event. Lines 107–111.

If multiple events can take place at a certain time instant, only one of them is executed (the min-function in MATLAB returns only one element in case of multiple minima). In the next loop, the events that can still take place will automatically show up in the new event vector and will be executed then.

```

1  clear
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % System parameters
   lambda1=9;
5  lambda2=3;
   mu1=24;
   mu2=27;
   sigma12=2;
   sigma21=2;
10  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Auxiliary variables
   rho1=lambda1/mu1;
   rho2=lambda2/mu2;
   sigma=sigma12+sigma21;
15  sigmas=[sigma21 sigma12];
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Initial conditions
   x0=0;
   x1=50;
20  x2=20;
   m=2;
   x1max=70;
   x2max=40;
   c1=1;
25  c2=1;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Investigate slow-mode
   if c1*lambda1*(rho1+rho2)+(c2*lambda2-c1*lambda1)*(1-rho2)>=0
       alpha=0;
30  else
       alpha=max(roots([c1*lambda1*rho2^2*(1-rho1)+c2*lambda2*(1-rho1)^2*(1-rho2)
                        2*(c1*lambda1*rho2^2+c2*lambda2*(1-rho1)*(1-rho2))
                        c1*lambda1*(rho1+rho2)+(c2*lambda2-c1*lambda1)*(1-rho2)]));
   end
35  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % The coordinates for m switching with buffer level constraints
   x1sharp=lambda1*(sigma12+(sigma*rho2*(1+alpha*(1-rho1)))/(1-rho1-rho2));
   x2sharp=lambda2*(sigma21+(sigma*(alpha*(1-rho1)*(1-rho2)+rho1))/(1-rho1-rho2));
   x1barsharp=min([ x1max-lambda1*sigma21;
40                    lambda1*(sigma21+x2max/(mu2-lambda2));
                    x1sharp ]);
   x2barsharp=min([ (mu2-lambda2)/lambda1*(x1max-lambda1*sigma)-lambda2*sigma12;
                    x2max-lambda2*sigma12;
                    x2sharp ]);
45  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % START OF SIMULATION
   t=0;
   output=[t x0 x1 x2 m];
   while t<50 % Duration of the simulation
50     % Compute process rates workstation
       if m==1 && x0==0
           if x1>0
               u1=mu1;
           elseif x1==0
55               u1=lambda1;
           end
           u2=0;

```

```

elseif m==2 && x0==0
    u1=0;
60    if x2>0
        u2=mu2;
    elseif x2==0
        u2=lambda2;
    end
65    elseif x0>0
        u1=0;
        u2=0;
    end
    % Clear the event matrix
70    events=[];
    % Now compute which events can take place and when.
    % Event matrix: [timestep x0 x1 x2 m]
    if x0>0
        events=[events; x0 0 x1+(lambda1-u1)*x0 x2+(lambda2-u2)*x0 m];
75    end
    if x1>0 && u1>0
        step=x1/(u1-lambda1);
        events=[events; step 0 0 x2+(lambda2-u2)*(step) m];
    end
80    if x1>0 && u1>0
        step=(x2max-lambda2*sigma12-x2)/lambda2;
        events=[events;
                step sigmas(3-m) x1+(lambda1-u1)*step x2+(lambda2-u2)*step 3-m];
    end
85    if x2>0 && u2>0
        step=x2/(u2-lambda2);
        events=[events; step 0 x1+(lambda1-u1)*step 0 m];
    end
    if x2>0 && u2>0
90    step=(x1max-lambda1*sigma21-x1)/lambda1;
        events=[events;
                step sigmas(3-m) x1+(lambda1-u1)*step x2+(lambda2-u2)*step 3-m];
    end
    if m==1 && x1==0 && u1>0
95    step=max([0; (x2barsharp-x2)/lambda2]);
        events=[events;
                step sigmas(3-m) x1+(lambda1-u1)*step x2+(lambda2-u2)*step 3-m];
    end
    if m==2 && x2==0 && u2>0
100    step=max([0; (x1barsharp-x1)/lambda1]);
        events=[events;
                step sigmas(3-m) x1+(lambda1-u1)*step x2+(lambda2-u2)*step 3-m];
    end
    % Take the earliest event, update buffer levels, mode, remaining setup
105    % time and put forward the clock.
    [q,i]=min(events(:,1));
    x0=events(i,2);
    x1=events(i,3);
    x2=events(i,4);
110    m=events(i,5);
    t=t+q;
    output=[output; t events(i,2:end)];
end

```

One workstation with two lot types in χ

The following χ model represents a workstation consisting of buffer B and machine M . Jobs are generated by generator G and arrive at the server with constant arrival rate. All process times, setup times and interarrival times are constant.

