Event-driven Model Predictive Control of a manufacturing line

G.J. Samson

SE 420431

Master's thesis

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

Eindhoven University of Technology
Department of Mechanical Engineering
Systems Engineering Group

Eindhoven, May 2005

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

February 2004

Student	G.J. Samson
Supervisor	Prof.dr.ir. J.E. Rooda
Advisor	Dr.ir. A.A.J. Lefeber
Start	February 2004
Finish	February 2005
Title	Event-driven Model Predictive Control of a reentrant line

Subject

Recent research on manufacturing control of reentrant semiconductor manufacturing lines consists of a hierarchical approach for inventory control and production optimization. At the top layer parameters of an aggregated model are obtained on-line. At the intermediate layer, production optimization and inventory control are performed using model predictive control. At the bottom layer the aggregated targets issued by the MPC-controller are translated into discrete-event decisions.
In the current approaches, time is discretized. Once every pre-determined period, based on the current data, the optimal control action is determined. Events occurring during a sample period are only responded at after the sample period has passed.

Assignment

Investigate the possibilities for using a hierarchical approach for inventory control and production optimization of a reentrant semiconductor manufacturing line using an event driven sample scheme.
For that purpose, study the work of G. Weiss on optimal control of linear systems and implement this optimal controller using MATLAB. Apply this optimal controller to a case study using an event driven sample scheme for the MPC-controller.
Also, consider the influence of the bottom control layer, i.e. the conversion of the continuous control signals generated by the MPC controller into discrete-event decisions.
Compare the performance of the proposed control scheme and describe under what conditions the new approach is valid. Present the results in a report, and provide recommendations for further research.

Prof.dr.ir. J.E. Rooda

Dr.ir. A.A.J. Lefeber

Preface

In front of you lies my Master Thesis, this is the final project of the Mechanical Engineering studies at the Technische Universiteit Eindhoven. This thesis also brings an end to a period of eight years, during which I experienced the life of a student. In these years I have made many friends and I learned a lot of things both related to study activities as well as non-study activities (such as the department board, my board year with the student association W.S.V. Simon Stevin).

The path that leads to this final project started with the basic mechanical engineering course and in my fourth year I made the choice of applying for the group "Systems Engineering" of Prof.dr.ir. J.E. Rooda. From there my path lead to an internship at University of Florida with Prof.dr. P.M. Pardalos. After this wonderful period in Florida and a small internship, my Master Thesis followed. Both were coached by dr.ir. Erjen Lefeber. My master thesis was an assignment that tested my mathematical knowledge thoroughly and at some moment made me desperate, but at the same time it challenged me.

Before I start with new challenges that lie ahead, I would like to take the opportunity to thank a few people in my life. First of all I would like to thank my parents for all the opportunities they gave me and for their love and support, my sister for being my big sister, and my girlfriend Annerie for the love and support she gave me in the last few years. I would also like to thank my friends for making my student years, a period that I will never forget.

A special thanks goes to dr.ir. Erjen Lefeber for coaching me and teaching me how to perform research during my internship and during my master thesis, Prof. G. Weiss for his help with the algorithm and for supplying us a solver of his algorithm, Prof.dr.ir. J.E. Rooda for supervising my internships and my master thesis, and I would like to thank all the other SE-staff members for helping me when I needed help. And finally I would like to thank the boys from the SE-lab for creating a pleasant atmosphere to work in.

I would like to end with a sentence in my native language:

"A jari bing tof, ma now-now mie kba"
"The year was difficult, but now I'm finished"

Gerald Samson
Eindhoven, May 2005

Summary

Nowadays we demand more luxury and more comfort. A mobile phone is not small enough or does not have enough functions. We want everything in one machine. This results in more complex systems and an even more complex manufacturing system. To make sure that the manufacturing line produces at the maximum of its capacity, it needs to be controlled or managed. A method to control a manufacturing line is the hierarchical approach for inventory control and production optimization. This method uses a Model Predictive Control (MPC) controller and two conversions to interact with the manufacturing system. The controller uses a dynamical model to predict the effects of the future behavior of a system and calculates the future control actions for that system, by solving an optimization problem for a certain prediction horizon. The control action are then implemented, and the process is repeated after a fixed interval. However, it is possible that all the control actions are put into effect before the end of the period. In such a case it would be desirable to calculate the new control actions after this event had occurred. This leads to the objective of this project.

The objective of this project is to investigate the possibility of using an event-driven sample scheme in a hierarchical approach for inventory control and production optimization of a manufacturing line. In addition, the influence of the conversion of the continuous MPC into discrete-event decisions is to be investigated.

To use an event-driven sample scheme it is important that a controller is able to calculate the control action at any arbitrary point in time, which means that the optimization problem has to be continuous along the time line. This creates a Continuous Linear Programming optimization problem.

To solve this problem a new algorithm has been used. This algorithm, called a Separated Continuous Linear Programming (SCLP) algorithm is, has been developed by Weiss [Wei04]. The algorithm solves the problem by using techniques from the Linear Programming (LP) theory. The algorithm performs the following actions. It first starts with finding a feasible solution at $t = 0$ or $t = T$, this is done by solving an LP problem (Boundary-LP/LP*). This LP problem is derived from the main problem, by neglecting the control actions. In the second step the algorithm determines in which direction the optimal solution moves, by using the properties that control variables are piecewise constant and the state variables are continuously piecewise linear. The directions are determined by solving a new LP problem (Rates-LP/LP*), the problem is a function of the control action and the derivative of the states. With the result of rates the optimal base-sequences, the set of variables in an optimal solution, is found, when the direction in which the solution moves is known. The duration

of this optimal solution can be determined, this duration is called a validity region. At the end of this interval a new optimal solution needs to be found. The algorithm does this by preforming a pivot step, similar as in LP theory. This pivot step causes a change of the lower and upper boundary of the Rates-LP/LP* and, by solving this problem, again a new optimal solution is found. The next step is to determine whether the new sequence is adjacent, i.e. the solutions only differ by one variable, to the existing base-sequences. This is required to find all the optimal solutions for the desired time horizon. If the solutions are not adjacent, a subproblem is required to find the sequence(s) that connect all the base-sequences. By repeating the following step until the desired time horizon has been reached, the algorithm is able to find all the optimal solutions. These steps are: determining the Validity Region of the new solution, pivoting at the end of the interval, changing the lower and upper boundary of the rates, solving the Rates-LP/LP*, creating, and solving a subproblem when needed.

With the ability to solve the optimization problem of the controller it is possible to calculate the targets for the manufacturing line. Except that there is one more problem. The controller creates a continuous signal and the manufacturing line uses a discrete signal as input. In order to interact between the two, the signals have to be converted. The first conversion is the conversion between the manufacturing line and the controller. This is achieved by measuring the buffer level and by determining the fraction of time a lot has been processed on that machine. This fraction is added to the buffer after the machine and the remaining part of that fraction is added to the buffer in front of the machine. The second conversion is the conversion from the controller to the manufacturing line. This is performed by calculating the target of each machine. The targets are calculated by taking the optimal throughputs of the controller and multiplying them with the duration that the throughputs are optimal. This results into a set of targets that have to be reached in the specified time. And at the machine level a sequencing policy determines which lot should be processed first to reach the given target.

The next step is to create a simulation model of a manufacturing line. The manufacturing line that has been considered is GBMBME line (a line with two finite buffers and two machines). With this model four types of sampling schemes are tested. The first scheme, used as a reference, is using as a fixed sample period of 12 hours. The second sample scheme, samples after a product finishes on the machine on which it is processed. The third sample scheme, samples after a product leave the line as finished product. The final sampling scheme, samples after a machine breakdown. The sample schemes are simulated with two types of process times. The first simulation uses a fixed process time. The second simulation uses a stochastic process time.

The results show that for both types of simulation and all types of sample schemes, the controller is able to reach a steady state after a few hours. The number of hours before the steady state is reached, depends on the initial buffer levels and the type of sample scheme. The results also show that both the high and low frequency of (re)optimizations, end in a steady state. However, with a sample scheme that uses a high frequency of (re)optimization, the Discrete Event Model (DEM) follows the control signal better.

The final conclusion of this project is that it is possible to use an event-driven sample scheme, when using a hierarchical approach for inventory control and production optimization of a

manufacturing line. But there are still a few questions that need to be answered. Due to time-limitations, it was not possible to answer all these questions. But they are now formulated as recommendation for further research. The first recommendation is to program the algorithm for a better understanding of the algorithm, the second recommendation is to investigate what the limitations of the algorithm are and to find the best way is to use this algorithm in an MPC setting. The third recommendation is about investigating the conversion from the manufacturing system/DEM to the controller to find the best way is to perform this conversion. The final recommendations are about the simulations, it is recommended to simulate using a larger system and to make a simulation with more complex model manufacturing lines, for example a reentrant line. It is also advisable to improve the error registration.

Samenvatting

In de hedendaagse wereld wil de mens steeds meer luxe en gemak. Een mobiele telefoon kan niet klein genoeg zijn en de functies die de telefoon moet hebben lijken wel haast oneindig. Eigenlijk zouden we het liefst alles in een apparaat willen hebben. Dit brengt met zich mee dat de apparaten veel complexer worden en het fabricage systeem om zulke apparaten te maken is zelfs nog een graad complexer. Om ondanks deze complexiteit het toch mogelijk te maken dat deze fabricage lijn op een maximale bezettingsgraad kan opereren is het nodig dat zo'n fabricage lijn wordt bestuurd of geregeld. Eén van deze methodes om een lijn te besturen is de hiërarchisch methode voor voorraad beheersing en productie optimalisatie. Deze methode maakt gebruikt van een Model Predictive Control (MPC) regelaar. Deze regelaar gebruikt een dynamisch model om te voorspellen wat de effecten in de toekomst zullen zijn op het gedrag van het systeem. En zo berekent het welke regel-acties in de toekomst nodig zijn. Deze berekening wordt gedaan door gebruik te maken van een optimalisatie probleem met een bepaalde voorspellings horizon. Deze regelactie wordt vervolgens geïmplementeerd en het proces herhaalt zich na een bepaalde periode. Het is echter denkbaar dat alle regelacties zijn uitgevoerd, maar het einde van de periode nog niet in zicht is. In zo'n geval zou het wenselijk zijn om een nieuwe berekening te doen nadat zo'n gebeurtenis heeft plaatsgevonden. Dit leidt tot het formuleren van de doelstelling van dit onderzoek.

De doelstelling van dit onderzoek is het onderzoeken van de mogelijkheid om een test planning van een hiërarchisch methode voor voorraad beheersing en productie optimalisatie van fabricage lijn, die aangestuurd op basis van gebeurtenissen. De tweede doelstelling is het onderzoeken van de invloeden van de conversie van de continue MPC naar de discrete-event beslissingen.

Bij het gebruik van gebeurtenis aangedreven test planning is het belangrijk dat de regelaar in staat is, om een berekening te maken van de regel-actie op elk willekeurig moment. Dit betekend dat het optimalisatie probleem continu in de tijd moet zijn. Dit leidt tot een "Continuous Linear Programming" optimalisatieprobleem.

Om dit optimalisatieprobleem op te lossen is een nieuw algoritme nodig. Dit algoritme heet een "Separated Continuous Linear Programming" (SCLP) algoritme, ontwikkeld door Weiss [Wei04]. Het algoritme lost het probleem op door gebruik te maken van technieken die bekend zijn uit de Lineare Programming (LP) . Het SCLP algoritme voert de volgende handelingen uit om tot een oplossing te komen. Het zoekt eerst naar een geschikte oplossing voor het tijdstip $t = 0$ of $t = T$. De oplossingen worden gevonden door het oplossen van een LP probleem (Boundary-LP/LP*), dat afgeleid is van het hoofd probleem door de regel acties te verwaarlozen. De volgende stap is het bepalen in welke richting de optimale oplossing zal

bewegen. Dit wordt bepaald door gebruik te maken van de aanname dat de regel variabelen stuksgewijs constant zijn en de toestandsvariabelen stuksgewijs continu lineair zijn. De richting wordt bepaald door het oplossen van een nieuw LP probleem (Rates-LP/LP*), dat een functie is van de regel variabelen en de afgeleide van de toestandsvariabelen. De oplossing van het Rates probleem levert een set van optimale variabelen op, deze set wordt ook wel de base-sequences genoemd. De huidige optimale oplossing en de Rates maken het mogelijk om te bepalen hoelang deze optimale oplossing optimaal blijft. Dit wordt berekend met de validity region. Wanneer de oplossing niet meer optimaal is zal het algoritme een pivoteer stap uitvoeren om een nieuwe optimale oplossing te creëren. Deze pivoteer stap gebeurt op een soortgelijke manier als de pivoteer stappen in LP theorie. Met de gemaakte pivoteer stap is het mogelijk om de boven- en ondergrens van het Rates-LP/LP* te veranderen en op deze manier een nieuwe optimale oplossing te vinden. De volgende stap is om te kijken of de nieuwe oplossing aangrenzend is aan de reeds bestaande oplossingen. Indien dit niet het geval is zal er een sub-probleem worden geformuleerd en opgelost om alle oplossingen aan elkaar te verbinden. Als het probleem wel aangrenzend is kan er een nieuwe validity region bepaald worden en begint het verhaal weer van vooraf aan.

Nu dat het optimaliseringsproblemen is opgelost kan de regelaar een target voor de machines in de fabricagelijn vinden. Er ontstaat echter een probleem tussen de signalen van de regelaar en de fabricagelijn. De regelaar gebruikt namelijk continue signalen en de fabricagelijn heeft discrete signalen nodig. Dit betekent dat de signalen geconverteerd dienen te worden. De eerste conversie is de conversie van de fabricagelijn naar de regelaar. Hierbij wordt eerst het buffer niveau gemeten en wordt er bepaald welk deel van een lot reeds is gefabriceerd. Dit deel wordt opgeteld bij het buffer niveau van de buffer achter de machine en het resterende deel wordt bij de buffer voor de machine opgeteld. De tweede conversie gaat van de regelaar naar de fabricagelijn. Van deze conversie is bekend dat de regelaar de doorzet [lots/uur] voor de machine opgeeft. Verder is het bekend hoelang de doorzet geldig is (dit wordt uit de optimalisatie verkregen). De doorzet en de duur leveren samen een target op die een machine moet produceren binnen een bepaalde tijd. Wanneer er machines in de lijn zijn die voorzien worden door meerdere buffers, zal er op machine niveau nog een sorteer algoritme gebruikt worden om te bepalen welke machine/buffer het eerst moet produceren.

De volgende stap is het creëren van een simulatie model van een fabricagelijn om een test case te maken. De lijn die in beschouwing wordt genomen is een GBMBME lijn. Een lijn met twee buffers en twee machines, waarbij de buffers elk een machine bedienen. Om de gebeurtenis aangedreven test planning te testen zijn er vier planningen gedefinieerd. De eerste planning wordt gebruikt om een referentie kader te creëren. Hierbij wordt als planning gedefinieerd met een vaste tijd periode van 12 uren. De tweede planning is het moment dat er een product of deel product is gefabriceerd op een van de machines. De derde planning is gedefinieerd als de gebeurtenis dat een product de lijn verlaat. De laatste planning is wanneer een machine kapot gaat. Deze planning worden voor twee test casussen getest. Namelijk voor de situatie dat de bewerkingstijden van de machines vast zijn en de andere casussen is wanneer de machines stochastisch bewerkingstijden gebruiken.

De resultaten tonen aan dat bij beide casussen en voor alle type planningen, de regelaar in staat is het systeem na een paar uur naar een steady state te brengen. De tijdsduur die nodig is om het systeem in een steady state situatie te brengen, is afhankelijk van de begin

conditie van de buffer en de type planning. Tevens tonen de resultaten aan, dat zowel voor een hoge als een lage frequentie van (her)optimalisatie de regelaar instaat is om het systeem in een steady state toestand te brengen. Echter is het wel zo dat bij een hoge frequentie het Discrete Event Model (DEM) beter in staat is het regel signaal te volgen.

De eindconclusie van dit onderzoek is dan ook dat het mogelijk is om een hiërarchische methode voor voorraadbeheersing en productie optimalisatie te gebruiken om een fabricagelijn te regelen op basis van gebeurtenis gedreven herplanning.
Er zijn tijdens het onderzoek nieuwe vragen ontstaan. Hier was echter geen tijd meer voor om ze goed op te lossen, en daarom zijn het aanbevelingen geworden. De eerste aanbeveling is het programmeren van het algoritme om zo een beter begrip voor het algoritme te krijgen. De tweede aanbeveling is het onderzoeken van de beperkingen van het algoritme en om te onderzoeken wat de beste manier is om het algoritme te gebruiken in een MPC setting. De derde aanbeveling heeft betrekking op de conversie van de fabricagelijn naar de regelaar. Het is aan te raden om te onderzoeken wat de beste manier is om deze conversie uit te voeren. De laatste aanbevelingen hebben betrekking op de simulaties, het is aan raden om simulaties te doen met grotere systemen en een simulatie met complexere systemen zoals een reëntrant lijn. Tevens moet de fout registratie verbeterd worden.

Contents

Table of symbols and abbreviations

Symbols	***Description***
T_p	prediction horizon
T_c	control horizon
T	horizon
$\vec{r}$	cost factor on the control variables
$\vec{p}$	cost factor on the state variables
$\vec{x}(k+i\|k)$	the state variables of the sampled system
$\vec{y}(k+i\|k)$	the output variables of the sampled system
Δu	control effort
$\mathbf{A}$	system matrix
$\mathbf{B}$	input matrix
$\mathbf{C}$	output matrix
$\mathbf{D}$	direct transmission matrix
$\vec{\lambda}$	arrival rate
$\bar{\mathbf{B}}$	designates the position of the arrival rate(s)
E	output constraint matrix
F	input constraints matrix
$\mathbf{\Phi}$	system matrix for a sampled dynamical model
$\mathbf{\Gamma}$	input matrix
$\vec{\mathcal{Y}}$	vector containing the all buffer levels
$\vec{\mathcal{M}}$	vector containing the all machine capacities
$\vec{x}(t)$	state variable
$\vec{u}(t)$	control variable [lots/hours]
$\vec{q}(t)$	dual state variable
$\vec{p}(t)$	dual control variable
$\vec{y}(t)$	output variable [lots]
$\vec{c}$,$\vec{\gamma}$	cost factors on the control variables in the SCLP formulation
$\vec{d}$	cost factors on the state variables in the SCLP formulation
G	matrix in the SCLP formulation (4.1)
F	matrix in the SCLP formulation (4.1)
H	matrix in the SCLP formulation (4.1)
$\vec{\alpha}$	vector in the SCLP formulation (4.1)
$\vec{a}$	vector in the SCLP formulation (4.1)

Symbols	***Description***
$\vec{b}$	vector in the SCLP formulation (4.1)
$\vec{x}^0$	vector with the boundary values of the primal problem
$\vec{q}^N$	vector with the boundary values of the dual problem
τ	interval length of the validity region
σ	slack variable of the validity region
B'	first of the two base-sequence
B''	second of the two base-sequence
D	the new base-sequence
B	base-sequence of the primal problem
$B*$	base-sequence of the dual problem
$\mathcal{K}$	Set variable needed to determine the boundary of primal rates-LP
$\mathcal{J}$	Set variable needed to determine the boundary of dual rates-LP*
S_x	upstream buffer
S_y	downstream buffer
ΔS_x	Difference between Actual Outs and Estimate Outs of the upstream buffer
ΔS_y	Difference between Actual Outs and Estimate Outs of the downstream buffer
y_{max}	Maximum buffer level
SCLP	primal SCLP problem
SCLP*	dual SCLP problem

Abbreviations

MBPC	Model Based Predictive Control
MPC	Model Predictive Control
DEM	Discrete Event Model
LP	Linear Programming
CLP	Continuous Linear Programming
SCLP	Separated Continuous Linear Programming
AO	Actual Outs
EO	Estimate Outs
VPP	Variable Property Policy

Chapter 1

Introduction

The world is changing rapidly, everything has to be smaller, faster and cheaper. This means that the complexity of the products and their manufacturing systems is increase. However, to produce cheaper products, manufacturing systems must realize high throughputs. A control strategy used to manage the flow through a manufacturing system is one of the methods that can contribute to the optimization of high throughputs in complex manufacturing systems. In this research project the flow in the manufacturing system is presumed to be a discrete flow.