The iconic representation of the χ model is shown in Figure B.1. Channels a , b , c and d are used to send and receive lots between processes. Channel e (dashed line in the figure) is used to switch the mode of the machine. The controller has been implemented in buffer B .

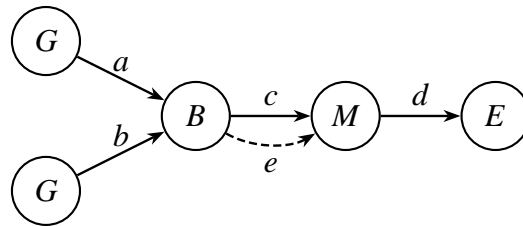


Figure B.1: Iconic representation of χ model of a switching server with two queues.

```

type lot = nat

proc G(chan a! : lot, val  $\lambda$  : real) =
  [[ var i : nat = 0
  :: * (  $\Delta 1/\lambda$ ; i := i + 1; a!i )
  ]]

proc B(chan a?, b?, c! : lot, d! : nat, val  $x_2^\sharp$  : real,  $x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0) : \text{nat}$ ,
 $\lambda_1, \lambda_2, \sigma_{21}, \sigma_{12} : \text{real}$ ) =
  [[ var  $x_1, x_2 : [\text{lot}] = ([ ], [ ])$ ,  $x : \text{lot}, m : \text{nat} = m(0)$ 
  ::  $x_1(0) > 0 * > (x_1 := x_1 ++ [0]; x_1(0) := x_1(0) - 1)$ 
  ;  $x_2(0) > 0 * > (x_2 := x_2 ++ [0]; x_2(0) := x_2(0) - 1)$ 
  ; * ( len( $x_1$ ) <  $x_1^{\max}$                                       $\rightarrow a?x; x_1 := x_1 ++ [x]$ 
      [ len( $x_2$ ) <  $x_2^{\max}$                                       $\rightarrow b?x; x_2 := x_2 ++ [x]$ 
      [  $m = 0 \wedge \text{len}(x_1) > 0 \wedge \text{len}(x_2) < x_2^{\max} - \lambda_2 \sigma_{12}$   $\rightarrow c! \text{hd}(x_1); x_1 := \text{tl}(x_1)$ 
      [  $m = 0 \wedge \text{len}(x_1) > 0 \wedge \text{len}(x_2) \geq x_2^{\max} - \lambda_2 \sigma_{12}$   $\rightarrow d!1; m := 1$ 
      [  $m = 0 \wedge \text{len}(x_1) = 0 \wedge \text{len}(x_2) \geq x_2^\sharp$   $\rightarrow d!1; m := 1$ 
      [  $m = 1 \wedge \text{len}(x_2) > 0 \wedge \text{len}(x_1) < x_1^{\max} - \lambda_1 \sigma_{21}$   $\rightarrow c! \text{hd}(x_2); x_2 := \text{tl}(x_2)$ 
      [  $m = 1 \wedge (\text{len}(x_2) = 0 \vee \text{len}(x_1) \geq x_1^{\max} - \lambda_1 \sigma_{21})$   $\rightarrow d!0; m := 0$ 
      )
  ])
  ]]

proc M(chan a?, b! : lot, c? : nat, val  $\mu, \sigma_{ij} : 2 * \text{real}, m(0) : \text{nat}$ ) =
  [[ var  $x : \text{lot}, m : \text{nat} = m(0)$ 
  :: * ( a?x;  $\Delta 1/\mu.m$ ; b!x [ c?m;  $\Delta \sigma_{ij}.m$  )
  ]]

proc E(chan a?lot) =
  [[ var x : lot
  :: *a?x
  ]]

model S(val  $\lambda_1, \lambda_2, \mu_1, \mu_2, \sigma_{21}, \sigma_{12}, x_2^\sharp : \text{real}, x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0) : \text{nat}$ ) =
  [[ chan a, b, c, d : lot, e : nat
  :: G(a,  $\lambda_1$ )
  || G(b,  $\lambda_2$ )
  || B(a, b, c, e,  $x_2^\sharp, x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0), \lambda_1, \lambda_2, \sigma_{21}, \sigma_{12}$ )
  || M(c, d, e,  $\langle \mu_1, \mu_2 \rangle, \langle \sigma_{21}, \sigma_{12} \rangle, m(0)$ )
  || E(d)
  ]]

```

The following χ specification models a single switching server with two queues and switchover times. The interarrival times and the process times of lots are exponentially distributed. The graphical representation is similar to the deterministic case, Figure B.1.

```

type lot = nat

proc G(chan a! : lot, val  $\lambda$  : real) =
  [[ var i : nat = 0,  $dt_a$  :  $\rightarrow$  real,  $t_a$  : real =  $1/\lambda$ ,  $s$  : real
  ::  $dt_a := \text{exponential}(t_a)$ 
  ;  $(s := \sigma dt_a; \Delta s; i := i + 1; a!i)$ 
  ]]

proc B(chan a?, b?, c! : lot, d! : nat, val  $x_2^\sharp$  : real,  $x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0) : \text{nat}$ ,
 $\lambda_1, \lambda_2, \sigma_{12}, \sigma_{21} : \text{real}$ ) =
  [[ var  $x_1, x_2 : [\text{lot}] = ([ ], [ ])$ ,  $x : \text{lot}, m : \text{nat} = m(0)$ 
  ::  $x_1(0) > 0 * > (x_1 := x_1 ++ [0]; x_1(0) := x_1(0) - 1)$ 
  ;  $x_2(0) > 0 * > (x_2 := x_2 ++ [0]; x_2(0) := x_2(0) - 1)$ 
  ; * ( len( $x_1$ ) <  $x_1^{\max}$   $\rightarrow a?x; x_1 := x_1 ++ [x]$ 
    [ len( $x_2$ ) <  $x_2^{\max}$   $\rightarrow b?x; x_2 := x_2 ++ [x]$ 
    [  $m = 0 \wedge \text{len}(x_1) > 0 \wedge \text{len}(x_2) < x_2^{\max} - \lambda_2 \sigma_{12}$   $\rightarrow c!hd(x_1); x_1 := tl(x_1)$ 
    [  $m = 0 \wedge \text{len}(x_1) > 0 \wedge \text{len}(x_2) \geq x_2^{\max} - \lambda_2 \sigma_{12}$   $\rightarrow d!1; m := 1$ 
    [  $m = 0 \wedge \text{len}(x_1) = 0 \wedge \text{len}(x_2) \geq x_2^\sharp$   $\rightarrow d!1; m := 1$ 
    [  $m = 1 \wedge \text{len}(x_2) > 0 \wedge \text{len}(x_1) < x_1^{\max} - \lambda_1 \sigma_{21}$   $\rightarrow c!hd(x_2); x_2 := tl(x_2)$ 
    [  $m = 1 \wedge (\text{len}(x_2) = 0 \vee \text{len}(x_1) \geq x_1^{\max} - \lambda_1 \sigma_{21})$   $\rightarrow d!0; m := 0$ 
    ]
    )
  ]]

proc M(chan a?, b! : lot, c? : nat, val  $\mu, \sigma_{ij} : 2 * \text{real}, m(0) : \text{nat}$ ) =
  [[ var  $x : \text{lot}, dt_{e,0}, dt_{e,1} : \rightarrow \text{real}, t_e : 2 * \text{real} = \langle 1/\mu.0, 1/\mu.1 \rangle$ ,  $s : \text{real}, m : \text{nat} = m(0)$ 
  ::  $dt_{e,0} := \text{exponential}(t_e.0)$ 
  ;  $dt_{e,1} := \text{exponential}(t_e.1)$ 
  ; * ( ( a?x; ( m = 0  $\rightarrow s := \sigma dt_{e,0}$ 
    [ m = 1  $\rightarrow s := \sigma dt_{e,1}$ 
    )
    ;  $\Delta s$ 
    ; b!x
    )
    [ (c?m;  $\Delta \sigma_{ij}.m$ )
    ]
  )

proc E(chan a? : lot) =
  [[ var x : lot
  :: *a?x
  ]]

model S(val  $\lambda_1, \lambda_2, \mu_1, \mu_2, \sigma_{12}, \sigma_{21}, x_2^\sharp : \text{real}, x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0) : \text{nat}$ ) =
  [[ chan a, b, c, d : lot, e : nat
  :: G(a,  $\lambda_1$ ) || G(b,  $\lambda_2$ )
  || B(a, b, c, e,  $x_2^\sharp, x_1(0), x_2(0), x_1^{\max}, x_2^{\max}, m(0), \lambda_1, \lambda_2, \sigma_{12}, \sigma_{21}$ )
  || M(c, d, e,  $\langle \mu_1, \mu_2 \rangle, \langle \sigma_{12}, \sigma_{21} \rangle, m(0))$  || E(d)
  ]]