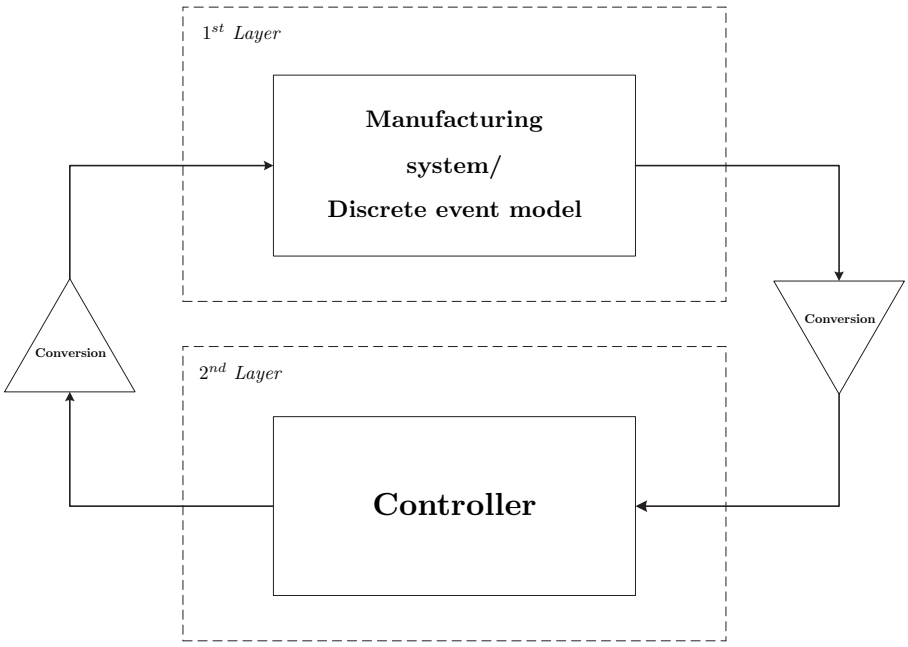


Figure 1.1: *Hierarchical control structure*

1.1 Framework

The framework in which the flows in a manufacturing system are controlled, is called a hierarchical approach (Figure 1.1). This approach consists of two layers interacting with each other. The first layer is the manufacturing system or a discrete-event model of the manufacturing system and the second layer is the controller. These two layers interact by communicating the measured parameters of the manufacturing system to the controller and by sending the calculated target from the controller to the manufacturing system. The problem with these communications is that the manufacturing system is a discrete system and the controller is a continuous system. This means that in order to communicate between the layers the signals have to be converted, see Figure 1.1.

1.2 Control strategy

By making a discrete-event model of a manufacturing system, the process of the manufacturing system can be simulated. To control the model of the manufacturing system a controller needs to be designed. In recent years several research groups have done research on designing controllers, that are able to control the flows of a manufacturing system.
One way to design a controller is to use theories and strategies that are common in control engineering. These theories and strategies are adapted and extended for the use in the field of manufacturing control.

This research project focuses on the theory of inventory control and production optimization. This theory uses a model to optimize the production and to control the system. The controller used in this project is a Model Predictive Controller (MPC). This controller uses a model of the system to predict the required trajectory to reach the desired set-point within a finite time horizon. By using the current output data of the Discrete-Event Model (DEM), the MPC can determine the optimal control actions. These control actions are implemented for the next interval and this process is repeated every sample period (k). In the current approaches, the MPC uses a fixed pre-determined sample period, this is also known as a discrete-time MPC (see Figure 1.2). This figure shows the samples k, the measured level x and the asterisk (*) are measure points.
A mayor disadvantage of current discrete-time MPC is the use of a fixed sample period. This means that the MPC always has to wait until the end of a fixed time step, even if all the control actions have already been carried out. The following research objective has been defined to cope with this problem.

1.3 Research objective

The objective of this project is to investigate the possibility of using an event-driven sample scheme in a hierarchical approach for inventory control and production optimization of a manufacturing line. In addition, the influence of the conversion of the continuous MPC into discrete-event decisions, on the controller is investigated.

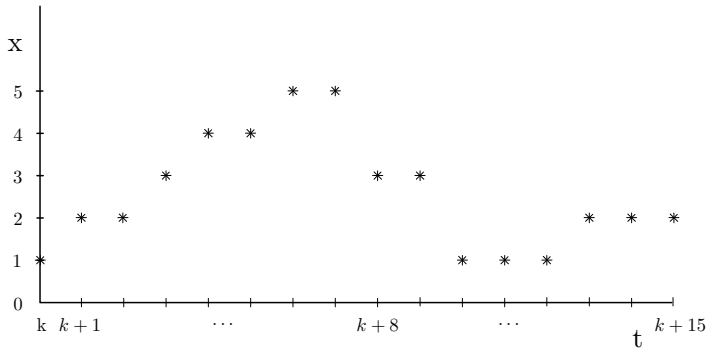


Figure 1.2: *Predefined sample scheme.*

1.4 Approach

In order to investigate the applicability of an event-driven sample scheme, a different approach is needed in comparison with a fixed time step based MPC. When using an event-driven sample scheme, the Model Predictive Controller needs to be able to determine the optimal control action at any arbitrary moment in time. This is achieved by using a continuous-time approach for the Model Predictive Controller(also known as continuous-time MPC). Consequently, one obtains a controller that is continuous along the time line, but the optimal control actions are discrete. This is shown in Figure 1.3, it also shows the measured values of the manufacturing system represented by asterisks (*).

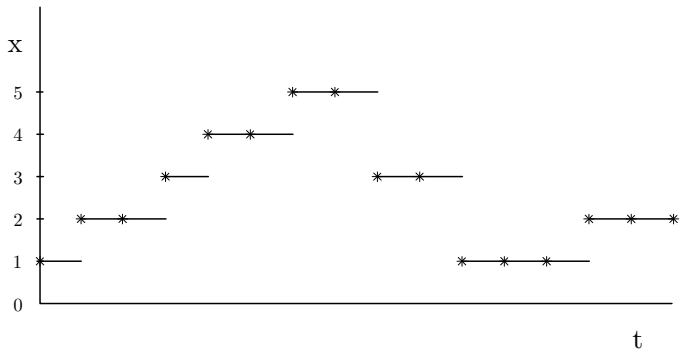


Figure 1.3: *Event-driven sample scheme*

For solving the optimization problem of a discrete-time MPC problem a Linear Programming (LP) algorithm can be used. Unfortunately an LP-algorithm cannot be used to solve a continuous-time MPC-problem directly. In order to solve such a problem a Continuous Linear Programming (CLP) algorithm is required. In 2004, Weiss [Wei04] initiated the development of a new algorithm to solve a CLP problem. With this algorithm he is able to find an exact solution for a subclass of the CLP problem, namely the Separated Continuous Linear Programming (SCLP). This algorithm finds an optimal solution by solving several LP problems and combines these results to a solution for the entire problem.

By using the algorithm presented by Weiss, it is possible to solve the continuous-time MPC-problem directly, which makes the implementation of an event-driven hierarchical structure possible. In order to implement and test the principle of an event-driven sample scheme, it is important to consider the influences of the conversions in the hierarchical structure as well. An important issue of these conversions to investigate the influences of the conversions on the controller.

1.5 Outline of the report

So far, an introduction into this research project is given and the objective and the approach have been defined. In the following chapters, the approach is elaborated. The first step in elaborating the approach is to give a short introduction into MPC, this is described in Chapter 2. The second phase is to formulate a continuous-time MPC problem, shown in Chapter 3. In order to solve the continuous-time MPC problem, it is important to understand how the SCLP algorithm solves the problem. In Chapter 4 the theory behind the SCLP algorithm is explained. To complete the hierarchical framework the conversion needs to be explained, this is presented in Chapter 5. The next phase in this report is Chapter 6 that explains the definition of the event-driven sample scheme and the results of the case study. The report ends with a conclusion and recommendations.

Chapter 2

Model Predictive Control (MPC)

The controller used in the hierarchical structure is a Model Predictive Controller. In this chapter a short introduction into the Model Predictive Control (MPC) theory is given. For a more extensive review of MPC-theory see the referred articles.

2.1 Model Predictive Control theory

The Model predictive control theory was first used in the late 1970s and early 1980s. In those days, the theory was known under several different names: Model Predictive Heuristic Control, Dynamic Matrix Control, Generalized Predictive Control or Model Algorithmic Control. Nowadays the generic names Model Predictive Control or Model-Based Predictive Control (MBPC) are widely used. An overview is presented in e.g. [Gar89, Cam99, Mac02].

MPC or MBPC are considered advanced control strategies, that use explicit (internal) dynamical models of the process to control the system. The model is used to predict the effects of future control actions on the output of the system by minimizing the predicted error subject to the operational restrictions.

In the industry MPC has a wide range of applications, especially in the process industry. In most of these applications, MPC is calculated off-line, because the calculations are time-consuming.

Many of the models used in the industry are linear models. However, these models are mostly inadequate for these systems, because the system is inherently nonlinear. In addition, the operational restrictions on the systems have increased in recent years. Nonlinear MPC models deals with these restrictions. For an overview on this topic see [All00].

2.2 The concept of MPC

As mentioned, MPC uses a dynamical model to predict the future behavior of the system. The predictions are made by solving an optimization problem. This problem consists of an objective function and constraints which represent the limitations of the system. The optimization problem computes the inputs for the entire prediction horizon, but only the values of the first sample are actually implemented. After measuring the effects, the horizon is shifted forward by one step and the procedure is repeated. This is called a moving or receding horizon.

In Figure 2.1 the basic principles of MPC are shown. One of these principles is the prediction (T_p) and control horizon (T_c). The control horizon represents the time during which the input can change. The prediction horizon is the time over which the system is evaluated. When the prediction horizon is longer than the control horizon, the input is assumed to be constant after the end of the control horizon.

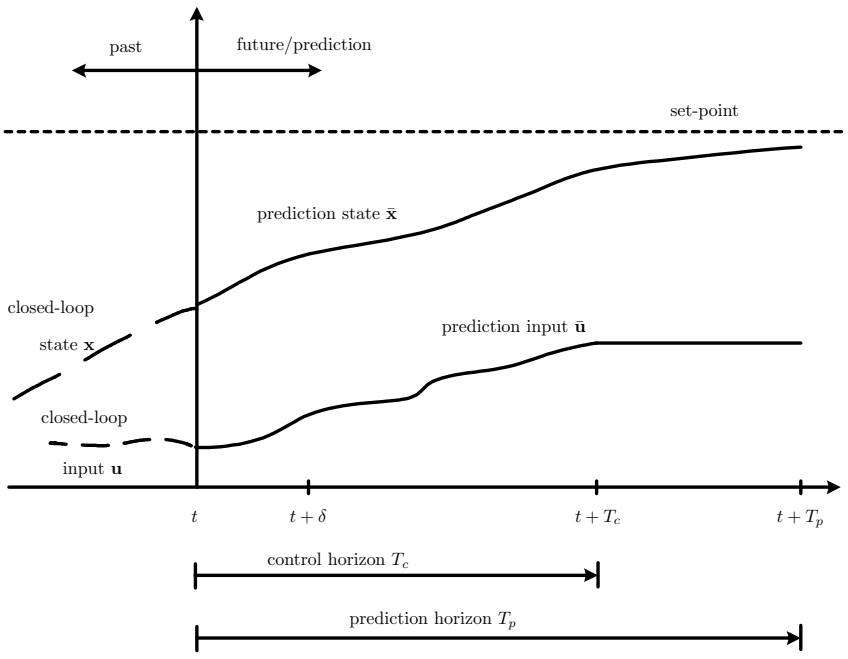


Figure 2.1: *Concept of Model Prediction Control.*

The dynamical model can be represented in several ways, e.g. by writing the dynamical model as a linear input-state-output model (2.1).
In this model $\vec{x}(t)$ denotes the state, $\vec{y}(t)$ denotes the output, $\vec{u}(t)$ denotes the input and $\vec{\lambda}$ is the arrival rate, $\mathbf{A}$ is the systems matrix, $\mathbf{B}$ is the input matrix, $\mathbf{C}$ is the output matrix, $\mathbf{D}$ is the direct transmission matrix, and $\bar{\mathbf{B}}$ designates the position of the arrival rate(s).

$$\begin{aligned} \dot{\vec{x}}(t) &= \mathbf{A}\ \vec{x}(t) + \mathbf{B}\ \vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda} \\ \vec{y}(t) &= \mathbf{C}\ \vec{x}(t) + \mathbf{D}\ \vec{u}(t). \end{aligned} \tag{2.1}$$

Now that the dynamical model is defined, the next step is to formulate a linear MPC problem. This can be done by formulating either a discrete-time or a continuous-time MPC problem.

For both categories the dynamical model presented in (2.1) forms the basis. In the case of the discrete-time MPC problem, the dynamical model is sampled and the sum of the costs of every sample is taken. In (2.2) an example is shown of a discrete-time MPC problem. In this example, a discrete cost function is shown with x as the state and u as the control variable. The states are summed along the prediction horizon, which are p samples and the control variables are summed along the control horizon, with m samples. Equation (2.2b) is the model constraints and Equation (2.2c) are the inequality constraints. The samples for the state and control variables are calculated for every sample $k+i$, and implemented at k.

$$\min_{\mathcal{U}} \quad J = \sum_{i=1}^{p} \vec{c}\, \vec{x}(k+i|k) + \sum_{j=0}^{m-1} \vec{d}\, \vec{u}(k+j|k)), \tag{2.2a}$$

s.t. model constraint

$$\begin{aligned} &\vec{x}(k+i|k) = \mathbf{\Phi}\ \vec{x}(k+i-1|k) + \mathbf{\Gamma}\ \vec{u}(k+j-1|k) + \bar{\mathbf{B}}\vec{\lambda}, \\ &\vec{y}(k+i|k) = \mathbf{C}\ \vec{x}(k+i-1|k) + \mathbf{D}\ \vec{u}(k+j-1|k), \\ &i = 1, \ldots, P \qquad j = 1, \ldots, m, \end{aligned} \tag{2.2b}$$

s.t. inequality constraints

$$\begin{aligned} &\mathbf{E}\vec{y}(k+i|k) \leq \vec{0}, \\ &\mathbf{F}\vec{u}(k+j|k) \leq \vec{0}, \\ &\vec{x}(k+i|k), \vec{u}(k+j|k) \geq \vec{0}, \\ &\vec{\mathcal{U}} = \begin{bmatrix} \vec{u}(k|k) & , \cdots & , \vec{u}(k+m-1|k) \end{bmatrix}^T. \end{aligned} \tag{2.2c}$$

Where $\vec{\mathbf{c}}$ are the costs on the states, $\vec{\mathbf{d}}$ are the costs on the inputs, $\vec{x}(k+i|k)$ are the state variables of the sampled system, $\vec{y}(k+i|k)$ are the output variables of the sampled system, $\vec{u}(k+j|k)$ are the input variables of the sampled system, $\mathbf{\Phi}$ is the system matrix of the sampled system, $\vec{\mathbf{\Gamma}}$ is the input matrix of the sampled system, $\mathbf{C}$ is the output matrix, $\mathbf{D}$ is the direct transmission matrix, $\bar{\mathbf{B}}$ is designates the position of the arrival rate(s), the matrix $\mathbf{E}$ is the output constraint matrix, and $\mathbf{F}$ is the input constraints matrix.

The continuous-time MPC problem is slightly different from discrete-time, i.e. it has a smooth control function. Moreover, the cost function uses an integral for summing the costs. Equation (2.3) shows a continuous-time example. This example has the same structure as (2.2), but instead of taking the sum of the horizon, the integral of the horizon is used.

$$\min_{t} \quad J = \int_{t}^{t+T_p} \vec{c}(\tau)\vec{x}(\tau) + \int_{t}^{t+T_c} \vec{d}(\tau)\vec{u}(\tau)d\tau, \tag{2.3a}$$

s.t. model constraint

$$\dot{\vec{x}}(t) = \mathbf{A}\ \vec{x}(t) + \mathbf{B}\ \vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda}, \tag{2.3b}$$

$$y(t) = \mathbf{C}\ \vec{x}(t) + \mathbf{D}\ \vec{u}(t),$$

s.t. inequality constraints

$$\begin{aligned} \mathbf{E}\vec{y}(\tau) &\leq \vec{0}, \\ \mathbf{F}\vec{u}(\tau) &\leq \vec{0}, \\ \vec{x}(\tau), \vec{u}(\tau) &\geq \vec{0}. \end{aligned} \tag{2.3c}$$

In linear MPC-theory, the inputs can be represented in two ways. The first ways is to represent the input as Δu, this is called the control effort ($\Delta u(k|k) = u(k|k) - u(k-1|k)$). In literature, this method is often used. The second way of representing the input is using absolute inputs (u).

In this project absolute inputs are used to represent the input values of the system, as can be seen in (2.2,2.3). This is done, because the use of absolute input makes it easier to formulate the prediction Matrix. It is also possible to weigh the inputs [Ess02] and, in addition, it is difficult to use Δu in the continuous-time formulation.

2.3 Stability and Robustness of an MPC-controller

In control theory, stability and robustness are very important [Ess02, All00, All01, Mac02]. The stability of an MPC controller can be divided into two categories, i.e. nominal and robust stability. In the case of nominal stability the model is assumed to describe the system perfectly, which means that the internal model of the MPC controller is identical to the model of the system. Robust stability refers to the case that the internal model is not equal to the model of the system.

It is difficult to provide nominal stability of an MPC-controlled system , because the controller uses a moving horizon and the nonlinearities of explicit constraints. One way of guaranteeing the stability in an open-loop optimal control problem is to consider an infinite horizon, with the condition that the problem is feasible at $t = 0$ [Fin00]. If this is valid, the closed-loop system is asymptotically stable for a small enough sampling time and the region of attraction contains the set of states for which the optimization problem of the open-loop problem has a solution [Fin00]. The use of an infinite horizon in practice is usually relatively difficult. Another way of ensuring the stability with a finite horizon is by adding a zero terminal constraint or to add an extra term to the cost function (quasi infinite horizon control)[Fin00].

For the robustness of an MPC-controller, it is important to take the uncertainties and disturbances into account. This can be achieved by using the nominal stability model, which only takes them into account indirectly. An alternative is to make a new model that takes the uncertainties and disturbances into account directly.

2.4 Advantages and Disadvantages of MPC

The MPC method has some advantages, when comparing MPC with other standard feedback control methods. First it can handle constraints. With the constraints it is possible to take physical limitations, safety and performance constraints of a system into account. It also has some disadvantages. For example, it needs an accurate model of the process. The most important disadvantage is the relatively high computational demand.

2.5 Discussion

Model Predictive Control is a theory that uses a dynamical model to predict the effects of the future behavior of a system and calculates the future control actions for that system, by solving an optimization problem for a certain prediction horizon.
MPC can be used in two ways, namely as a discrete-time problem or a continuous-time problem. With discrete-time MPC the dynamical model is sampled and the cost function is the minimum of the sum of all cost of each sample. The continuous-time problem uses a continuous dynamical model and an integral function of the cost is minimized.
The cost function of the discrete-time and continuous-time MPC are solved for a certain horizon. The length of this horizon is important to provide stability of the controlled system. To guarantee the stability of an open-loop optimal control problem, an infinite horizon with a feasible solution in $t = 0$ is desirable.
When the advantages and disadvantages are kept in mind a Model Predictive Controller can be an useful tool for controlling a manufacturing system.

As mentioned in Chapter 1 the hierarchical approach uses an continuous-time problem to control a manufacturing system. This chapter showed the formulation of continuous-time controller, a continuous-time MPC controller, and a small introduction into the MPC theory was given. The next step is to formulate a continuous-time MPC problem based on a manufacturing line.

Chapter 3

Formulating a continuous-time MPC problem

The previous chapter showed an introduction into the concept and the theory of MPC. The next phase is to formulate an MPC problem. In this research project a continuous-time MPC problem is formulated. This formulation can be divided into three elements. The first one is the definition of a dynamical model. The second step is to define a cost function and finally the physical constraints of the system have to be translated into constraints for the optimization problem.