```

(B.2)

Computation of mean weighted work in process levels for servers with multiple lot types

The following MATLAB script computes the mean weighted work in process level for a server that processes multiple lot types, given a process cycle order. A clearing policy is assumed, meaning that buffers are always emptied completely before the switchover to the next lot type takes place. Furthermore, a slow-mode (in time units) can be inserted after each mode, before switching to the next mode.

```

1  % maximum process rates for type 1, 2, ...
   mu=[16 16 16];
   % setup times in (from,to)-type format
   sigmas=[0 1 1;
5      1 0 1;
      1 1 0];
   % arrival rates for type 1, 2, ...
   lambda=[4 2 1];
   % weighting factors for type 1, 2, ...
10  c=[1 1 1];
   % the process cycle order
   cycle=[1 2 1 3];
   % amount of slowmode after clearing of the process cycle step (in time units)
   sm=[0 0 0 0];
15  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   rho=lambda./mu;
   rhocycle=lambda(cycle)./mu(cycle);
   extcycle=[cycle cycle(1)];
   totalsetup=0;
20  for j=1:length(cycle)
       totalsetup=totalsetup+sigmas(extcycle(j),extcycle(j+1));
   end
   lc=length(cycle);
   A=[]; B=[]; dblcycle=[cycle cycle]; dblsm=[sm sm];
25  for j=lc+1:length(dblcycle)
       type=dblcycle(j);
       ind=find(dblcycle==type);
       laststep=ind(max(find(ind<j)));
       slows=0; setups=0; a=zeros(1,lc);
30  for steps=laststep:j-1
       slows=slows+dblsm(steps);
       setups=setups+sigmas(dblcycle(steps),dblcycle(steps+1));
       a(mod(steps,lc)+1)=lambda(cycle(j-lc));
   end
35  a(j-lc)=lambda(cycle(j-lc))-mu(cycle(j-lc));
       b=-lambda(cycle(j-lc))*(setups+slows-dblsm(laststep));
       A=[A;a]; B=[B;b];
   end
   taus=A\B;
40  area=-diag(A).*taus.*taus./rhocycle'/2;
   ccycle=c(cycle);
   period=sum(taus)+sum(sm)+totalsetup;
   meanweightedwip=sum(area.*ccycle'/period)
   asfraction=rats(meanweightedwip)

```

Chapter 6: Flow lines of switching servers

Two workstations with two lot types in a χ model

The following χ specifications (B.3) and (B.4) model a flow line consisting of two workstations, serving two different lot types (see Figure B.2). The state feedback controllers of Propositions 6.3 and 6.6 have been implemented in the buffer process B . With this model the control goals are validated. A few notes on the χ specifications:

- The physical processes (four separate buffers, two workstations) have not all been modelled or instantiated as separate processes in χ . The four individual buffers are part of one buffer process B . An iconic model of the χ specifications is shown in Figure B.3.
- For modelling reasons, the lot types are labelled as lot type 0 and lot type 1, instead of 1 and 2 respectively. In addition, workstations are also labelled as workstation 0 and workstation 1, representing A and B respectively. Capitals A and B are also used, especially in model S , where all parameters are fed to the processes.
- Only the physical processes and controller have been modelled. All diagnostics have been omitted here for clarity reasons.
- Apart from the discrete event nature of the χ model, the dynamics differs from the hybrid fluid model dynamics (6.1). Setups cannot be interrupted in the χ model, while this is possible in the hybrid fluid model dynamics. However, the controller of Proposition 6.3 never interrupts a setup, so this difference is not a problem.
- Buffer processes B (which also accommodates the state feedback controller) looks very different from the buffer processes in χ specifications (B.1) and (B.2). In those χ specifications, the individual controller actions are recognizable, resulting in a readable but quite large process. In specifications (B.3) and (B.4) however, folding techniques have been used to make the specification as small as possible. Disadvantage of this approach is that the specification is not intuitively readable anymore.
- Extension of this model to stochastic inter-arrival and process times goes in a similar way as in the single switching server situation. Specifications of the stochastic models have not been included here.

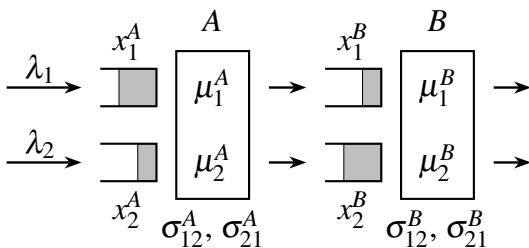


Figure B.2: Flow line of two switching servers.

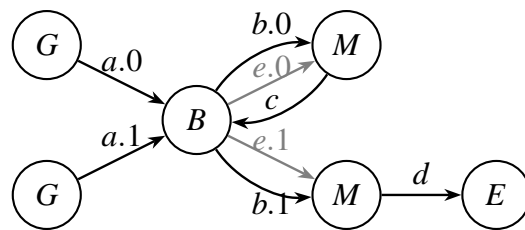


Figure B.3: Iconic model of χ specification (B.3).

```

type lot = nat

proc G(chan a! : lot, val  $\lambda$  : real, i : nat) =
  [[ * (  $\Delta 1/\lambda$ ; a!i )]]

proc B(chan a?, b! : 2#lot, c? : lot, e! : 2#nat, val  $x^{A\#} : 2 * \text{real}$ ,
 $x(0) : 2 * (2 * \text{nat})$ ,  $m^A(0), m^B(0) : \text{nat}$ ) =
  [[ var xs : 2 * (2 * [lot]) = <<[ ], [ ]>, <[ ], [ ]>, x : lot, m : 2 * nat = <mA(0), mB(0)>
  :: ( ; , i  $\leftarrow$  0..1, ( ; , j  $\leftarrow$  0..1, (x(0).i.j > 0) * > (xs.i.j := xs.i.j ++ [j]; x(0).i.j := x(0).i.j - 1)))
  ; * ( ([ , i  $\leftarrow$  0..1, a.i?x; xs.0.i := xs.0.i ++ [x])
    [ c?x; xs.1.x := xs.1.x ++ [x]
    [ ([ , i  $\leftarrow$  0..1, ([ , j  $\leftarrow$  0..1, m.i = j  $\wedge$  len(xs.i.j) > 0  $\rightarrow$  b.i!hd(xs.i.j); xs.i.j := tl(xs.i.j)))
    [ ([ , i  $\leftarrow$  0..1, m.0 = i  $\wedge$  m.1 = i  $\wedge$  len(xs.0.i) = 0  $\wedge$  len(xs.1.i) = 0
       $\wedge$  len(xs.0.(1 - i))  $\geq$   $x^{A\#}.$ (1 - i)  $\rightarrow$  e.0!(1 - i); e.1!(1 - i); m := <1 - i, 1 - i>
    )
  ]
]]

proc M(chan a?, b! : lot, c? : nat, val  $\mu, \sigma : 2 * \text{real}$ , m(0) : nat, x0 : real) =
  [[ var x : lot, m : nat = m(0)
  ::  $\Delta x_0$ 
  ; * ( a?x;  $\Delta 1/\mu.m$ ; b!x
    [ c?m;  $\Delta \sigma.m$ 
  )
]]

proc E(chan a? : lot) =
  [[ var x : lot
  :: *a?x
  ]]

model S(val  $\lambda_1, \lambda_2, \mu_1^A, \mu_2^A, \mu_1^B, \mu_2^B, \sigma_{21}^A, \sigma_{12}^A, \sigma_{21}^B, \sigma_{12}^B, x_1^{A\#}, x_2^{A\#} : \text{real}$ ,
 $x_1^A(0), x_2^A(0), x_1^B(0), x_2^B(0), m^A(0), m^B(0) : \text{nat}$ ,  $x_0^A(0), x_0^B(0) : \text{real}$ ) =
  [[ chan a, b : 2#lot, c, d : lot, e : 2#nat
  :: G(a.0,  $\lambda_1$ , 0)
  [ G(a.1,  $\lambda_2$ , 1)
  [ B(a, b, c, e, < $x_1^{A\#}, x_2^{A\#}$ >, << $x_1^A(0), x_2^A(0)$ >, < $x_1^B(0), x_2^B(0)$ >>,  $m^A(0), m^B(0)$ )
  [ M(b.0, c, e.0, < $\mu_1^A, \mu_2^A$ >, < $\sigma_{21}^A, \sigma_{12}^A$ >,  $m^A(0), x_0^A(0)$ )
  [ M(b.1, d, e.1, < $\mu_1^B, \mu_2^B$ >, < $\sigma_{21}^B, \sigma_{12}^B$ >,  $m^B(0), x_0^B(0)$ )
  [ E(d)
  ]
]]