3.1 Model

The model used for the continuous-time MPC problem is a representation of a manufacturing system, which has n machines and a specified routing. This model is derived by using a linear-input-state-output model as presented in (2.1). The model presented in this chapter is slightly different. The difference is caused by the assumption made to accurately model the manufacturing system. This results in the following model, with $\dot{x}$ the derivative of the state, $\mathbf{B}$ represents the routing of a lot, $\vec{u}(t)$ is the speed with which the lots are produced and $\vec{y}(t)$ is the buffer level:

$$\begin{aligned} \dot{\vec{x}}(t) &= \mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda}, \\ \vec{y}(t) &= \mathbf{I}\vec{x}(t). \end{aligned} \tag{3.1}$$

The difference with (2.1) is that the states ($\vec{x}(t)$) of the system have no influence on the derivative $\dot{\vec{x}}$, because in a manufacturing line the change of the buffer content can only be achieved by production. A second assumption is that $\vec{u}(t)$ has no direct influence on output $\vec{y}(t)$. The last assumption is that the buffer levels ($\vec{y}(t)$) are equal to the states ($\vec{x}(t)$) of the system ($\mathbf{C} = \mathbf{I}$).

3.2 The cost function and constraints

MPC uses an optimization function to find the desired control action. This means that a cost function and its constraints need to be formulated. Both functions are derived from the physical properties of a manufacturing system.

Cost function

When analyzing the production costs in a manufacturing system. The costs can be divided into two groups. The costs for having products or parts in stock (buffer level), these costs are increase with every step in manufacturing system (value is added to the product) ($\vec{p}$). The second set of costs are the costs of producing ($\vec{r}$), with the notion that producing products too early should cost more. This means that the production costs are weighted with the factor of time. This results in the following cost function:

$$\min_{u,x} J = \int_0^T (T-t)\vec{r}' \begin{pmatrix} u(t)_1 \\ \vdots \\ u(t)_n \end{pmatrix} + \vec{p}' \begin{pmatrix} x(t)_1 \\ \vdots \\ x(t)_n \end{pmatrix} d\tau. \tag{3.2}$$

Constraints

The final step is to translate the physical constraints of the system, into constraints for the optimization problem. The analysis of the system leads to the conclusion that the limitation of the buffer and machine capacities cause the constraints on the system.
The buffer constraint is determined by the fact that the machine cannot produce more than the content of the buffer plus the number of lots produced by the machine in front of it ($\vec{y} \geq 0$) in combinations with the buffer cannot be larger than the maximal buffer content. This results in the following constraint:

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \leq \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \leq \begin{pmatrix} y_{max_1} \\ \vdots \\ y_{max_n} \end{pmatrix} \qquad \text{(the buffer constraints)}, \tag{3.3}$$

$$\vec{0} \quad \leq \quad \vec{\mathcal{Y}} \quad \leq \quad \vec{\mathcal{Y}}_{max} \tag{3.4}$$

The second limitation is the capacity of the machine:

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \leq \mathbf{R}\vec{u}(t) \leq \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} \qquad \text{(the machine capacity)} \tag{3.5}$$
$$\vec{0} \quad \leq \mathbf{R}\vec{u}(t) \leq \vec{\mathcal{M}}$$

where matrix $\mathbf{R}$ contains the information which buffer(s) serves which machine and the process time of that machine and $\vec{\mathcal{M}}$ containing the machine capacities.

3.3 Resulting continuous-time MPC problem

By combining the Equations (3.2), (3.3) and (3.5) the following continuous-time MPC problem is formulated:

$$\min_{u,x} J = \int_0^T (T-t)\vec{r}'\ \vec{u}(t) + \vec{p}'\ \vec{x}(t) d\tau, \tag{3.6a}$$

s.t. model constraint

$$\dot{\vec{x}}(t) = \mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda}, \tag{3.6b}$$
$$\vec{y}(t) = \mathbf{I}\vec{x}(t),$$

s.t. inequality constraints

$$\vec{0} \leq \vec{\mathcal{Y}} \leq \vec{\mathcal{Y}}_{max} \tag{3.6c}$$
$$\vec{0} \leq \mathbf{R}\vec{u}(t) \leq \vec{\mathcal{M}}$$

3.4 Examples of continuous MPC problems

In the previous sections a continuous-time MPC problem was formulated. To create a better understanding, the formulation is illustrated with a simple manufacturing line with two buffers (B) and two machines (M). G is a generator and E is the exit process. This type of line is also called a GBMBME model, presented in Figure 3.1. A simple example of a reentrant line is shown in Figure 3.2. For this line the dynamical model has the following shape:

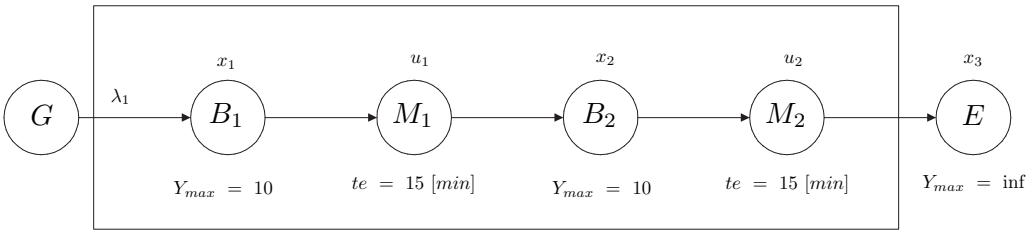


Figure 3.1: *GBMBME manufacturing line*

$$\begin{aligned} \dot{x}_1(t) &= -u_1(t) + \lambda_1 \\ \dot{x}_2(t) &= \ \ u_1(t) - u_2(t) \end{aligned} \tag{3.7}$$

$$\dot{\vec{x}}(t) = \begin{pmatrix} -1 & 0 \\ 1 & -1 \end{pmatrix} \vec{u}(t) + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \vec{\lambda}$$
$$\dot{\vec{x}}(t) = \mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda},$$

The exit process and generator are not taken into account for the dynamical model.

With the dynamical model the buffer constraints are formulated as:

$$\vec{y}(t) = \int_0^t (\mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda})dt \tag{3.8}$$
$$\vec{0} \leq \int_0^t (\mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda})dt \leq \vec{Y}_{max}$$

The capacity of the machines is formulated as follows:

$$0 \leq te_1 u_1(t) \leq 1 \tag{3.9}$$
$$0 \leq te_2 u_2(t) \leq 1$$

$$\vec{0} \leq \begin{pmatrix} te_1 & 0 \\ 0 & te_2 \end{pmatrix} \vec{u}(t) \leq \vec{1}$$
$$\vec{0} \leq \mathbf{R}\vec{u}(t) \leq \vec{\mathcal{M}}$$

For a reentrant line the dynamical model has the following shape:

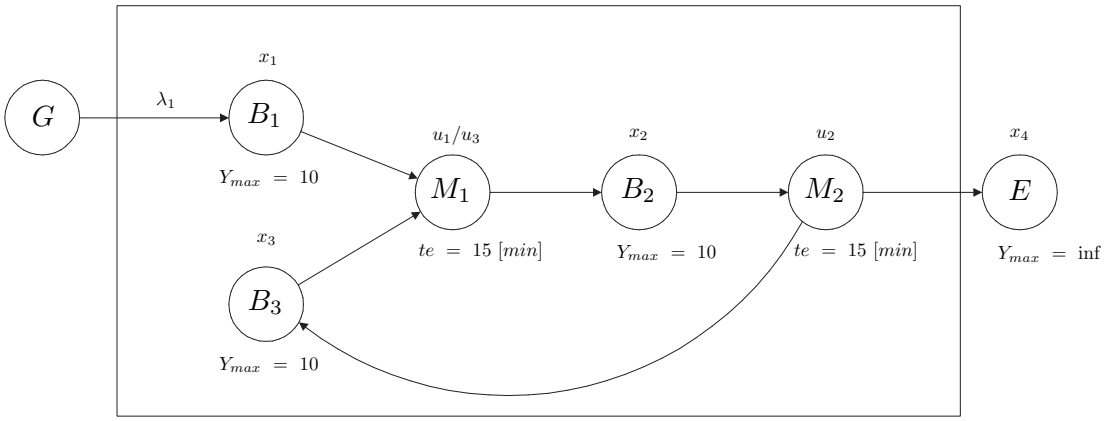


Figure 3.2: *Reentrant manufacturing line*

$$\dot{x}_1(t) = -u_1(t) + \lambda_1 \tag{3.10}$$
$$\dot{x}_2(t) = \ \ u_1(t) - u_2(t)$$
$$\dot{x}_3(t) = \ \ u_2(t) - u_3(t)$$

$$\dot{\vec{x}}(t) = \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \vec{u}(t) + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \vec{\lambda}$$
$$\dot{\vec{x}}(t) = \mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda},$$

The exit process and generator are not taken into account for the dynamical model.

The buffer constraints of the reentrant line is formulated in the same way as in (3.8).

The capacity of the machines are formulated as below:

$$\begin{aligned} &0 \leq te_1 u_1(t) + te_1 u_3(t) \leq 1 \\ &0 \leq te_2 u_2(t) \leq 1 \end{aligned} \tag{3.11}$$

$$\begin{aligned} &\vec{0} \leq \begin{pmatrix} te_1 & 0 & te_1 \\ 0 & te_2 & 0 \end{pmatrix} \vec{u}(t) \leq \vec{1} \\ &\vec{0} \leq \mathbf{R}\vec{u}(t) \leq \vec{\mathcal{M}} \end{aligned}$$

This concludes the simple examples of formulating a continuous-time MPC problem.

3.5 Discussion

This chapter formulated and illustrated a continuous-time MPC problem. The following elements were presented so far: a hierarchical structure, an explanation of the type of controller (MPC) used in the hierarchical structure was given and a small introduction to the MPC control theory was shown. And in this chapter a continuous-time MPC problem was formulated. The next step is to solve this problem. However, the problem at hand cannot be solved adequately with the standard optimization solvers. To do so a new algorithm is needed. This algorithm has been developed by Weiss [Wei04]. The next chapter explains this algorithm.

Chapter 4

Separated Continuous Linear Programming (SCLP)

In the previous chapters the MPC theory and the formulation of a continuous-time MPC problem have been discussed. The next step is to explain the algorithm which is capable to solve the formulated MPC problem, a Continuous Linear Programming problem. The algorithm used is called "a simplex based algorithm to solve a Separated Continuous Linear Program" [Wei04].

In this chapter the Separated Continuous Linear Program (SCLP) algorithm is explained by showing the equations and rules needed to solve an SCLP problem. The details of this algorithm are cited from [Wei04]. To create a better understanding of the algorithm some of the assumptions, theorems and definitions are shown. For the mathematical proof of the theory and a more detailed description of the algorithm the cited article should be read.

To help to understand the concept of the SCLP algorithm the reader is referred to Appendix B. This appendix shows a graphical interpretation of the algorithm.

4.1 The SCLP algorithm

The SCLP algorithm finds the solution to a subclass of the continuous linear programming problem, by using a theory which is analogous to linear programming and the Simplex algorithm (in parametric form). With these techniques Weiss was able to anchor the Continuous Linear Programming into LP theory. (To help to understand the background of this algorithm better, a short introduction to LP-theory is given in Appendix C.)

There are several ways of formulating an SCLP problem. The following formulation is used by Weiss:

$$\begin{aligned} \max \quad & \int_0^T ((\vec{\gamma}' + (T-t)\vec{c}')\vec{u}(t) + \vec{d}'\vec{x}(t))dt, \\ s.t. \quad & \int_0^t G\vec{u}(s)ds + F\vec{x}(t) \leq \vec{\alpha} + \vec{a}t, \\ & H\vec{u}(t) \leq \vec{b}, \\ & \vec{x}(t), \vec{u}(t) \geq 0, \qquad t \in [0,T]. \end{aligned} \tag{4.1}$$

In this equation G, H, F are fixed $K \times J$, $I \times J$, $K \times L$ matrices; the remaining variables are the K-vector $\vec{\alpha}, \vec{a}$, the I-vector $\vec{b}$, J-vector $\vec{\gamma}, \vec{c}$ and the L-vector $\vec{d}$. The unknown variable $u_j(t)$ and $x_k(t)$ are the control variables with $j = 1, \ldots, J$ and the state variables with $k = K+1, \ldots, K+L$. Both are a function of time $0 \leq t \leq T$ and they have additional control and state variables, $j = J+1, \ldots, J+I$ and $k = 1, \ldots, K$ which denotes the non-negative slack variables of respectively the second set of I and the first set of K inequality constraints.

To solve the SCLP problem, the algorithm uses duality theory to find an optimal solution (for an explanation of duality theory see Appendix C.2 or [Van96]). The duality theory creates a symmetric dual problem.
When the solution of both the primal and dual problem are feasible, the solution of the two problems are related by the complementary Slackness Condition, which states that:

$$u_j(t) > 0 \;\Rightarrow\; q_j(T-t) = 0, \; j = 1, \ldots, J+I \tag{4.2}$$

$$x_k(t) > 0 \;\Rightarrow\; p_k(T-t) = 0, \; j = 1, \ldots, J+I \tag{4.3}$$

The optimal solution of the problem is found, whenever the two solutions are optimal and have the same objective value. This is also called strong duality, there is no duality gap between the solutions.

The symmetric dual problem is formulated as follows:

$$\begin{aligned} \min \quad & \int_0^T ((\vec{\alpha}' + (T-t)\vec{a}')\vec{p}(t) + \vec{b}'\vec{q}(t))dt, \\ s.t. \quad & \int_0^t G'\vec{p}(s)ds + H'\vec{q}(t) \leq \gamma + \vec{c}t, \\ & F'\vec{p}(t) \leq \vec{d}, \\ & \vec{p}(t), \vec{q}(t) \geq 0, \qquad t \in [0,T] \end{aligned} \tag{4.4}$$

in which $p_k(t),\ k = 1, \ldots, K$ are the dual controls and $q_j(t),\ j = J+1, \ldots, J+I$ are the dual states, both are a function of time $0 \leq t \leq T$. The non-negative slacks of the inequality constraints of the dual states ($q_j(t),\ j = 1, \ldots, J$) and the dual controls ($p_k,\ k = K+1, \ldots, K+L$) are added in the same way as the primal problem.
In the remainder of this report the primal problem (4.1) is represented by SCLP and the dual problem (4.4) is represented by SCLP*.

With the primal and dual problem the SCLP algorithm is able to find the optimal solution for a range of $0 \leq t \leq T$. This is achieved by partitioning the time horizon into N intervals, $0 = t_0 < t_1, \cdots, t_N = T$. The partitioning of the horizon is allowed whenever Theorem 1 holds.

Theorem 1 ([Wei04], *Theorem 1*)**.** *Assume that the feasibility and boundedness assumption holds (see assumption 2 in 4.3). Then the SCLP/SCLP* problems (4.1,4.4) possess complementary slack optimal primal and dual solutions (strong duality holds), with continuous piecewise linear x, q, and piecewise constant u, p.*

As a result of Theorem 1 the SCLP algorithm can move from one interval to another, when it has a feasible initial solution. This is caused by the fact that the solutions are either piecewise continuous linear or piecewise constant. So with an initial solution the algorithm can move from one optimal solution to another by making pivot steps, similar to a pivot step in LP theory.

With the knowledge of Theorem 1, an algorithm can be formulated. This algorithm first searches for an initial solution with an end time of $t = 0$. This solution is optimal for a certain period of time. At the end of this time period the algorithm performs a pivot step to find a new optimal solution and the end time is increased. This is repeated until a solution is found which is optimal for the desired time horizon.

The steps of the algorithm are presented below and each step is explained in this chapter.

Algorithm 1 The SCLP algorithm

initialization:
- Solve the *boundary-LP/LP**,
- Solve the *Rates-LP/LP**,
- Determine the *base-sequences* B_1, B_N,
- Determine the *Validity region*,
- Perform a *Pivot* operation,
- Change the lower boundaries of the Rates-LP and Rates-LP*

end initialization

while $t < T$ **do**
- Solve the *Rates-LP/LP**
- Determine the new *base-sequences*,

if the base-sequences are not adjacent **then**
- *Formulate and solve the subproblem*(a smaller SCLP).

end if
- Determine the *Validity region*,
- Perform a *Pivot* operation,
- Change the lower boundaries of the Rates-LP and Rates-LP*

end while

4.2 Boundary-LP/LP*

The first step in the initialization of the SCLP-algorithm is to solve a Boundary-LP/LP*. This optimization problem calculates the starting values for the optimal solution of the SCLP/ SCLP* problem. These values are found by taking the primal problem at $t = 0$ and dual problem at $t = N$. (Note that the control actions at the boundary can be neglected for both the primary and the dual problem). This results in the boundary-LP/LP* problems (4.5,4.6). When these optimization problems are solved, the boundary values for $\vec{x}^0$, $\vec{q}^N$ can be obtained, if the values of α, γ are given.

Boundary-LP

$$\begin{aligned} \max \quad & \vec{d}'\vec{x}^{\,0}, \\ st. \quad & F\vec{x}^{\,0} \leq \vec{\alpha}, \\ & \vec{x}^{\,0} \geq \vec{0}, \end{aligned} \tag{4.5}$$

*Boundary-LP**

$$\begin{aligned} \min \quad & \vec{b}'\vec{q}^{\,N}, \\ st. \quad & H'\vec{q}^{\,N} \geq \vec{\gamma}, \\ & \vec{q}^{\,N} \geq \vec{0}. \end{aligned} \tag{4.6}$$

When the solution to the boundary-LP/LP* is found the optimal solution for the SCLP problem at $t = 0$ and $t = T$ is found. The next step is to find out in which direction the optimal solution changes. This is achieved by finding the solution to the Rates-LP/LP*.

4.3 Rates-LP/LP*

The next step in the Algorithm 4.1 is determining the Rates-LP/LP*. They are used in both the initialization and the mean part of the algorithm. Rates-LP/LP* are LP problems that uses the properties of Theorem 1, which states that the control variables $(\vec{u}(t), \vec{p}(t))$ are piecewise constant and that the state variables $(\vec{x}(t), \vec{q}(t))$ are continuously piecewise linear. So for each interval of the partitions, the rates $\vec{u}(t), \dot{\vec{x}}, p(t), \dot{\vec{q}}$ are piecewise constant. When the optimal value of the control variables and the derivative of the state variables are found, the optimal solution of the mean problem is also found automatically.

For each interval the rates problems are slightly different. The sign restrictions on the lower and upper boundaries of the rate problem cause the difference. In the initialization phase of the algorithm, the sign restrictions are determined by using the results of the boundary-LP/LP*. In the main part of the algorithm the sign restrictions are determined by taking the first sign restriction as the starting point and for each interval one or more sign restriction of the lower or upper boundaries of the rate problems are changed. The sign restrictions are defined as follows:

- "P" i.e. the variable is non-negative,
- "U" i.e. the variable is unrestricted,
- "Z" i.e. the variable is restricted to be 0.

For determining the sign restriction, the following set of variables are created:

$$\mathcal{K}_n \subseteq \{1,\ldots,K+L\},$$
$$\mathcal{J}_n \subseteq \{1,\ldots,J+I\}.$$

The sets $\mathcal{K}_n$ and $\mathcal{J}_n$ are determined by using the following equations:

$$\mathcal{K}_n = \{k : x_k^{n-1} > 0\}, \tag{4.7}$$
$$\mathcal{J}_n = \{j : q_j^n > 0\}, \qquad n = 1,\ldots,N. \tag{4.8}$$

This leads to the following rates-LP and rates-LP* problem:

rates-LP$(\mathcal{K}_n, \mathcal{J}_n)$

$$\begin{aligned} \max\ & \begin{bmatrix}\vec{c}\,' & 0\end{bmatrix}\vec{u} + \begin{bmatrix}0 & \vec{d}\,'\end{bmatrix}\dot{\vec{x}}, \\ \text{St.}\ & \begin{bmatrix}G & 0\end{bmatrix}\vec{u} + \begin{bmatrix}I & F\end{bmatrix}\dot{\vec{x}} = \vec{a}, \\ & \begin{bmatrix}H & I\end{bmatrix}\vec{u} = \vec{b}, \\ & \dot{x}_k \text{ is "U" for } k \in \mathcal{K}, \quad \dot{x}_k \text{ is "P" for } k \notin \mathcal{K}, \quad k = 1,\ldots,K+L, \\ & u_j \text{ is "Z" for } j \in \mathcal{J}, \quad u_j \text{ is "P" for } j \notin \mathcal{J}, \quad j = 1,\ldots,J+I. \end{aligned} \tag{4.9}$$

rates-LP*$(\mathcal{K}_n, \mathcal{J}_n)$

$$\begin{aligned} \min\ & \begin{bmatrix}\vec{a}\,' & 0\end{bmatrix}\vec{p} + \begin{bmatrix}0 & \vec{b}\,'\end{bmatrix}\dot{\vec{q}}, \\ \text{St.}\ & \begin{bmatrix}G' & 0\end{bmatrix}\vec{p} + \begin{bmatrix}-I & H'\end{bmatrix}\dot{\vec{q}} = \vec{c} \\ & \begin{bmatrix}F' & -I\end{bmatrix}\vec{p} = \vec{d}, \\ & \dot{q}_j \text{ is "U" for } j \in \mathcal{J}, \quad \dot{q}_j \text{ is "P" for } j \notin \mathcal{J}, \quad j = 1,\ldots,J+I, \\ & p_k \text{ is "Z" for } j \in \mathcal{K}, \quad u_j \text{ is "P" for } j \notin \mathcal{K}, \quad k = 1,\ldots,K+L. \end{aligned} \tag{4.10}$$

The first set lower boundary sign restrictions, is obtained by using the boundary values from the boundary-LP/LP*. The remaining sets are found by applying (4.7) and (4.8) on the solution of the rates-LP/LP* and after that the sign are changed due to the pivot steps.