```

(B.3)

The following χ specification models the flow line of two workstations (Figures B.2 and B.3), with the controller of Proposition 6.6 implemented in buffer process B . Control goal in this specification is to make the flow line behave as if it were the downstream workstation stand-alone, with respect to work in process levels.

```

type lot = nat
proc G(chan a! : lot, val  $\lambda$  : real, i : nat) =
  [ [ * (  $\Delta 1/\lambda$ ; a!i ) ] ]
proc B(chan a?, b! : 2#lot, c? : lot, e! : 2#nat, val  $x^\#$  : 2 * (2 * real),
  x(0) : 2 * (2 * nat),  $m^A(0)$ ,  $m^B(0)$  : nat) =
  [ [ var xs : 2 * (2 * [lot]) = << [ ], [ ] >, < [ ], [ ] >, x : lot, m : 2 * nat = <  $m^A(0)$ ,  $m^B(0)$  >
  :: ( ; , i  $\leftarrow$  0..1, ( ; , j  $\leftarrow$  0..1, (x(0).i.j > 0) * > (xs.i.j := xs.i.j ++ [j]; x(0).i.j := x(0).i.j - 1)))
  ; * ( ( [ ], i  $\leftarrow$  0..1, a.i?x; xs.0.i := xs.0.i ++ [x])
    [ c?x; xs.1.x := xs.1.x ++ [x]
    [ ( [ ], i  $\leftarrow$  0..1, ( [ ], j  $\leftarrow$  0..1, m.i = j  $\wedge$  len(xs.i.j) > 0  $\rightarrow$  b.i!hd(xs.i.j); xs.i.j := tl(xs.i.j)))
    [ ( [ ], i  $\leftarrow$  0..1, len(xs.0.i) = 0  $\wedge$  len(xs.1.i) = 0  $\wedge$  m.0 = i  $\wedge$  m.1 = i
       $\wedge$  len(xs.0.(1 - i))  $\geq$   $x^\#.0.(1 - i) \rightarrow$  e.1!(1 - i); m := <i, 1 - i>)
    [ ( [ ], i  $\leftarrow$  0..1, len(xs.0.i) = 0  $\wedge$  m.0 = i  $\wedge$  m.1 = 1 - i  $\wedge$  len(xs.1.i)  $\geq$   $x^\#.1.i$ 
       $\rightarrow$  e.0!(1 - i); m := <1 - i, 1 - i>)
    )
  ] ]
proc M(chan a?, b! : lot, c? : nat, val  $\mu$ ,  $\sigma$  : 2 * real, m(0) : nat, x0 : real) =
  [ [ var x : lot, m : nat = m(0)
  ::  $\Delta x_0$ 
  ; * ( a?x;  $\Delta 1/\mu.m$ ; b!x
    [ c?m;  $\Delta \sigma.m$ 
    )
  ] ]
proc E(chan a? : lot) =
  [ [ var x : lot
  :: *a?x
  ] ]
model S(val  $\lambda_1, \lambda_2, \mu_1^A, \mu_2^A, \mu_1^B, \mu_2^B, \sigma_{21}^A, \sigma_{12}^A, \sigma_{21}^B, \sigma_{12}^B, x_1^{A\#}, x_2^{A\#}$  : real,
 $x_1^A(0), x_2^A(0), x_1^B(0), x_2^B(0), m^A(0), m^B(0)$  : nat,  $x_0^A(0), x_0^B(0)$  : real) =
  [ [ chan a, b : 2#lot, c, d : lot, e : 2#nat
  :: G(a.0,  $\lambda_1$ , 0)
  [ G(a.1,  $\lambda_2$ , 1)
  [ B(a, b, c, e, < $x_1^{A\#}, x_2^{A\#}$ >, << $x_1^A(0), x_2^A(0)$ >, < $x_1^B(0), x_2^B(0)$ >>,  $m^A(0), m^B(0)$ )
  [ M(b.0, c, e.0, < $\mu_1^A, \mu_2^A$ >, < $\sigma_{21}^A, \sigma_{12}^A$ >,  $m^A(0), x_0^A(0)$ )
  [ M(b.1, d, e.1, < $\mu_1^B, \mu_2^B$ >, < $\sigma_{21}^B, \sigma_{12}^B$ >,  $m^B(0), x_0^B(0)$ )
  [ E(d)
  ] ] ]