To proof that the solutions of the boundary-LP/LP* and the rates-LP/LP* are part of the solution of the SCLP, SCLP* the following assumptions are made.

Assumption 2 ([Wei04], *Assumption 1*)**.** ***(Feasibility and Boundedness)***

case(i) The boundary-LP/LP problems (4.5,4.6) have a solution* $\vec{x}^{\,0}, \vec{q}^{\,N}$*.*
Denote $\mathcal{K}_1 = \{k : x_k^0 > 0\}$,$\mathcal{J}_N = \{j : q_j^N > 0\}$.

case(ii) The primal rate-LP($\emptyset, \mathcal{J}_N$) and the dual rate-LP($\mathcal{K}_1, \emptyset$) are both feasible.*

This assumption states that the problem has a solution at its boundaries and that the problem moves from its boundary solutions to a neighboring solution. The second assumption is made to ensure the algorithm can find an optimal solution.

Assumption 3 ([Wei04], *Assumption 2*)**.** ***(Non-degeneracy)***

The column $\begin{bmatrix}\vec{a}\\ \vec{b}\end{bmatrix}$ *is in general position to the matrix* $\begin{bmatrix}G & F & I & 0\\ H & 0 & 0 & I\end{bmatrix}$ *(it is not a linear combination of any less than* $K+I$ *columns), and the column* $\begin{bmatrix}\vec{c}\\ \vec{d}\end{bmatrix}$ *is in general position to the matrix* $\begin{bmatrix}G' & H' & -I & 0\\ F' & 0 & 0 & -I\end{bmatrix}$.

Theorem 4 ([Wei04], *Theorem 2*)**.** *Assumptions 3 implies that SCLP, SCLP* (4.1,4.4) are non-degenerate in the sense that optimal* $\vec{u}(t), \vec{x}(t), \vec{p}(t), \vec{q}(t)$ *are uniquely determined for all* $0 < t < T$*, up to an arbitrary set of values of* t *with Lesbesgue measure* 0*.*

After the Boundary-LP/LP* and the Rates-LP/LP* are solved the SCLP-algorithm has the first optimal solution of the SCLP problem and it has the direction in which the solution changes. This results in a set of variables which are called the Base sequence.

4.4 Base sequences

The solutions of the rates-LP/LP*($\mathcal{K}_n, \mathcal{J}_n$) create two sets of variables that are optimal base-sequences of the SCLP problem for that time interval, B_n for the primal problem and B_n^* for the dual problem. The base-sequence B denotes the set of the corresponding variables $\dot{x}_k, u_j \in B$ or the set of their indices. The complementary dual basis B^* is the set of the corresponding variables $\dot{q}_j, p_k \in B^*$ or the set of their indices. The sets are related by the complementary dual theory, which states that the indices in B, B^* partition $k \in \{1, \ldots, K+L\}, j \in \{1, \ldots, J+I\}$, i.e. $\dot{x}_k \in B \Rightarrow p_k \notin B^*$, $\dot{q}_j \in B^* \Rightarrow u_j \notin B$.

The bases B_n $(n = 2, \ldots, N-1)$ are admissible if $u_j^n, p_k^n, j = 1, \ldots, J+I, k = 1, \ldots, K+L$ are non-negative. The bases B_1, B_N are consistent with the boundary data $\vec{\alpha}, \vec{\gamma}$ if $x_k^0 > 0 \Rightarrow \dot{x}_k \in B$, $q_j^N > 0 \Rightarrow u_j \notin B_N$ and $B_1 \supseteq \mathcal{K}_1, B_N^* \supseteq \mathcal{J}_N$. The base-sequence B_1, found by solving rates-LP($\mathcal{K}_1, \mathcal{J}_N$) is optimal for a neighborhood $t > 0$. The base-sequence B_N is optimal for a neighborhood $t < T$.

Two bases B_i, B_j are adjacent if $B_i \setminus B_j$ consists of a single variable v_n and $B_j \setminus B_i = w_n$. This makes it possible to go from B_i to B_j in a single LP simplex pivot, in which v_n leaves the basis and w_n enters.

When the SCLP-algorithm has a Base-sequence and the Rates-LP/LP*, it now needs to determine how long this solution is optimal for the given solution. This is done by calculating the validity regions.

4.5 Validity regions

The validity region is necessary for the SCLP algorithm to determine how long an optimal solution remains optimal. To calculate a validity region it is important to keep in mind that the algorithm divides the time horizon into N intervals (4.11), for which a base-sequence is optimal. These intervals are the validity regions that have to be calculated. The length of a validity region interval is denoted by τ_n. The value of τ_n is unknown and it can change when a new base-sequence is found. To determine this length, Equations (4.11-4.13) are needed. These equations determine how long the optimal base-sequence remains valid.

$$\tau_1 + \cdots + \tau_N = T, \tag{4.11}$$

With the information found by the boundary-LP/LP* and the rates-LP/LP*, the validity region can determine how long it would take to hit or reach zero, when starting at x^0 or q^N and moving with a rate $\dot{x}$ or $\dot{q}$.

By moving from one optimal solution to another the algorithm calculates the rates for each optimal solution as it expands its time horizon to find all the optimal solutions for the desired time horizon. This means that a validity region can change when a new set of base-sequences are found (N increases). To calculate the length of the validity regions two sets of equations are needed ((4.12) and (4.13)). The first set of equations calculates the validity regions that have reached zero in the previous steps. These equations are calculated for ranges of $n = 1, \ldots, N-1$, but only for those states, which have left the base-sequence in that interval.

$$\begin{aligned} &\sum_{m=1}^{n} \dot{x}_k^m \tau_m = -x_k^0, && \text{when } v_n = \dot{x}_k \\ &\sum_{m=n+1}^{N} \dot{q}_j^m \tau_m = -q_j^N, && \text{when } v_n = u_j \end{aligned} \tag{4.12}$$

The second set of equations checks the regions where the rates go from negative in one region to a positive rate in the next region (and vice versa for the dual problem) or when the rates are negative and the region is at the boundary ($n = N$ or $n = 0$) the equations are formulated for $n = 1, \ldots, N$ for primal problem and $n = 0, \ldots, N-1$ for the dual problem.

$$\begin{aligned} &\sum_{m=1}^{n} \dot{x}_k^m \tau_m \geq -x_k^0, && \text{whenever } \dot{x}_k^n < 0, \text{ and either } \dot{x}_k^{n+1} > 0 \text{ or } n = N \\ &\sum_{m=n+1}^{N} \dot{q}_j^m \tau_m \geq -q_j^0, && \text{whenever } \dot{q}_k^{n+1} < 0, \text{ and either } \dot{q}_k^n > 0 \text{ or } n = 0. \end{aligned} \tag{4.13}$$

Equations (4.11-4.13) can be combined as:

$$\begin{pmatrix} 1 \ldots 1 & 0 \\ \mathbf{A} & 0 \\ \mathbf{B} & -I \end{pmatrix} \begin{pmatrix} \vec{\tau} \\ \vec{\sigma} \end{pmatrix} = \begin{pmatrix} T \\ g \\ h \end{pmatrix}. \tag{4.14}$$

Where equation (4.14) the appropriate values of $\dot{x}_k^n, \dot{q}_j^n$ are represented in the coefficients of $\mathbf{A}, \mathbf{B}$ and the values $-x_k^0, -q_j^N$ are represented in g, h. The (unknown) slack is represented in σ.

With the definitions of a base-sequence and the validity regions in mind the following theorems and definitions can be stated.

Theorem 5 ([Wei04], *Theorem 3*)**.**
Consider the SCLP, SCLP problems (4.1,4.4), and assume assumption 3. Let x_0, q_N be the solution of (4.5,4.6). Let $B_1, \ldots, B_N$ be a sequence of bases, which satisfy:*
(i) The bases $B_1, \ldots, B_N$ are admissible and adjacent.
(ii) B_1, B_N are consistent with the boundary values x_0, q_N.
(iii) The solution of the linear equations (4.14) is $\tau > 0, \sigma > 0$.

Definition 6 ([Wei04], *Definition 2*)**.**
Two base-sequences whose validity regions have intersecting boundaries are called neighboring base-sequences.

Theorem 7 ([Wei04], *Theorem 5*)**.**
Let $l(\theta) = (T(\theta), x^0(\theta), q^N(\theta)) = (T, x^0, q^N) + \theta(\delta T, \delta x^0, \delta q^N)$ be a line of boundary values. As θ changes, within the validity region of a single base-sequence, each of the interval lengths and slacks τ_n, σ_m, and each of the $x_k(t_n), q_j(T(\theta) - t_n)$ are affine functions of θ.

Theorem 8 ([Wei04], *Theorem 6*)**.**
Assume the non-degeneracy assumption 3. Let $\mathcal{C}$ be the region of boundary values T, x^0, q^N for which the feasibility and boundedness assumption 2 holds. Then $\mathcal{C}$ is tiled by closures of validity regions of optimal base-sequences.

Corollary 9 ([Wei04], *Corollary 10*)**.**
Assume the non-degeneracy assumption 3. Let $l(\theta) = (1-\theta)(T_1, x_1^0, q_1^N) + \theta(T_2, x_2^0, q_2^N)$ be a line of boundary values, for all of which assumption 2 holds. Then there exists a partition $0 = \theta^{(0)} < \theta^{(1)} < \ldots < \theta^{(R)} = 1$ and neighboring base-sequences $B_1^{(r)}, \ldots, B_{Nr}^{(r)}$, $r = 1, \ldots, R$ such that the rth base-sequence is optimal for $\theta^{(r-1)} < \theta < \theta^{(r)}$.

When Theorem 8 and Corollary 9 are taken into account the SCLP algorithm can be seen as a parametric algorithm. This algorithm starts from an optimal solution at T_1, x_1^0, q_1^N and constructs the optimal base-sequence in the intervals of the partition $0 = \theta^{(0)} < \theta^{(1)} < \ldots < \theta^{(R)} = 1$, to reach the solution at $\theta(T_2, x_2^0, q_2^N)$. The solutions is reached by going from base-sequence $B_1^{(r)}, \ldots, B_{Nr}^{(r)}$ to its neighboring base-sequences $B_1^{(r+1)}, \ldots, B_{Nr+1}^{(r+1)}$ by pivoting.

Equation (4.15), presents the update equation for going along the line of boundary values. The length of the validity region is represent by $\bar{\Delta}$. The end of this line is reached when $\bar{\Delta} \to \infty$. At that point, the current base-sequence is optimal for all $T > T^R$ This solution

corresponds to the base-sequence B_∞ which is the optimal basis for the rates-LP$(\emptyset, \emptyset)$.

$$\begin{aligned} &r := r + 1, \\ &\theta^{(r)} := \theta^{(r)} + \bar{\Delta}, \\ &T(\theta^{(r)}) := T(\theta^{(r)}) + \bar{\Delta}\delta T, \\ &x^0(\theta^{(r)}) := x^0(\theta^{(r)}) + \bar{\Delta}\delta x^0, \\ &q^N(\theta^{(r)}) := q^N(\theta^{(r)}) + \bar{\Delta}\delta q^N. \end{aligned} \tag{4.15}$$

Reduction of the number of intervals of importance

The validity regions are determined by checking all the intervals for situations in which $\dot{x}$ or $\dot{q}$ is non-positive. But at every breakpoint t_n, exactly one event happens. When $v_n = \dot{x}_k$, then x_k reaches zero and when $v_n = u_j$, then q_j hits zero at t_n. In Equation (4.14), all the intervals that could reach zero are included, it can be reduced by only taking those intervals that reaches zero at that breakpoint.

Definition 10 ([Wei04], *Definition 3*)**.**
It is said that x_k hits zero at t_n if $x_k(t) > 0$ for $t < t_n$ in a left neighborhood of t_n, and $x_k(t_n) = 0$. We say that q_j hits zero at t_n if $q_j(T - t) > 0$ for $t > t_n$ in a right neighborhood of t_n, and $q_j(T - t_n) = 0$.

This can be done by modifying the equation (4.12,4.13). For any k, n define:

$$n'(k, n) = \max\{m : 0 < m \leq n, \dot{x}_k \notin B_m \text{ if such exists, or } 0\}$$

and for any j, n define:

$$n''(j, n) = \min\{m : n \leq m < N, u_j \in B_{m+1} \text{ if such exists, or } N\}$$

Replace (4.12) by:

$$\begin{aligned} \sum_{m=n'(k,n)+1}^{n} \dot{x}_k^m \tau_m &= \begin{cases} -x_k^0 & \text{if } n'(k,n) = 0 \\ 0 & \text{if } n'(k,n) > 0 \end{cases}, \quad \text{when } v_n = \dot{x}_k, \\ \sum_{m=n+1}^{n''(j,n)} \dot{q}_j^m \tau_m &= \begin{cases} -q_j^N & \text{if } n''(j,n) = 0 \\ 0 & \text{if } n''(j,n) < N \end{cases}, \quad \text{when } v_n = u_j. \end{aligned} \tag{4.16}$$

Equations (4.16) show that if x_k hits zero at t_n, the corresponding equation sums the increments of x_k over the period $t_{n'(k,n)} < t < t_n$ (or $0 < t < t_n$). In this period $x_k > 0$, while $x_k = 0$ in the neighborhood before this period. In the same way when q_j hits zero at reversed time t_n, the corresponding equation sums the increments of q_j over the period $t_n < t < t_{n''(j,n)}$ (or $t_n < t < T$). In this period, $q_j > 0$, while $q_j = 0$ in the neighborhood after this period.

Replace (4.13) by:

$$\sum_{m=n'(k,n)+1}^{n} \dot{x}_k^m \tau_m = \begin{cases} -x_k^0 & \text{if } n'(k,n) = 0 \\ 0 & \text{if } n'(k,n) > 0 \end{cases}, \qquad \forall\ \dot{x}_k^n < 0,\ \dot{x}_k^{n+1} > 0, \text{ or } n = N, \quad (4.17)$$

$$\sum_{m=n+1}^{n"(j,n)} \dot{q}_j^m \tau_m = \begin{cases} -q_j^N & \text{if } n"(j,n) = 0 \\ 0 & \text{if } n"(j,n) < N \end{cases}, \quad \forall\ \dot{q}_j^N > 0,\ n = 0, \text{ and } \dot{q}_j^{n+1} < 0.$$

Equations (4.17) can be interpreted in a similar way as (4.16)
The reduced Equations (4.11,4.16,4.17) can be combined as a function of τ, σ:

$$\begin{pmatrix} 1 \dots 1 & 0 \\ \mathbf{A}_{red} & 0 \\ \mathbf{B}_{red} & -I \end{pmatrix} \begin{pmatrix} \tau \\ \sigma \end{pmatrix} = \begin{pmatrix} T(\theta^{(r)}) \\ g_{red}(\theta^{(r)}) \\ h_{red}(\theta^{(r)}) \end{pmatrix} + \Delta \begin{pmatrix} \delta T \\ \delta g_{red} \\ \delta h_{red} \end{pmatrix}. \quad (4.18)$$

Definition 11 ([Wei04], *Definition 4*)**.**
As $l(\theta)$ approaches the boundary of the validity region, it is said that a collision occurs at time t_n (for some $0 \le n \le N$), if at least one of the following happens: The interval $\tau_n = t_n - t_{n-1}$, or the interval $\tau_{n+1} = t_{n+1} - t_n$ or $\sigma_m = x_k(t_n)$ or $\sigma_{m'} = q_j(T - t_n)$ shrink to zero. It also say that a collision occurs at 0 (at T) if some components of x^0 (of q^N) reach 0, or if T reaches 0.

So far the algorithm has determined its starting point and it has calculated the direction in which the solution moves. It has also determined how long the first solution is valid, by calculating the validity region. The next step is to find the optimal solution for the next interval. The validity region has determined which variable violates the constraints. By making a pivot step, as used in LP-theory, the algorithm is able to change the boundary set of the Rates-LP/LP* (the sign restrictions). This results in a new set rates and a new base-sequence is found. And the length of the new solutions can be determined by calculating the validity regions and the algorithm is back at its next pivot step. So the next phase of this report is to explain what a pivot step is and how this step is performed.

4.6 Pivoting

As mentioned at the end of a validity region the algorithm has to perform an action to create a new optimal base-sequence, a pivot step. The pivot operations allows the algorithm to change from one neighboring validity region to another. It moves along a line of boundary values $l(\theta)$ and pivots from one valid optimal base-sequence for $\theta < 0$ to another valid optimal base-sequence for $\theta > 0$, where $\theta = 0$ is on the common boundary. At the common boundary a collision occurs, this can be a single or multiple collision. These collisions are defined as follows:

Definition 12 ([Wei04], *Definition 5*)**.**
A point on the boundary of the validity region is called a single collision point if the solution at this point consists of $\tilde{N}$ intervals and exactly $\tilde{N}$ state variables (primal or dual) hit 0 at these breakpoints. Equivalently, $\tilde{A}_{red}$ is $\tilde{N} \times \tilde{N}$. It is also called a single collision if exactly one component of x_0, q_N shrinks to zero. Otherwise it is said there is a multiple collision.

The single collision can be divided into two categories, a simple or a compound single collision. Definition 13 explains these type of single collisions.

Definition 13 ([Wei04], *Definition 6*)**.**
A single collision in which a single interval shrinks to zero is a simple single collision. A single collision in which more than one interval shrinks to zero is a compound single collision.

The final assumption the algorithm makes is:

Assumption 14 ([Wei04], *Assumption 3*)**. *(Single Collision)***

All the boundary points of validity regions along the line $l(\theta)$ are single collisions.

With the definition of pivoting given, it is time to classify the pivot steps. The algorithm uses several pivot operations, these operations can be divided into four groups. And each group can also be divided into sub groups. The sub groups are defined by the moment in time the collision occurs. It can occur at different stages, between $0 < t < T$, at $t = 0$, or at $t = T$. The difference between the main groups, is that either an interval shrinks to zero or a slack variable hits zero. A fourth case exits which is a special one. This case is needed in case a subproblem is defined (the explanation of subproblem will follow). In next part of this report neighboring bases are described by $'$ as the first base of the two bases and $''$ as the second base.

The collisions can be classified according to the following cases:

i) At collision time, $0 < t < T$ intervals between bases B', B'' shrink to $0, B'$ and B'' are adjacent, and exactly one of the x_k, q_j has a strict local minimum of 0 at t.
   a) At collision time $t = 0$, the intervals preceding the basis B'' shrink to 0 and exactly one of the q_j has a strict local minimum of 0 at $t = 0$.
   b) At collision time $t = T$ the intervals following the basis B' shrink to 0 and exactly one of the x_k has a strict local minimum of 0 at $t = T$.

ii) At collision time $0 < t < T$ intervals between bases B', B'' shrink to 0, and $B' \setminus B'' = \{v', v''\}$ where v' appears to leave the basis before v'' when $\theta < 0$ (i.e. $R_{B',B'',v',v''}(\theta) < 1$ for $\theta < 0$, see Definition 16).

iii) At collision time $0 < t < T$, a slack value σ_m hits 0. The variable leaving the basis in the pivot $B' \longrightarrow B''$ is v. If the slack that hits 0 is $\sigma_m = x_k(t)$, choose $v' = v$ and $v'' = \dot{x}_k$. If the slack that hits 0 is $\sigma_m = q_j(T - t)$, choose $v'' = u_j$ and $v'' = v$.
   a) At collision time $t = 0$, a slack value $\sigma_m = q_j(T)$ hits 0. In that case choose $v' = u_j$.
   b) At collision time $t = T$, a slack value $\sigma_m = x_k(T)$ hits 0. In that case choose $v'' = \dot{x}_k$.

iv) a) At time $t = 0,\ x_k^0 = 0$ and $\dot{x}_k \notin B_1$, and with an increase of θ it is intended that x_k^0 increases. In that case let $B'' = B_{1''}$ and $v' = \dot{x}_k$.
b) At time $t = T,\ q_j^N = 0$ and $u_j \in B_N$, and with an increase of θ it is intended that q_j^N increases. In that case let $B' = B_N$ and $v'' = u_j$.

When i, i_a, i_b or ii are applicable, delete those bases whose intervals have shrunk to zero from the base-sequence.

Formulation and solution of the LP problem

After the pivot step has been performed a new LP problem needs to be solved. This means that the set $\mathcal{K}$ and $\mathcal{J}$ has to be changed and with them the bounds of the LP problems. The pivot cases cause the following changes in sets:

In cases ii, iii, iii_b and iv_b let $\mathcal{K} = \{k : \dot{x}_k \in B' \setminus v''\}$.
In cases iii_a and iv_a let $\mathcal{K} = \{k : \dot{x}_k^0 > 0 \text{ or } \dot{x}_k = v'\}$.
In cases ii, iii, iii_a and iv_a let $\mathcal{J} = \{j : u_j \in \bar{B''} \setminus v'\}$.
In cases iii_b and iv_b let $\mathcal{J} = \{j : u_j q_j^N > 0 \text{ or } u_j = v''\}$.

These sets are used to solve the rates-LP/LP*$(\mathcal{K}, \mathcal{J})$. The optimal results of the rates are D.

The ratio across the pivot

In the classification of the pivot operations a single collision of Case ii (simple or compound) at t_n is described. During this collision between the bases B', B'', two variables are leaving the base-sequence. Let $B' \setminus B'' = \{v', v''\}$, with B' the optimal basis of the rates-LP$(\mathcal{K}_{B'}, \mathcal{J}_{B'})$ and B'' the optimal basis for the rates-LP$(\mathcal{K}_{B''}, \mathcal{J}_{B''})$, where:

$$\begin{aligned}
&\mathcal{K}_{B'} \text{ is the set of indices of } k \text{ for which } \dot{x}_k \in B',\\
&\mathcal{J}_{B''} \text{ is the set of indices of } j \text{ for which } u_j \notin B'',\\
&\mathcal{K}_{B''} = \mathcal{K}_{B'} \setminus \{\text{indices of } \dot{x}\text{'s in } v', v''\},\\
&\mathcal{J}_{B'} \ = \mathcal{J}_{B''} \setminus \{\text{indices of } \dot{x}\text{'s in } v', v''\},
\end{aligned}$$

We can now define:

$$\begin{aligned}
\mathcal{K}_C &= \mathcal{K}_{B'} \setminus \{k : v' \ = \dot{x}_k\},\\
\mathcal{J}_C &= \mathcal{K}_{B''} \setminus \{k : v'' = u_j\},\\
\mathcal{K}_D &= \mathcal{K}_{B'} \setminus \{k : v'' = \dot{x}_k\},\\
\mathcal{J}_D &= \mathcal{K}_{B''} \setminus \{k : v' \ = u_j\}.
\end{aligned} \tag{4.19}$$

Here, the rates-LP$(\mathcal{K}_C, \mathcal{J}_C)$ and the rates-LP$(\mathcal{K}_D, \mathcal{J}_D)$ lead to the optimal bases C respectively D. If the collision is simple, the bases C and D are both adjacent to B' and to B''. When the collision is not simple, but compound, a subproblem has to be solved. This could be the case for one side or for both sides (C, D). When both collisions are compound, it is not possible to show that $C \in (C_1, \ldots, C_{M'}), D \notin (C_1, \ldots, C_{M'})$ or $D \in (D_1, \ldots, D_M), C \notin (D_1, \ldots, D_M)$. During a pivot operation the bases $C_1, \ldots, C_{M'}$ are erased and $D_1, \ldots, D_M$ are inserted. It is important that the algorithm can make a distinction between these two sequences. To do that, the following ratio is created.

For the collision at $0 < t < T$, the ratio R is defined in the neighborhood of $\theta = 0$:

$$R_{B',B'',v',v''}(\theta) = \begin{cases} \dfrac{\left(\frac{x_{\prime}(t')}{-\dot{x}_{\prime}^{B'}}\right)}{\left(\frac{x_{\prime\prime}(t')}{-\dot{x}_{\prime\prime}^{B'}}\right)} & v' = \dot{x}_{\prime}, v'' = \dot{x}_{\prime\prime}, \\ \dfrac{\left(\frac{q_{\prime\prime}\left(T(\theta)-t''\right)}{-\dot{q}_{\prime\prime}^{B''*}}\right)}{\left(\frac{q_{\prime}\left(T(\theta)-t''\right)}{-\dot{q}_{\prime\prime}^{B''*}}\right)} & v' = u_{\prime}, v'' = u_{\prime\prime}, \\ \dfrac{\left(\frac{x_{\prime}(t')}{-\dot{x}_{\prime}^{B'}} + \frac{q_{\prime\prime}\ \left(T(\theta)-t''\right)}{-\dot{q}_{\prime}^{B''*}}\right)}{(t''-t')} & v' = \dot{x}_{\prime}, v'' = u_{\prime\prime}, \\ \dfrac{(t''-t')}{\left(\frac{x_{\prime\prime}(t')}{-\dot{x}_{\prime\prime}^{B'}} + \frac{q_{\prime}\left(T(\theta)-t''\right)}{-\dot{q}_{\prime}^{B''*}}\right)} & v' = u_{\prime}, v'' = \dot{x}_{\prime\prime}. \end{cases} \tag{4.20}$$

The superscript (B) denotes the value of the rate in the basic solution B.

Proposition 15 ([Wei04], *Proposition 5*)**.**
(a) The ratio $R_{B',B'',v',v''}(\theta)$ is strictly monotone in θ.
(b) At the collision one has $R_{B',B'',v',v''}(0) = 1$.
(c) In the validity region of base sequence $B_1, \ldots, B', C, B'', \ldots, B_N$ where v' leaves the basis before v'', i.e. $v' = B' \backslash C v'' = C \backslash B''$, one has $R_{B',B'',v',v''}(0) < 1$
(d) In the validity region of base sequence $B_1, \ldots, B', C, B'', \ldots, B_N$ with $\sigma_m = x_{\prime}$ and $v' = \dot{x}_{\prime}$ or $\sigma_m = q_{\prime\prime}$ and $v'' = u_{\prime\prime}$, one has $R_{B',B'',v',v''}(0) > 1$.
(e) The ratio $R > 1$ for every point in one of the validity regions, and it is smaller than 1 for every point in the other validity region.

Definition 16 ([Wei04], *Definition 7*)**.**
It is said that v' appears to leave the basis before v'' if $R_{B',B'',v',v''}(0) < 1$, and v' appears to leave the basis after v'' if $R_{B',B'',v',v''}(0) > 1$.

The main elements of the algorithm are now explained. The algorithm is able to move from first optimal solution to the next optimal solution, with the condition that the base-sequences are adjacent (they only differ by one variable). If the solutions are not adjacent the algorithm needs to create a subproblem, to find the solution between these two base-sequences.

4.7 Subproblem

When the rates-LP is calculated, an optimal base-sequence D is found. This sequence is checked whether or not it is adjacent to its neighbor(s). The base-sequences D can be adjacent to B' and B'', (to B'' in Cases $\mathrm{iii}_a, \mathrm{iv}_a$, to B' in Cases $\mathrm{iii}_b, \mathrm{iv}_b$,) and the neighboring base can be reached by making a simple pivot, a new base sequence is found. But if the

base-sequence D is not adjacent to one or to both bases B', B'' a subproblem is needed to find the sequence(s) that connect all the base-sequences.

A subproblem is actually nothing more then a smaller SCLP problem, with boundary values $l(\theta)$, $0 < \theta < 1$ and at $\theta = 0$ the optimal base-sequence is D, at $\theta = 1$ the optimal base-sequences B', B'' (B'' if $t = 0$, B'' if $t = T$). The solution to the subproblem is a sequence of adjacent admissible bases $B', D_1, \ldots, D_M, B''$ ($D_1, \ldots, D_M, B''$ in cases $\text{iii}_a, \text{iv}_a$, $B', D_1, \ldots, D_M$ in cases $\text{iii}_b, \text{iv}_b$), in which v'' appears to leave the basis before v'.

Formulation and the solution of the subproblem

In this section the formulation and solution of a subproblem is described as in [Wei04]. The description mainly focuses on the collision cases at $0 < t < T$, and the instruction are placed for the simpler cases of $t = 0$ and $t = T$ is placed between brackets.

When the SCLP algorithm calls upon one of the pivots, in collision cases ii,iii,iii_a, iii_b, iv_a,iv_b and the new bases are not neighboring, a subproblem is called. The data for the subproblem consists of the bases B', B'', the basis D and the rates v', v''. With $v'' \in B' \setminus D$, $v' \in D \setminus B''$, and D is not adjacent to one or both of B', B''. The state variables are z', z'' and they correspond to v', v''. (At $t = 0$ the data is B'', D, Z', v', with D not adjacent to B''; At $t = T$ the data is B', D, Z'', v'', with D not adjacent to B').

Creating the subproblem

The subproblem is a reduced version of the mean problem. To create the subproblem the variables in both sequences are excluded. These variables are not needed to solve the subproblem, because they are already part of the known solution. This results in:

- Exclude $x_k, \dot{x}_k, p_k$ from the problem, for all $\dot{x}_k \in B' \cap B''$ ($\dot{x}_k \in D \cap B''$ for $t = 0$, $\dot{x}_k \in B' \cap D$ for $t = T$); This means that $x_k > 0$ and complementary slack states that $p_k = 0$, which means that the columns of p_k can be erased from the dual problem $\begin{pmatrix} \vec{a}' & 0 \\ G' & 0 \\ F' & -I \end{pmatrix}$.
- Exclude $q_j, \dot{q}_j, u_j$ from the problem, for all $u_j \notin B' \cap B''$ ($u_j \notin D \cap B''$ for $t = 0$, $u_j \notin B' \cap D$ for $t = T$); So $q_j > 0$ and $u_j = 0$ and these columns can be erased from the primal problem $\begin{pmatrix} \vec{c}' & 0 \\ G & 0 \\ H & I \end{pmatrix}$.

Boundary values

To solve the subproblem, it is needed to set the boundary values of the subproblem. The subproblem has two type of boundaries, the boundary at $\theta = 0$ and a boundary at $\theta = 1$.

The boundary values of z', z'' for $\theta = 0$ are set to:

$$\begin{aligned}
&\text{If } v' = \dot{x}_{,},\ v'' = \dot{x}_{,,} \text{ then } x^0_{,,} = \quad 0, \quad x^0_{,} = -\dot{x}^D_{,}, \\
&\text{If } v' = u_{,},\ v'' = u_{,,} \text{ then } q^N_{,,} = -\dot{q}^D_{,,},\ q^N_{,} = \quad 0, \\
&\text{If } v' = \dot{x}_{,},\ v'' = u_{,,} \text{ then } q^N_{,,} = -\dot{q}^D_{,,},\ x^0_{,} = -\dot{x}^D_{,}, \\
&\text{If } v' = u_{,},\ v'' = \dot{x}_{,,} \text{ then } x^0_{,,} = \quad 0, \quad q^N_{,} = \quad 0.
\end{aligned} \tag{4.21}$$

And for $\theta = 1$ the boundary values of z', z'' are:

$$\begin{aligned}
&\text{If } v' = \dot{x}_{,},\ v'' = \dot{x}_{,,} \text{ then } x^0_{,,} = -\frac{1}{2}\dot{x}^{B'}_{,,},\ \ x^0_{,} = -\frac{1}{2}\dot{x}^{B'}_{,}, \\
&\text{If } v' = u_{,},\ v'' = u_{,,} \text{ then } q^N_{,,} = -\frac{1}{2}\dot{q}^{B''}_{,,},\ q^N_{,} = -\frac{1}{2}\dot{q}^{B''}_{,}, \\
&\text{If } v' = \dot{x}_{,},\ v'' = u_{,,} \text{ then } q^N_{,,} = -\frac{1}{2}\dot{q}^{B''}_{,,},\ x^0_{,} = -\frac{1}{2}\dot{x}^{B'}_{,}, \\
&\text{If } v' = u_{,},\ v'' = \dot{x}_{,,} \text{ then } x^0_{,,} = -\frac{1}{2}\dot{x}^{B'}_{,,},\ \ q^N_{,} = -\frac{1}{2}\dot{q}^{B''}_{,}.
\end{aligned} \tag{4.22}$$

(z' only if $t = 0$, z'' only if $t = T$). For all remaining variables the boundary values are set to $x^0_k, q^N_j = 0\ \forall\ \theta$ (except the variables z', z''). The time horizon is set to $T(\theta) = 1,\ \forall\ \theta$.

Initializing the subproblem

The initialization of a subproblem is slightly different compared to a normal SCLP problem. The boundary of a subproblem can be found by making an Initial pivot. In addition, the initial pivot that provides the initial solution is valid for the subproblem in question.

Initial pivots

The initial base-sequence D is optimal for $\theta = 0$. This is the start of the subproblem. To obtain the basis D_- a pivot operation is needed (case iii_a if $v'' = u_{,,}$ or case iv_a if $v'' = \dot{x}_{,,}$). If the base-sequence D_- is not adjacent a new subproblem has to be solved to obtain the sequence of base $D_{-M'}, \ldots, D_{-1}$.

To obtain the base-sequence D_+, a pivot operation according to case iii_b if $v'' = \dot{x}_{,}$ or case iv_b if $v' = u_{,}$. When this sequence is not adjacent a subsubproblem is needed, to obtain the sequence of bases $D_1, \ldots, D_{M''}$.

(If the collision is in the calling problem at $t = 0$, the base-sequence to obtain is just D_+ or $D_1, \ldots, D_{M''}$. When the collision is at the calling problem $t = T$, just obtain the base-sequence D_- or $D_{-M'}, \ldots, D_{-1}$).

Initial solution

There are three possible sequences of adjacent bases, (D_-, D), (D, D_+) or (D_-, D, D_+). For the initial solution a choice has to be made on which one is valid for a neighborhood of $\theta > 0$.

When a subproblem is solved, replace D_- by $D_{-M'}, \ldots, D_{-1}$ and/ or D_+ by $D_1, \ldots, D_{M''}$. This base-sequence is valid for small $\theta > 0$.
(If $t = 0$ the possible sequences are (D, D_+) or $(D, D_1, \ldots, D_{M''})$ and for $t = T$ the sequences are (D_-, D) or $(D_{-M'}, \ldots, D_{-1}, D)$.)

Termination of the subproblem

A subproblem is terminated when an optimal base-sequence includes both B' and B'' for the first time. The resulting base-sequence is: $B', D_1, \ldots, D_M, B''$.
(When $t = 0$, the base-sequence is terminated when B'' enters the optimal sequence,$D_1, \ldots, D_M, B''$.
If $t = T$, the base-sequence is terminated when B' enters the optimal sequence, $B', D_1, \ldots, D_M$.)

After finding the solution to the subproblem the algorithm returns to the main problem. This is done by re-introducing the variables which were excluded. With these steps the algorithm is now able to find all the optimal solutions of an SCLP problem and their duration for which these solutions are optimal.

4.8 Discussion

The SCLP algorithm of Weiss [Wei04] is able to solve a Continuous Linear Programming problem. The algorithm solves the problem by using techniques from Linear Programming (LP) theory.
The algorithm starts with a feasible solution at $t = 0$ or $t = T$, this solution is found by solving an LP problem (Boundary-LP/LP*). The problem is derived from the SCLP problem, by neglecting the control actions ($u(t)$). In the second step the algorithm determines in which direction the optimal solution moves, by using the properties that control variables are piecewise constant and the state variables are continuously piecewise linear. The directions are determined by solving a new LP problem (Rates-LP/LP*), the problem is a function of the control action ($u(t)$) and the derivative of the states ($\dot{x}$). As a result of the Rates-LP/LP* the optimal base-sequences (B' or B''), the set of variables in an optimal solution, is found. When the direction in which the solution moves is known, the duration of this optimal solution can be determined. This is called the validity region. At the end of this interval a new optimal solution needs to be found. The algorithm does this by performing a pivot step, similar as in LP theory. This pivot step causes a change of the lower and upper boundary of the Rates-LP/LP* and by solving this problem again a new optimal solution is found (D). The next step is to determine whether or not D is adjacent, the solutions only differ by one variable, to B' or B''. This is needed to find all the optimal solutions for the desired time horizon. If the solutions are not adjacent a subproblem is created to find the optimal solution for the position between B' and D or D and B''.
By repeating the following step until the desired time horizon has been reached, the algorithm is able to find all the optimal solution. These steps are: determining the Validity Region of the new solution, Pivoting at the end of the interval and changing the lower and upper boundary of the rates, solving the Rates-LP/LP* and creating and solving a subproblem when needed.

With this algorithm it is now possible to solve a continuous-time MPC problem as described in Chapter 3. To use this algorithm the continuous-time MPC problem has to be translated to the formulation used by Weiss. This is done because the continuous-time MPC problem is a function of buffer level (y) and control actions (u) and SCLP is a function of the state variables (x) and the control actions (u). This transformation is shown in Appendix A. The transformation used is one of the possible transformations. Other transformations are possible. The choice for this transformation is based on the formulation that Weiss uses in his examples of similar problems as the one used in this project.

So far the report presented a hierarchical approach to control a manufacturing line. The formulation of the controller and theory behind it was presented and this chapter showed how to solve the problem created by the controller. The next phase is to explain the use of the conversions and this enables us to make a simulation using an event-driven sample scheme.

Chapter 5

Conversion

In the previous chapters focus was on explaining the hierarchial structure and the use of the controller in this structure. In this chapter a problem with different input and output signals between the controller and the manufacturing system is solved. The difference is caused by the fact that the controller used in the hierarchical structure uses a continuous input and output signal, but the models of the manufacturing systems used in this research project use a discrete signal. This means that the signals have to be converted. This is done by designing two converters, one from the manufacturing system to the controller and the other from the controller to the manufacturing system.

When designing these converters, it is important to take into account that the conversions should not have any influence on the dynamics of the controller or the discrete-event system. Otherwise the performance of the controller can not be evaluated properly.

5.1 Conversion from manufacturing system to the controller

The first conversion, the input conversion, is the one going from the manufacturing system to the controller (see Figure 1.1). In the manufacturing system the buffer levels are measured. When the system is measured it is, in fact, a picture of the buffer level. This means that the controller uses information that could have changed at the moment the controller implements its targets. By converting the signals the controller can use information which is closer to the actual buffer levels of the manufacturing system.
When taking into account that the input conversion is not allowed to influence the dynamics of the controller, it results in a converter which has no memory. Considering this aspect there are two possibilities, the first is to use an observer and the second possibility is to determine the fraction of time the lot has been processed and adding this to the buffer level.
In the case of an observer, a model of system is made and used to estimate the input. This estimation can be made by using a Kalman Filter [Gre93]. The filter estimates the actual buffer level, by using the measured data and the system parameters.
The second method is easier to implement, it measures the buffer level and the time a lot has been processed until the measuring moment by dividing processed time by the total process time of that machine, a fraction can be determined and sent to the controller. This fraction

is added to the buffer level of the buffer after the machine and the remaining part of that fraction is added to the buffer level of the buffer in front.

Due to time limitations of this project it was chosen to implement the second method. A better study of observers and a comparison between these methods is advisable.