```

(B.4)

Nederlandse samenvatting

De complexiteit van zowel producten als productieprocessen is de afgelopen jaren enorm toegenomen. Dit heeft geresulteerd in high-tech productiesystemen: het maken van ingewikkelde en dure producten met nog duurdere en ingewikkeldere productiemiddelen. Fouten in de productie worden hierdoor erg duur en dienen vermeden te worden om de doelstellingen van de fabrikant te verwezenlijken: het maken van producten met een zo groot mogelijke winst. Over het algemeen worden deze doelen bereikt door de bedrijfsprocessen, de machines, de voorraden en het personeel goed onder controle te houden. Dit is makkelijker gezegd dan gedaan. Immers, als het al mogelijk is controle te houden over alle processen in een fabriek is dat op zichzelf al erg duur. Door delen van het fabricageproces te modelleren kan de complexiteit hiervan teruggedrongen worden. Met behulp van het model kan toekomstig gedrag van het systeem voorspeld worden. Bovendien kunnen met het model ingewikkelde regelstrategieën vooraf getest worden, met een laag risico, zonder dat ze op het daadwerkelijke productiesysteem geïmplementeerd hoeven te worden.

In dit proefschrift worden fabricagelijnen gemodelleerd met behulp van verschillende modelleertechnieken. Deze technieken worden onderverdeeld in drie groepen: discrete gebeurtenis (*discrete event*) modellen, continue modellen en hybride modellen. De opsomming van modelvormen is geenszins bedoeld als totaaloverzicht van alle beschikbare modelvormen. De behandelde modelvormen worden in dit proefschrift gebruikt.

Een toestandsrepresentatie voor een werkstation wordt geïntroduceerd in dit proefschrift. Deze toestand heeft een eindige dimensie, kan instantaan gemeten worden en bevat bovendien geen enkele informatie over de gebruikte productie- of regelstrategie. Deze toestandsrepresentatie wordt gebruikt om verschillende modelvormen aan elkaar te koppelen, waarmee analysetechnieken van zowel het tijddomein als het eventdomein gebruikt kunnen worden. Daarnaast wordt de toestandsrepresentatie gebruikt voor de ontwikkeling van een terugkoppelingsregelaar (*feedback controller*) die continu in de tijd optimale productieplannen geeft, op basis van

een voortschrijdende horizon. Deze regelaar is ontwikkeld voor een productielijn met een willekeurig aantal werkstations, elk met een eigen capaciteit, en voor een willekeurige horizonsafstand. De regelaar levert optimale productieplanningen, ook wanneer er onverwachts verstoringen optreden.

Schakelende werkstations (*switching servers*) zijn er in vele vormen, zoals productiesystemen, verkeersnetwerken en call-centers. Schakelende werkstations verwerken meerdere typen producten (goederen, auto's, telefoontjes) en moeten omstellen tussen het bewerken van verschillende typen. Dit omstellen kost tijd. Een hybride vloeistof model (*fluid model*) wordt gebruikt om de dynamica van zo'n workstation te beschrijven, waarbij het continue deel van de dynamica het verloop van de bufferniveaus in de tijd beschrijft, terwijl het discrete deel het omstellen tussen producttypen beschrijft. Optimaal omstelbeleid met betrekking tot gemiddeld niveau van onderhanden werk wordt vastgesteld voor een workstation dat twee typen producten verwerkt die met een constante snelheid aankomen bij het workstation. Dit omstelbeleid is bepaald voor werkstations met zowel onbegrensde buffercapaciteiten als begrensde buffercapaciteiten. Een belangrijk verkregen inzicht is het mogelijk optreden van een langzaam-aan-modus (*slow-mode*) in de gevonden optimale productiecyclus. Tijdens een langzaam-aan-modus is de buffer van een bepaald type leeg en worden binnenkomende producten meteen verwerkt, in plaats van om te stellen naar een ander type. Deze langzaam-aan-modus kan worden gezien als een afweging tussen enerzijds capaciteit verliezen door langzamer te produceren dan de topsnelheid, en anderzijds capaciteit te verliezen door relatief vaak om te stellen tussen producttypen. Conditie op het optreden van een dergelijke langzaam-aan-modus worden afgeleid.