5.2 Conversion from controller to manufacturing system

The conversion from the controller to the manufacturing system is a conversion where the targets are translated into machine tasks. The targets calculated by the controller are machine throughputs. To translate these throughputs it is modelled that the converter divides the conversion into two parts. The first part calculates the number of lots the machine(s) has to produce in a certain time. The second part is to determine the sequence in which the lots should be made. This second part is only needed when multiple buffers serve one machine.
The conversion of the throughputs can be done by using several techniques. One of these techniques is the scheduling of jobs. This has the disadvantage that the information of the process times has to be deterministic and has to be known a priori. This is not possible when using non-deterministic process times.
Another way to convert the signal is to calculate the number of lots that have to be made during a certain time period and use sequencing policies to determine which machine will process the lot first. In literature several sequencing policies can be found. Some examples of sequencing policies are: First-In, First-Out (FIFO), Earliest Due Date (EDD), Weighted Shortest Process time (WSPT), Shortest Remaining Process time (SRPT), Least Slack (LS), or Least Setup Cost (LSC). Blackstone et al. [Bla82] and Wein [Wei82] have provided a survey of these rules. A more recent policy is the Variable Priority Policy (VPP) by Tsakalis et al. [Tsa97, Var03]. This policy is ideal to use in a system where two buffers serve one machine. It is a combination of two classic heuristic dispatching rules, a first buffer first serve policy (FBFS) for the upstream buffer and last buffer first serve policy (LBFS) for the downstream buffer.
The VPP sequencing policy decides which heuristic rule should be used by calculating the values of the equations (5.1) and (5.2). Then the values are used to determine the position in Figure 5.1, which leads to a dispatching rule and the buffer that should be used.

$$\Delta S_x = AO(S_x) - EO(S_x), \tag{5.1}$$

$$\Delta S_y = AO(S_y) - EO(S_y) \tag{5.2}$$

In (5.1) ΔS_x corresponds to the upstream buffer and in (5.2) ΔS_y corresponds to a downstream buffer. With AO standing for the Actual Out of a buffer of the corresponding stream and EO is the Estimated Outs (the targets) of that stream. The VPP rule also needs a second condition that the chosen buffer is not empty. When both buffers have reached their targets the machine is allowed to be idle.

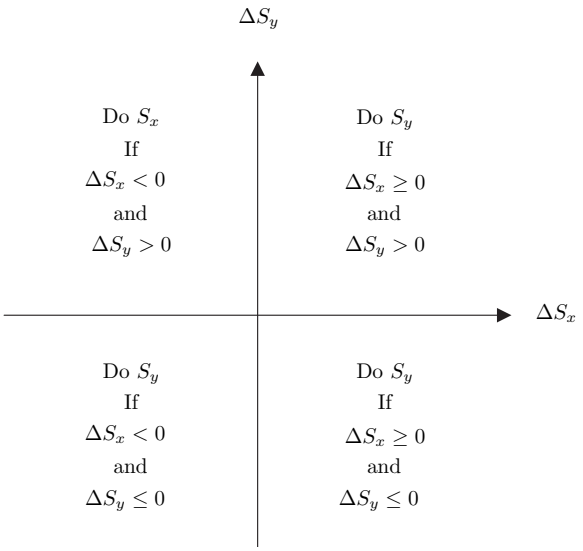


Figure 5.1: *Priority chart*

In this research project the machine targets are calculated by taking the calculated throughput and multiplying them with the time the throughput is optimal. This result in a number of lots that have to be made during each optimal interval. And at the machine level the VPP decides which buffer is served.

5.3 Discussion

The interaction between the controller and the manufacturing line is done by converting the signals between the two parts. This is needed because the controller uses a continuous signal and the manufacturing line uses a discrete signal. The conversion between the manufacturing line and the controller is done by measuring the buffer level and by determining the fraction of time a lot has been processed on that machine. This fraction is added to the buffer after the machine and the remaining part of that fraction is added to the buffer in front of the machine.
The conversion from the controller to the manufacturing line is performed, by calculating the target of each machine. The targets are calculated by taking the optimal throughputs of the controller and multiplying them with the duration that the throughputs are optimal. This results into a set of target that has to be reached in the specified time. At the machine level a sequencing policy determines which lot should be processed first to reach the given target.

With these definitions of the conversions, the hierarchal structure is completed. The next phase is to present the simulation models and the results of the simulation using a event-driven sample scheme.

Chapter 6

Event-driven simulations

The research objective of this project is to investigate the possibility of using an event-driven scheme in a hierarchical approach for inventory control and production optimization of a manufacturing line. To achieve this objective, a hierarchical structure (Figure 1.1) was presented. The elements in this structure were explained in the chapters that followed. So far, the gray marked elements in Figure 6.1 have been explained. This chapter explains the remaining element, the Discrete Event Model (DEM). This chapter first explains the DEM, then it defines what is considered to be a resample-event and finally the simulations and their results are presented.

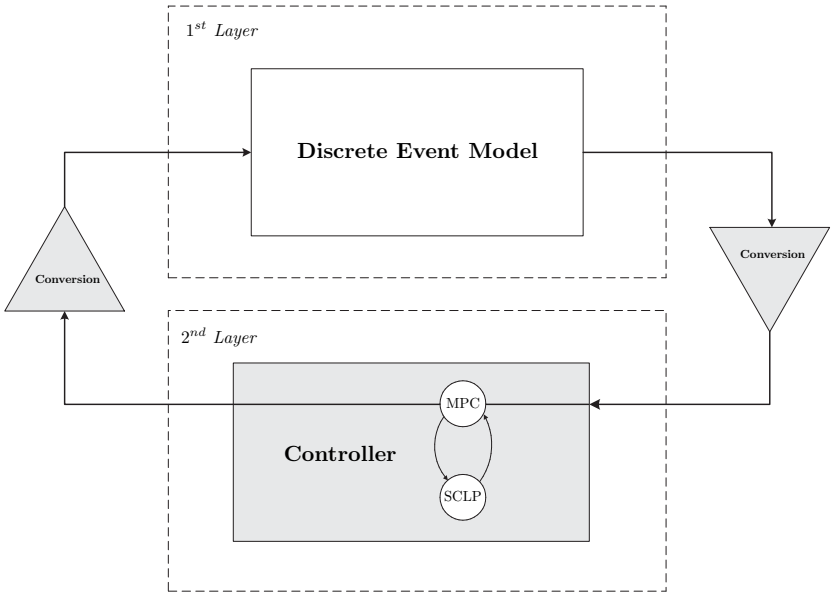


Figure 6.1: *Hierarchical control structure*

6.1 Discrete Event Models

The final step to complete the hierarchical structure is the creation of the DEM, a model of a manufacturing line. To test the new sample scheme, it is best to first test it on a simple manufacturing line. The manufacturing line is represented by a discrete event model of a GBMBME line, as shown in Chapter 3 in Figure 3.1. This model is placed into the hierarchical structure, see Figure 6.2. This figure shows the interactions between the Controller and the DEM and in what languages the elements are programmed.

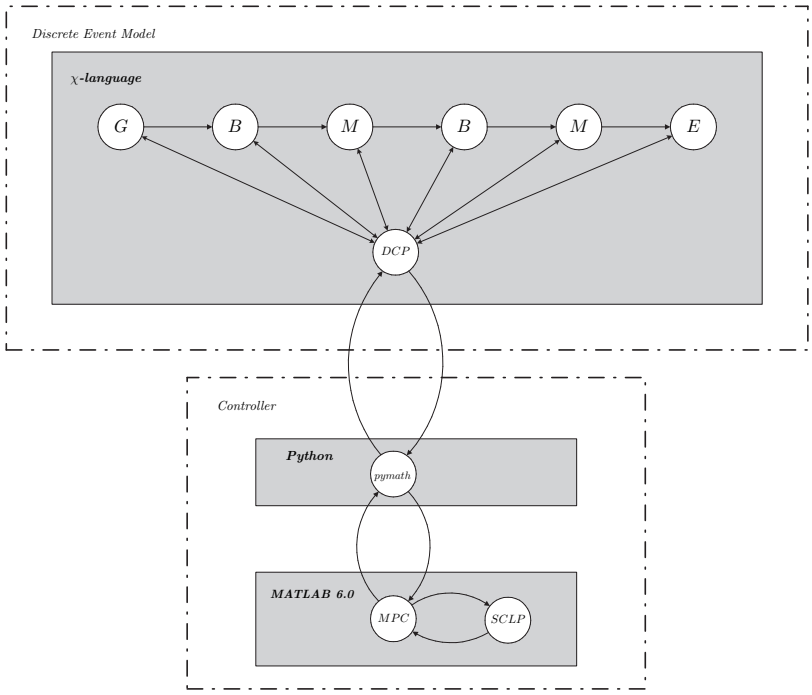


Figure 6.2: *Discrete event Model of GBMBME-line and its interactions*

The DEM is programmed in the specification language χ [Roo01, Ver03] developed by the Systems Engineering group at the Technische Universiteit Eindhoven. The χ-language is a modelling and simulation tool for designing manufacturing systems.

The DEM consist of 7 processes, namely: a Generator (G), two buffers (B), two machines (M), an Exit process, and a dispatcher (DCP). The Generator, buffers, machines and Exit process are processes that speak for themselves, but the dispatcher needs more explanation. The dispatcher is the process that arranges the communication between the controller and DEM, it also performs the conversions and it sends the targets to the machines. For a more elaborate explanation and for the χ-codes the reader is referred to Appendix D.3.

Figure 6.2 also shows the controller, this controller is divided into two parts, namely: the pymath process programmed in Python and the MPC/SCLP controller programmed in MATLAB 6.0. The pymath process is required for the communication between χ and MATLAB, and this communication is needed, because a SCLP solver operates in MATLAB. Ap-

pendix D.2 contains the codes for the communication between MATLAB and χ. For a user manual of pymath see [Ste99, Hof03].
The solver used was supplied by Weiss and the instruction on how to use it, can be found in Appendix D.1.

6.2 Type of sample schemes

In this research project a resample-event is defined as a moment in time when something happens relevant for the manufacturing system. For example an event is the moment when a finished product leaves the system or a machine breaks down. In this research project four type of resample-events schemes are defined:

Type 1) Moments of (re)optimization after 12 hours
Type 2) Moments of (re)optimization after a product finishes on one machine
Type 3) Moments of (re)optimization after a finished product leaves the line
Type 4) Moments of (re)optimization after a machine breakdown

The first type of scheme (type 1) is chosen as a reference, because the discrete-time MPC uses similar types of sample schemes. The second type (type 2) is chosen to see what happens when the targets are determined at a high frequency. The third type (type 3) is chosen to see what happens, when the line is regulated based on finished products. The final sampling scheme is (type 4) a breakdown of a machine, which is an event that is very likely to occur in any manufacturing line. Usually the line is not recalculated after a breakdown. However, the recalculation could be useful to adjust the system to a situation in which the capacity is temporarily reduced.

6.3 Set-up of experiments

In the previous sections the final part of the hierarchical structure was build and the definition of a sample scheme was given. This section focuses on the set-up of the experiments used as basis for the simulations.
The goals of these experiments is to investigate the use of an event-driven sample scheme, the influence of a high or a low frequency of (re)optimization on the DEM, and the third goal is to determine the influence of stochastic process times.
To achieve these goals, three experiments are performed with the system described above. For each goal one experiment is performed. The experiments are:

1. Simulating using one fixed process time for all machines,
2. Simulating using different process times for each machine, the process times are fixed,
3. Simulating using different process times for each machine, the process times are stochastic.

Each experiment is simulated using the four sample schemes defined in Section 6.2 and the initial conditions of the DEM are constant in each experiment. These conditions are:

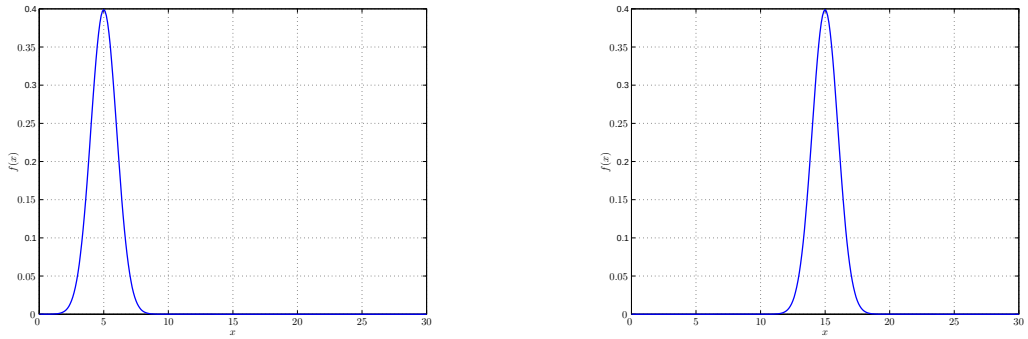
$$\begin{aligned}
&\text{The initial buffer levels} && \Longrightarrow && y = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \ [lots] \\
&\text{The maximal buffer levels} && \Longrightarrow && y_{max} = \begin{pmatrix} 10 \\ 10 \end{pmatrix} \ [lots] \\
&\text{The cost of having lot in the buffer} && \Longrightarrow && p = \begin{pmatrix} -10 \\ -20 \end{pmatrix} \ [100\ Euro/lot] \\
&\text{The cost of producing lot} && \Longrightarrow && r = \begin{pmatrix} 0.2 \\ 0.2 \end{pmatrix} \ [100\ Euro/lot] \\
&\text{A time horizon} && \Longrightarrow && T = \infty \\
&\text{The arrival rate of the system} && \Longrightarrow && \lambda = \begin{pmatrix} 3 \\ 0 \end{pmatrix} \ [lots/hour]
\end{aligned}$$

The last initial value the DEM requires is the process time of the machines. This depends on the type of experiment that is performed. In the case that experiment 1 is performed, the process time of both machines are set at 5 $[min/lot]$, in the second experiment the process time of the first machine is set at 5 $[min/lot]$ and the second machine has a process time of 15 $[min/lot]$, and in the final experiment the process times of both machines are stochastic, with a mean process time equal to the process times of the second experiment.

The stochastic process times are determined by taking the average process times of the machines, 5 $[min/lot]$ for the first machine and 15 $[min/lot]$ for the second machine, and adding a normal distribution with a mean of 0.0 and a variance of 1.0 to the process times. It is important that the process time may not become negative. This is achieved by checking the probability of a negative process time. The result are shown in Figure 6.3, these figures show the probability density function of the process times. The probability of a negative process time is e^{-6} for machine 1 and e^{-50} for machines 2. These probabilities are very small, which means that a new pulling is sufficient to prevent the use negative process time.

Experiment	Sample	Process time M_1	Process time M_1	Stochastic process times
1	type 1	5 $[min/lot]$	5 $[min/lot]$	n/a
	type 2	5 $[min/lot]$	5 $[min/lot]$	n/a
	type 3	5 $[min/lot]$	5 $[min/lot]$	n/a
	type 3	5 $[min/lot]$	5 $[min/lot]$	n/a
2	type 1	5 $[min/lot]$	15 $[min/lot]$	n/a
	type 2	5 $[min/lot]$	15 $[min/lot]$	n/a
	type 3	5 $[min/lot]$	15 $[min/lot]$	n/a
	type 4	5 $[min/lot]$	15 $[min/lot]$	n/a
3	type 1	5 $[min/lot]$	15 $[min/lot]$	$\mu = 0.0$ and $\sigma^2 = 1.0$
	type 2	5 $[min/lot]$	15 $[min/lot]$	$\mu = 0.0$ and $\sigma^2 = 1.0$
	type 3	5 $[min/lot]$	15 $[min/lot]$	$\mu = 0.0$ and $\sigma^2 = 1.0$
	type 4	5 $[min/lot]$	15 $[min/lot]$	$\mu = 0.0$ and $\sigma^2 = 1.0$

Table 6.1: *Machine process times.*



(a) Process time, machine 1

(b) Process time, machine 2

Figure 6.3: *Probability density functions of the process time*

6.4 Expected results

Now, the simulation models were created, the sample schemes were defined, and the experiments were formulated, the next step is to predict how the DEM reacts to the control actions of each sample scheme.

MPC/SCLP controller

The first expectation is to know how the MPC/SCLP controller reacts. The optimization function of the controller is designed in such away that, when a stock is in the line, it tries to empty the stock. As a result, it reduces the cost of cost-function. These costs are higher for lots that are further in the line, so these are emptied first. This means that the controller firsts set targets that will empty the buffers and subsequently issue a target that is equal to the rate in which the lots enter the line.

DEM

The expectation of the DEM is divided into four categories which are directly related to the sample schemes defined in Section 6.2. The first category is Type 1.

Type 1. *Moments of (re)optimization after 12 hours*

The first sampling scheme is the scheme where (re)optimization occurs after 12 hours. This frequency of (re)optimization is considered to be a low frequency. This means that the controller is not able to adjust the DEM when the buffer increases, until the next (re)optimization moment.

In the first experiment, all machines have the same process time, the controller starts emptying the buffers. When the buffers are empty, the controller and DEM produce with the

same rate as arrival rate of the lots. In the second experiment, the machines have different fixed process times and the controller tries to empty the buffers. However, since the process times differ, the buffer of the slower machine is initially increase. In the third experiment, the machine with stochastic process times is almost the same as in the second experiment. In this case the stochastic behavior has the consequence that the DEM might not reach its targets and this causes an increase of the buffer level.

Type 2. *Moments of (re)optimization after a product finishes on one machine*

In this case the (re)optimization is determined after a product has finished on a machine and the controller operates at a high frequency. It is expected that the controller will have difficulties to stabilize to zero due to the many control actions, so that the DEM cannot reach its targets.
When experiment one is executed, the DEM starts emptying the buffers and when it reaches zeros it starts fluctuating between zero and certain buffer level. In the second and third experiment the DEM also starts emptying the buffers, but it takes longer because of different process times per machine and the stochastic nature of the process times.

Type 3. *Moments of (re)optimization after a finished product leaves the line*

In this type of sample scheme the departure of a product triggers the controller. The number of products that leave the line can be large. This means that the controller has to (re)optimize at a higher frequency. This results in a lot of control actions and this causes that the DEM does not reach the targets given. The expectations are the same as with the sample scheme type 2, but with a lower intensity.

Type 4. *Moments of (re)optimization after a machine breakdown*

The last expectation is of a sample scheme, where the sample is performed after a machine breakdown. This sample scheme is expected to show that the DEM is controlled with a low frequency of control actions, which can cause the DEM to initially operate at a higher buffer level. In the first experiment the DEM starts to empty the buffers, when the buffers are empty it produces with the same rate as the arrival rate. The second experiment shows that, when using this sample scheme, the buffer level of the first buffer does not fluctuate close to a buffer level of zero, because the second machine produces at a slower rate. In the final experiment the DEM is expected to empty the buffers and also to fluctuate close to an empty buffer level. However, the time in which the buffer is emptied is longer, due to the fact that not all targets are reached. The stochastic process time causes an error between the target and the actual output.

6.5 Results

This section presents the results of each experiment, with the four different sample schemes. These categories show the results of the three experiments of the specified sample scheme. Each experiment shows the two buffer levels, the level of the exit buffer, and a figure that shows the error between the actual output and the target buffer level.

In the figure of the buffer level, the red line represents the calculated targets issued by the MPC/SCLP controller (translated to buffer levels), the blue lines are the buffer levels produced by the DEM. The plot of the exit buffer also show the desired output level. This level is determined by using the arrival rate ($\vec{\lambda}$) of the system multiplied it with the duration of the simulation.
Then, the first three categories are compared to answer the questions formulated in the goal of this experiment. The fourth category is presented to show the effects of an event-driven sample scheme.

Type 1. Moments of (re)optimization after 12 hours

The first category is the sample scheme, that samples every 12 hours. This means that the controller (re)optimizes after 12 hours and calculates the targets for that period. The simulations have a duration of 2 shifts of 12 hours.

Experiment 1

In the first experiment the process times are fixed and both machines have the same processing rate (see Table 6.1) . The results are shown in Figure 6.4 and Figure 6.5. Figure 6.4 shows the buffer levels and the level of the exit process and Figure 6.5 shows the error between the actual output and the estimated output. The first figure in Figure 6.4 shows the level of the first buffer. This buffer receives the lots with a rate of 3 [$lots/hour$]. At the beginning of the simulation the first machine is idle, this causes the buffer to increase. The controller sets the machine to idle to enable the second machine to empty its buffer first, because those lots have more added economical value than the lots in the first buffer. After 0.25 [$hours$] the second buffer is emptied and the first buffer starts emptying its content. The second machine operates with the same rate as the first since both machines have the same process time, which means that the lots are almost immediately processed. When both buffers are empty the machines produces with the same rate as the lots arrive. The MPC/SCLP controller shows this by forcing the buffer level to zero and with the DEM this can be seen the by fact that when the DEM receives a lot, it immediately sends the lot to the machine.
The final figure of Figure 6.4 shows the cumulative number of lots produced during the simulations. The dotted line shows the rate of finished products. The cumulative level of the DEM and controller are higher than the arrival rate. A closer look shows that the difference between the level of the arrival rate and the number of lots leaving the system is roughly 5 *lots* and this is the number of lots that initially were in the buffers.

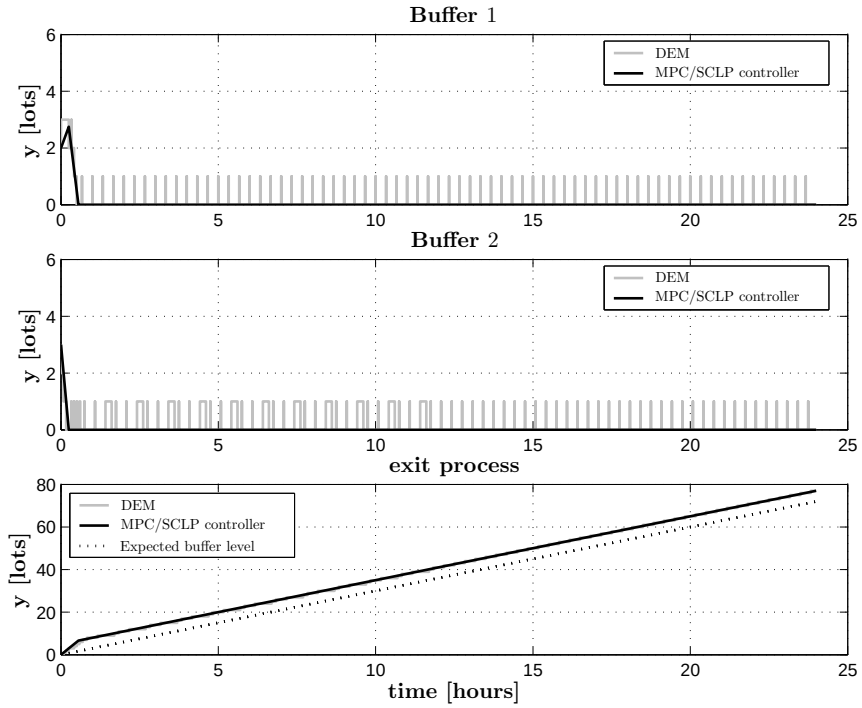


Figure 6.4: *Buffer levels of experiment* 1 *with sample scheme type* 1

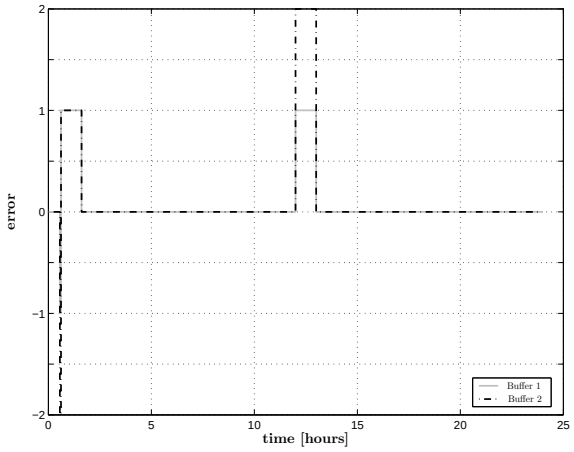


Figure 6.5: *Error between actual output and the estimated output of experiment* 1 *with scheme type* 1

Figure 6.5 shows the error between the actual output and the estimated output. The figure shows that DEM is reaches its targets most of the time except in one situation. This is caused by the fact that the buffer is empty and is waiting for a new lot to arrive. After this error, the system corrects this by producing more. In the figure it seems that the reaction to the shortage is longer, but this is caused by a numerical delay of the DEM. The DEM uses the optimal targets of the controller, but when the sample period is longer than the beginning of the last validity region the targets are calculated by taking one hour and multiplying this with the target. This is repeated until the end of the sample. However, the error measurements

are taken at the moment a new target is calculated, which results in a larger error.

The second error in Figure 6.5 occurs at moment a (re)optimization is done (after 12 *hours*). At this point it is possible that the actual output is lower then the estimated output, because the time period of that target has not yet ended.

Experiment 2

The second experiment uses a fixed process time and the machines have different processing rates (see Table 6.1). The first machine produces with a rate of 12 [*lot/hour*] (process time of 5 [*min/lot*]) and the second machine produces with a rate of 4 [*lots/hour*] (process time of 15 [*min/lot*]). In Figure 6.6 and Figure 6.7 the results experiment 2 are presented for a simulation of 24 *hours*. These results are similar to the results of experiment 1. The difference is that due to the different processing times, the second machine takes longer to empty its buffer and this causes an increase of the first buffer. When the first machines start emptying its buffer it processes the lots at a higher rate than the second machine, this causes an increase of the second buffer. The figure also shows that the control signal is different from the DEM signal. This is caused by the fact that the controller only sets the targets at the beginning of the sample period. The second machine processes the lots of machine 1. When both buffers reach the level of zero, the system has reached it steady state.

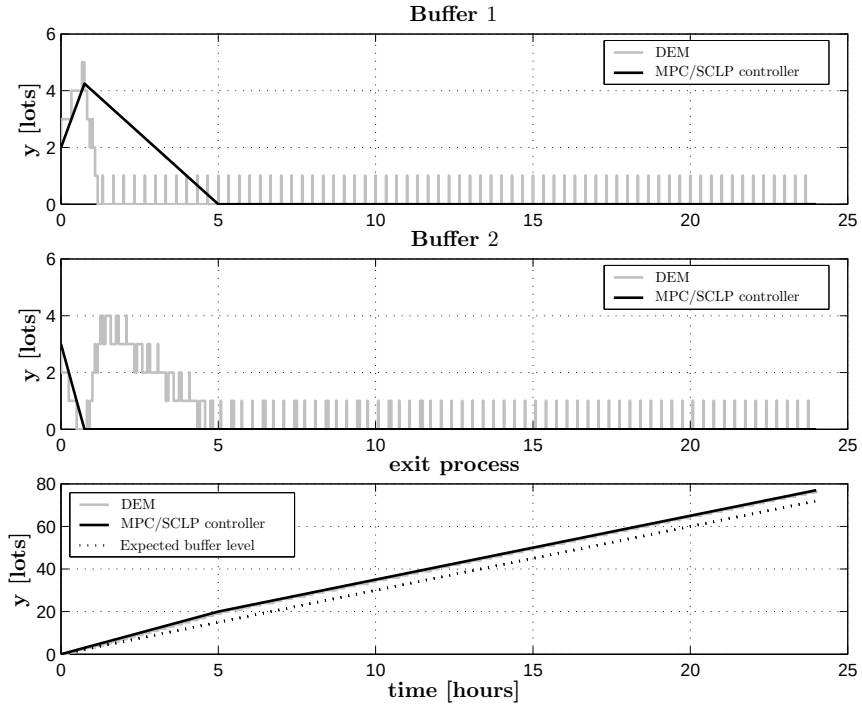


Figure 6.6: *Buffer levels of experiment* 2 *with sample scheme type* 1

The error plot in Figure 6.7 has the same moments when an error occurs, namely: the moment a buffer reaches an empty state or when the (re)optimization is done. However there is something wrong with the error moment when a buffer becomes empty. In Figure 6.7, buffer 2 become empty at $t = 1$ *hour* and at that point an should occur.

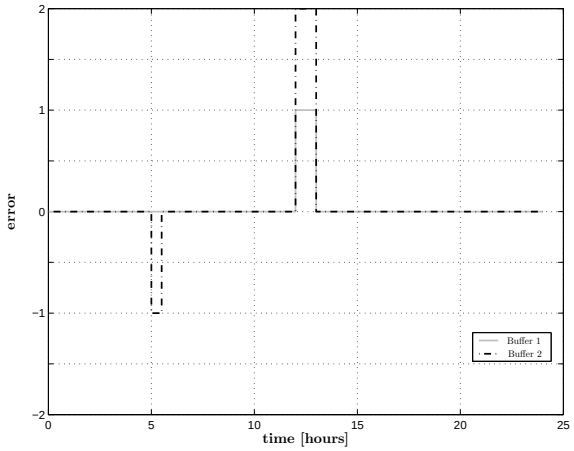


Figure 6.7: *Error between actual output and the estimated output of experiment* 2 *with scheme type* 1

Experiment 3

The final experiment for this sample scheme uses a stochastic process time and the machines have different processing rates (see Table 6.1). The mean of these stochastic process time is equal to the process time described in experiment 2. The results are presented in Figure 6.8 and Figure 6.9. The simulations are performed for a duration of 24 [$hours$], just as in the first two experiments. The results of the stochastic system have the same structure as the results of experiment 2, with the difference that the system needs more time to reach its steady state. The stochastic process times causes a higher level of the second buffer. The second machine causes the second buffer to increase due to the fact that the process times are longer. The stochastic process times are also has the consequence that the buffer is sometimes not able to send a lot directly to the machine.
The error plot shows the error of the actual output minus the estimated target. The results of experiment 3, Figure 6.9, show that the number of error are larger than in the previous experiments. The errors occur at the moment the buffer is empty and waiting for a new lot. In Figure 6.9 the duration of this error is larger then in previous experiment, this is caused by the fact that the controller presumes that the buffer is empty, but the buffer in the DEM increases to a level of 4 $lots$. Due to the low frequency of (re)optimization an error occurs.

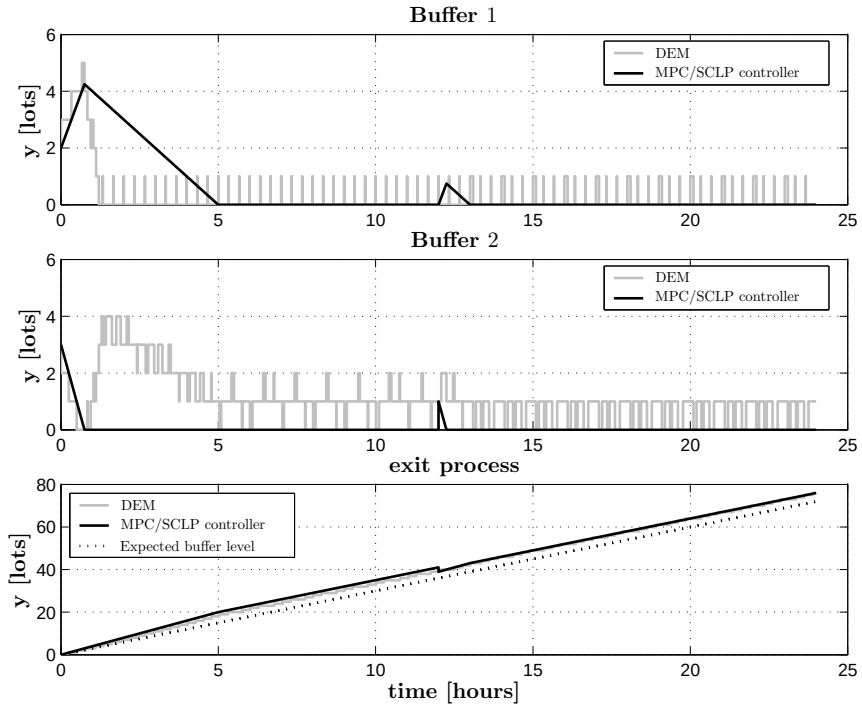


Figure 6.8: *Buffer levels of experiment* 3 *with sample scheme type* 1

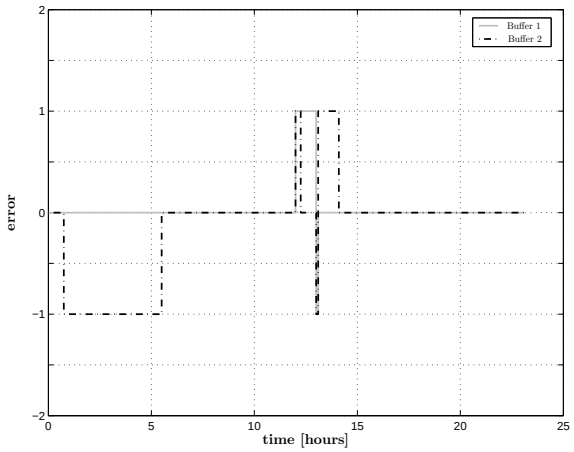


Figure 6.9: *Error between actual output and the estimated output of experiment* 3 *with scheme type* 1

Discussion

Comparing the sample types of experiment 1

The results of experiment 1 showed that when the results of the sample types are compared the differences are very small. With sample type 1 the steady state is reached after 1 hour. Sample type 2 takes about 5 hours to reach its steady state mode and sample type 3 reaches the steady state mode after 13 hours, with the notion that due to the stochastic process times

the lots remain a little longer in the buffer compared to the other sample types.
The error figures of experiment 1 all show that at the moment of (re)optimization an error occurs. However the results of the sample type 2 and 3 have a different shape, but the reason for the difference between the errors at the moment a buffers empties is not clear.

Comparing the results with the expectations

When comparing the results of the simulations with the expectations given in Section 6.4, the results of the first experiment are correct. The buffers are first emptied and when they are empty they produce with the rate the lots arrive. The second expectation is slightly different, it does take longer to reach the empty state of the buffer and the second buffer increase initially. However, the system corrects this and reaches the same state as in experiment 1. The differences between the prediction and the actual results are caused by the fact that the prediction did not take the idle time of the first machine into account. The third expectation was wrong, the controller is influenced by the stochastic of the process times, but the DEM and controller are able to adjust in such away that the system reaches a steady state. During the prediction it was expected that the stochastic would have more influence on the system.

Type 2. Moments of (re)optimization after a product finishes produced on one machine

The second category is the simulation were a sample scheme is used that samples after a product has be produced on of the machines. These simulations have been running for 150 sample periods which is equal to 2 shifts of 12 $[hours]$.

Experiment 1

In the first experiment the process times are fixed and both machines have the same processing rate (see Table 6.1). The results are shown in Figure 6.10 and Figure 6.11.

The results show that the controller initially performs a lot of control actions to empty the buffers. After that the control actions have little influence on the behavior of the system. This is logical, because when the buffers are empty, the system produces with the same rate as the arrival rate. This means that each lot that comes into the buffer is immediately sent to the machine.
The exit process shows that the line produces with the rate of the arrival rate and it has an offset of the size of the total initial buffer level.

In Figure 6.11 the error is plotted. The figure shows that the error is between zero and one and the number of error is larger due to the high frequency of (re)optimizations.

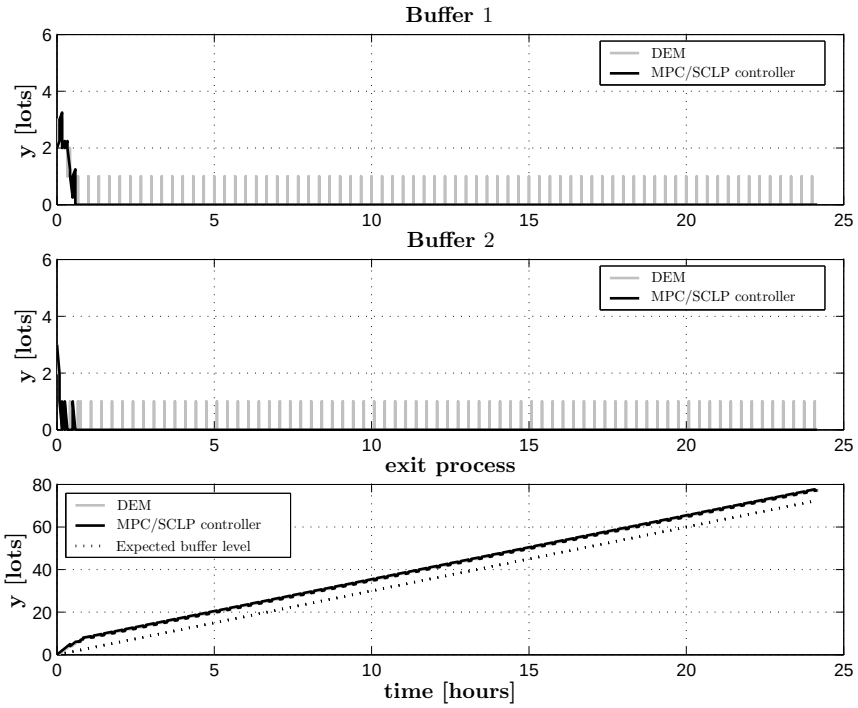


Figure 6.10: *Buffer levels of experiment* 1 *with sample scheme type* 2

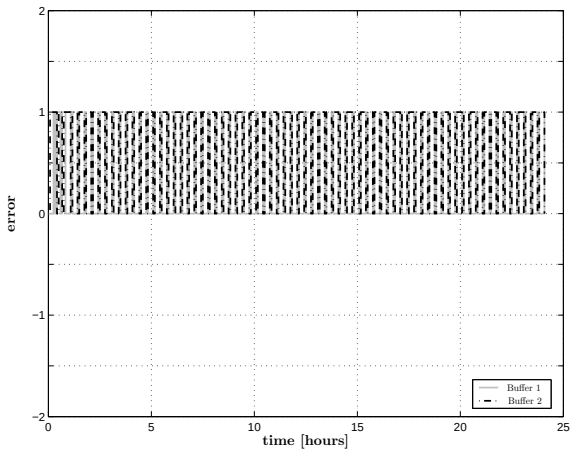


Figure 6.11: *Error between actual output and the estimated output of experiment* 1 *with scheme type* 2

Experiment 2

This experiment uses a fixed process time and the machines have different processing rates (see Table 6.1). The first machine produces with a rate of 12 $[lot/hour]$ (process time of 5 $[min/lot]$) and the second machine produces with a rate of 4 $[lots/hour]$ (process time of 15 $[min/lot]$). The results are presented in Figure 6.12 and in Figure 6.13.

The results of the second experiment show that controller needs more time to empty its

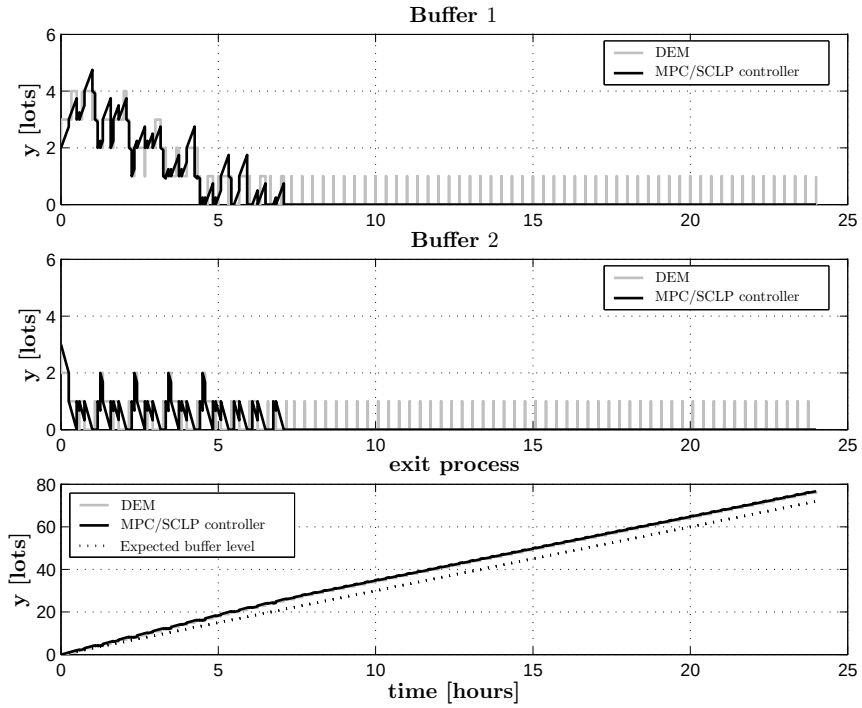


Figure 6.12: *Buffer levels of experiment* 2 *with sample scheme type* 2

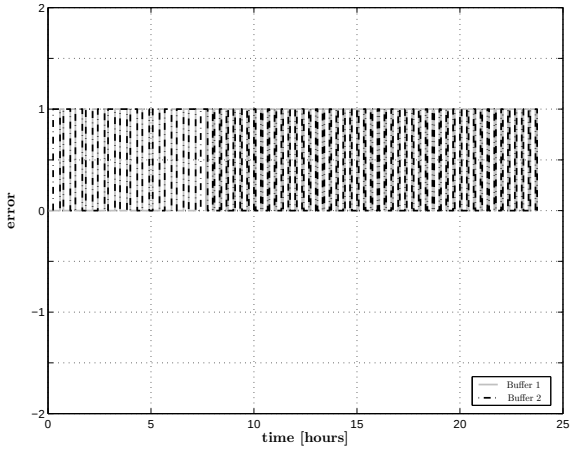


Figure 6.13: *Error between actual output and the estimated output of experiment* 2 *with scheme type* 2

buffer, similar as in experiment 2 of type 1. The difference with that figure is that the control and DEM signal are almost the same. In the second figure of Figure 6.12, the second buffer empties its content but the level does not stay close too zero, because the first machine has not reached zero yet and produces with higher rate than the second machine. After 8 [$hours$] the system reaches it steady state.