In een netwerk van werkstations is de aankomstsnelheid van producten over het algemeen niet constant. Voor een schakelend workstation dat twee typen producten verwerkt, die aankomen met een stuksgewijs constante snelheid (aan/uit), wordt in dit proefschrift een optimale schakelstrategie bepaald. Het optimalisatieprobleem wordt dan opgesplitst in meerdere deelproblemen, die afzonderlijk opgelost dienen te worden.

De optimale hoeveelheid onderhanden werk voor een enkel schakelend workstation is tevens een ondergrens voor de hoeveelheid onderhanden werk in een fabricagelijns waarvan dat specifieke workstation deel uitmaakt. Voor een fabricagelijns met twee schakelende werkstations die elk twee typen producten verwerken, worden de voorwaarden afgeleid waaronder dit optimale niveau van onderhanden werk bereikt kan worden. Een belangrijke conclusie is dat in veel gevallen in een fabricagelijns een synchronisatiemechanisme tussen werkstations onderling nodig is om het gewenste totaalgedrag van de fabricagelijns te bereiken.

Voor zowel een enkel schakelend workstation als voor fabricagelijns van schakelende werkstations met constante productaanvoer worden toestandsterugkoppelingsregelaars voorgesteld die vanaf ieder toegestaan beginpunt de trajectorie van het systeem naar de gewenste trajectorie brengen. In tegenstelling tot veel gebruikte regelmethode in de literatuur wordt in dit onderzoek eerst gewenst (mogelijk optimaal) systeemgedrag gedefinieerd, waarna een regelaar wordt ontworpen die dit gewenste gedrag kan bereiken.

De vraag rijst of altijd gezocht moet worden naar optimaal systeemgedrag. Het onderzoek in dit proefschrift toont aan dat momenteel alleen voor een kleine klasse van werkstations en fabricagelijnen optimaal gedrag en bijbehorend omstelbeleid bepaald kunnen worden. Meer dan twee producttypen of meer dan twee werkstations in een fabricagelijns leiden tot ingewikkelde optimalisatieproblemen. Los van deze complexiteit is het bovendien onbekend of optimaal periodiek gedrag bestaat voor deze grotere systemen. Ondernemers zijn vaak niet geïnteresseerd in de theoretisch optimale oplossing. Vaak zal een oplossing die 'beter is dan de huidige' volstaan. Bovendien zal een ondernemer een suboptimale oplossing verkiezen boven een optimale oplossing, als de suboptimale minder gevoelig is voor verstoringen of onzekerheden.

Dit proefschrift kan als uitgangspunt dienen voor vervolgonderzoek op het gebied van modelleren en regelen van fabricagenetwerken. De geïntroduceerde toestandsrepresentatie kan uitgebreid worden voor fabricagenetwerken en andere soorten machines dan hier behandeld. Bovendien kunnen andere onderzoeksgebieden gekoppeld worden aan dit onderzoek, zoals stochastische analyse en effectieve procestijden.

(Van enkele termen de Engelse vertaling toegevoegd, omdat de Engelstalige termen volledig ingeburgerd zijn in het Nederlandse vakjargon.)

Curriculum vitae

Joost van Eekelen was born on February 16th, 1979 in 's-Hertogenbosch, The Netherlands. In 1996, he finished 'Gymnasium Beekvliet' in Sint-Michielsgestel and started the Mechanical Engineering educational program at the Eindhoven University of Technology (TU/e). Parallel to his education, he was involved in several extra-curricular activities. He attended and organized conferences and joined excursions, took part in a study tour to China, and was a member of the Education Committee and Department Council of the Department of Mechanical Engineering at the TU/e.

He received the master's degree in 2003 after writing his master's thesis on the development of continuous approximation models of manufacturing systems. This research was continued in a Ph.D. project on modelling and control of manufacturing systems in the Systems Engineering group of professor Rooda at the TU/e. Results of the project have been written in this thesis and have been presented at several conferences.

During the Ph.D. project, Joost was involved in a number of educational activities, such as coaching freshmen, lecturing foreign pre-master students and coaching students in their bachelor or master projects.

Joost van Eekelen is currently employed as Simulation Engineer with Vanderlande Industries.