The third figure in Figure 6.12 and the error plot in Figure 6.13 are the same as in experiment 1.

Experiment 3

The final experiment for this sample scheme uses a stochastic process time and the machines have different processing rates (see Table 6.1). The mean of these stochastic process time is equal to the process time described in experiment 2. The results are presented in Figure 6.14 and Figure 6.15.

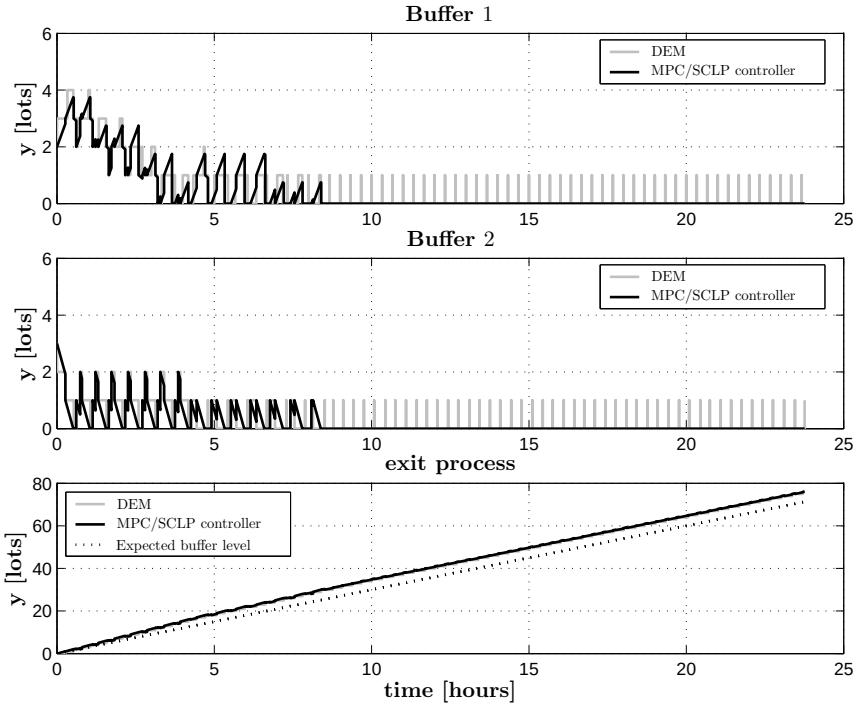


Figure 6.14: *Buffer levels of experiment* 3 *with sample scheme type* 2

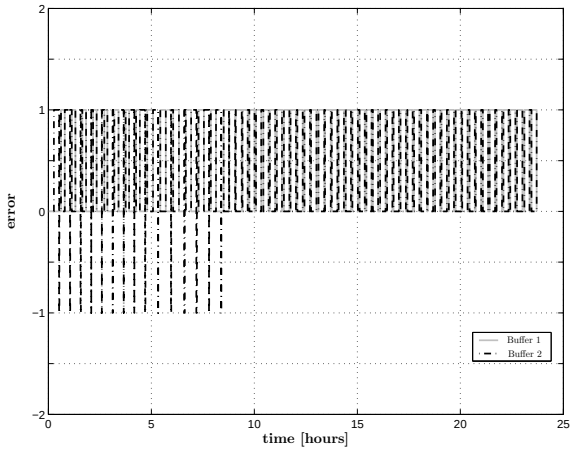


Figure 6.15: *Error between actual output and the estimated output of experiment* 3 *with scheme type* 2

In the third experiment the process times become stochastic. This influence of the stochastic process times on the result using the current sample scheme are limited. The buffer level initially fluctuates a bite more, but the system reaches its steady state at the same moment as in experiment 1. The stochastic process times influence the calculated error of the second buffer. In the transient time period the error fluctuates between one and minus one.

Discussion

Comparing the sample types of experiment 2

When comparing the transient periods of the results of experiment 2, the transient time of sample type 1 is smaller than the time of the other types. Sample type 1 reaches the steady state mode after 1 *hour*. And the types 3 and 4 reach the steady state after 7 respectively 8 hours.
The error all have the same characteristic features, expect that the error of sample type 3 differs during the transient time. This is due to the stochastic process times.

Comparing the results with the expectations

In Section 6.4 the expectations were that the system would have difficulties with finding its steady state. The results show that only when the buffer is not near to zero, the system has some difficulties with stabilizing. But long after the buffers are emptied the system operates in a steady state mode. It was also expected that in the second and third experiment the DEM would need more time to empty the buffer. The results confirmed this.

Type 3 Moments of (re)optimization after a finished product leaves the line

The final case of the simulations, is the case where a product leaves the line is considered an event. The simulations are run for 78 product leaving the lines (this is equal to a duration of one day). The results are shown in Figure 6.18 - 6.21.

Experiment 1

In the first experiment the process times are fixed and both machines have the same processing rate (see Table 6.1). The results are shown in Figure 6.16 and Figure 6.17. Experiment 1

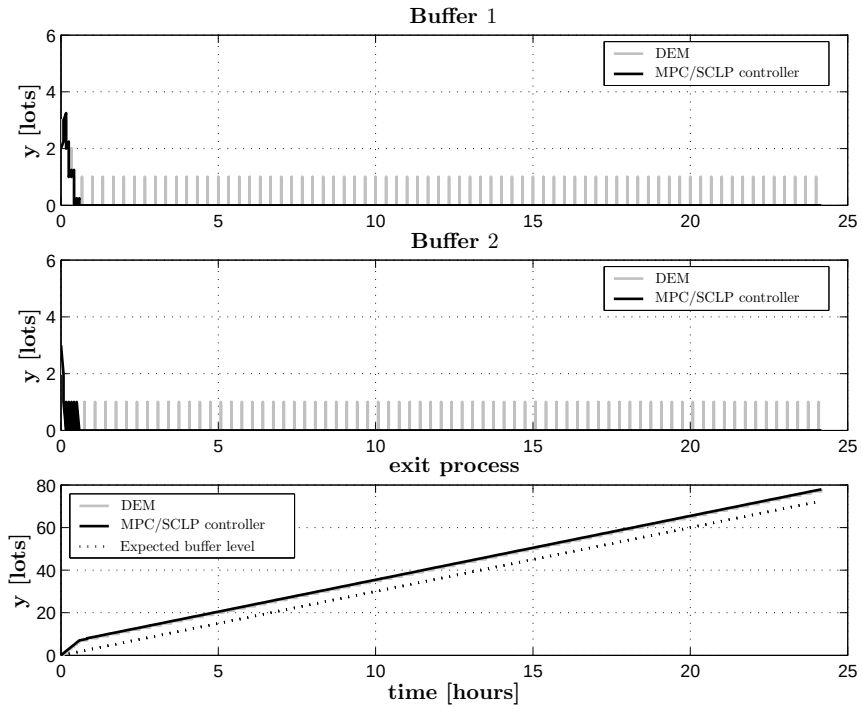


Figure 6.16: *Buffer levels of experiment* 3 *with sample scheme type* 1

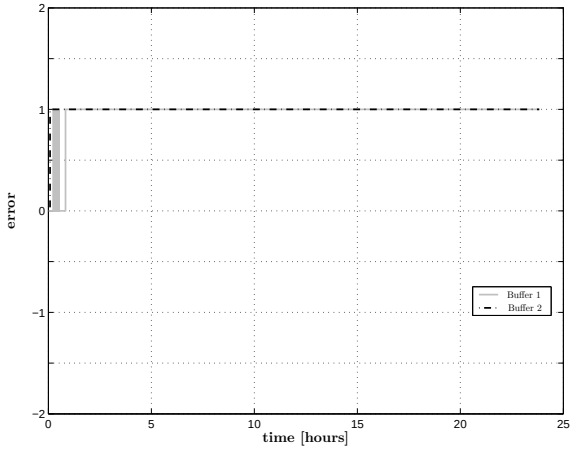


Figure 6.17: *Error between actual output and the estimated output of experiment* 3 *with scheme type* 1

showed identical results as the results of experiment 1 with sample type 2. The only differences are the errors. In this experiment the error only fluctuates at the beginning between zero and one. And it becomes one after the line has reached it steady state.

Experiment 2

This experiment uses a fixed process time and the machines have different processing rates (see Table 6.1). The first machine produces with a rate of 12 $[lot/hour]$ (process time of 5 $[min/lot]$) and the second machine produces with a rate of 4 $[lots/hour]$ (process time of 15 $[min/lot]$). The results are presented in Figure 6.18 and Figure 6.19.

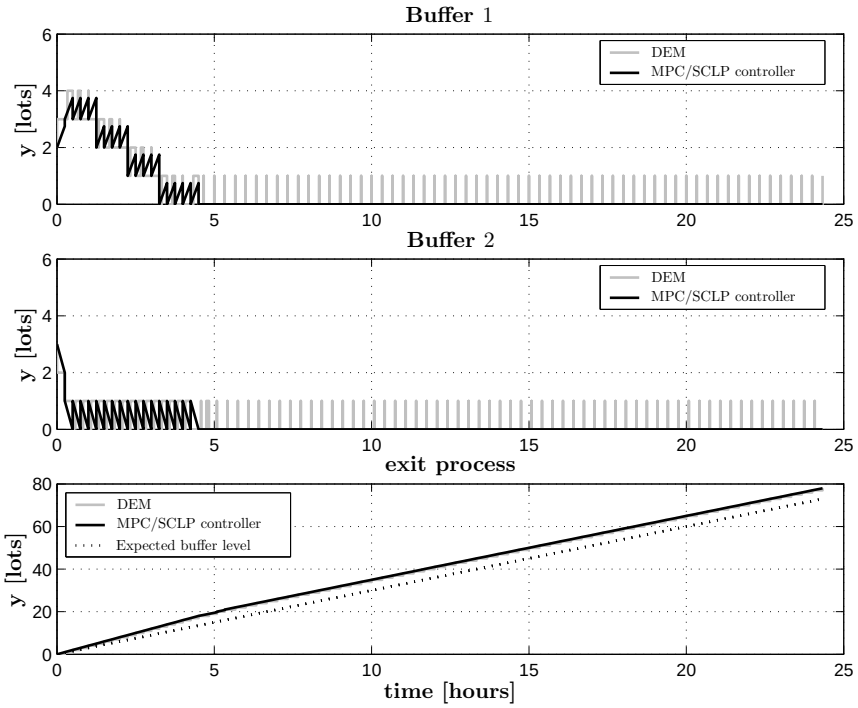


Figure 6.18: *Buffer levels of experiment* 2 *with sample scheme type* 3

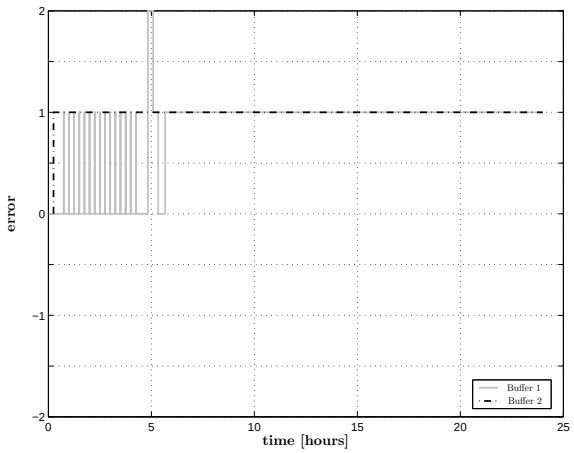


Figure 6.19: *Error between actual output and the estimated output of experiment* 2 *with scheme type* 3

The second experiment also shows that there is no difference between sample type 2 and 3. Only the error is slightly different, it has a lower frequency, but the errors are between zero and one. Except for one measurement and this is caused by the fact that the buffer became empty.

Experiment 3

The final experiment for this sample scheme uses a stochastic process time and the machines have different processing rates (see Table 6.1). The mean of these stochastic process time is equal to the process time described in experiment 2. The results are presented in Figure 6.20 and Figure 6.21.

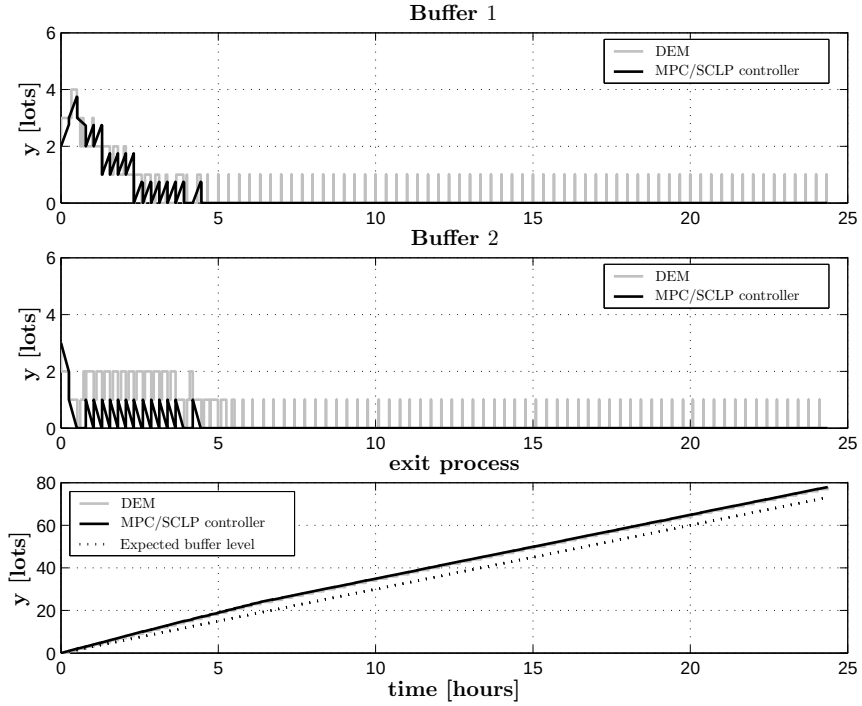


Figure 6.20: *Buffer levels of experiment* 3 *with sample scheme type* 3

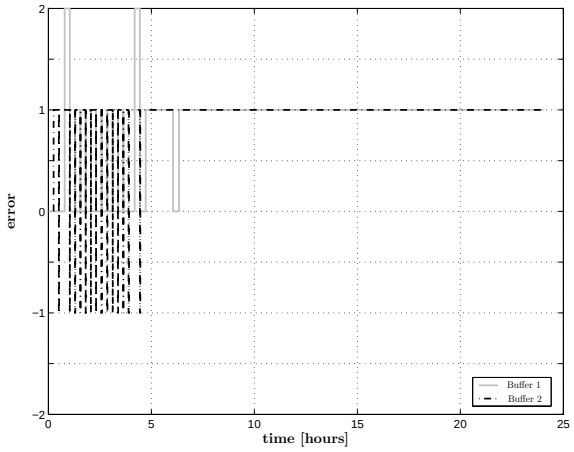


Figure 6.21: *Error between actual output and the estimated output of experiment* 3 *with scheme type* 3

In the third experiment the results show a more regular pattern than the results of experiment 3 with a sample type 2. The stochastic has less influence on the behavior of the controller. The same applies for the error plot.

Discussion

Comparing the sample types of experiment 3

The result show that sample type 1 reaches a steady state mode after 1 hour and the other two sample types reach the steady state mode at $t = 4$ *hours*.
During the steady state mode, the error of the sample types have the same shape and value. But during the transient phase the samples differ slightly, due to the stochastic process times.

Comparing the results with the expectations

The expectation was the the behavior would be the same as in sample type 2, but with a lower intensity. And the results confirmed these findings.

Type 4 Moments of (re)optimization after a machine breakdown

The third sample scheme is the sample after a machine breakdown. The machine breakdowns are created by using an exponential distribution, with a mean of one shift of 12 [*hours*]. The simulations were run for 5 breakdowns. The duration of the simulation is different than other experiments, due to stochastic breakdown time the The results are shown in Figure 6.22-Figure 6.27.

Experiment 1

In the first experiment the process times are fixed and both machines have the same processing rate (see Table 6.1). The results are shown in Figure 6.22 and Figure 6.23.

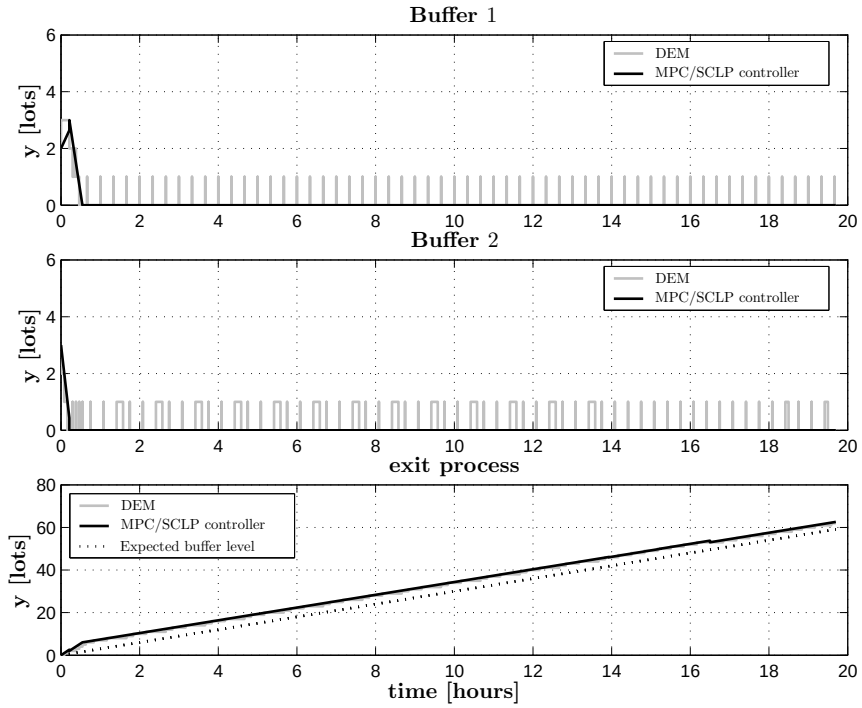


Figure 6.22: *Buffer levels of experiment* 1 *with sample scheme type* 4

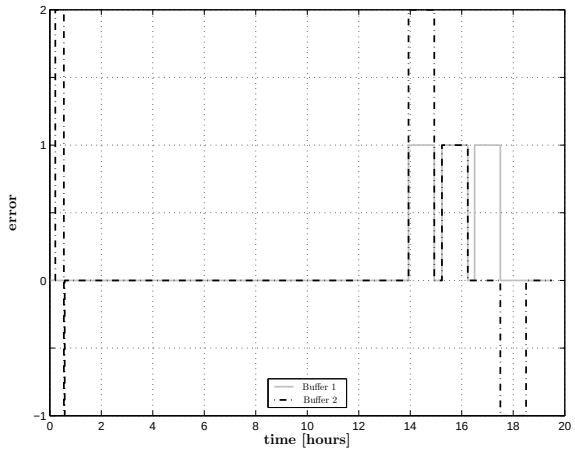


Figure 6.23: *Error between actual output and the estimated output of experiment* 1 *with scheme type* 4

The results of experiment 1 show that, when evaluating a system with equal process times, a breakdown has little influence on the output of the line. It is noted that during the (re)optimization de capacity constraints did not change, only the buffer level changed.
In this situation the results are similar to the results found with experiment 1 with a sample type 1.

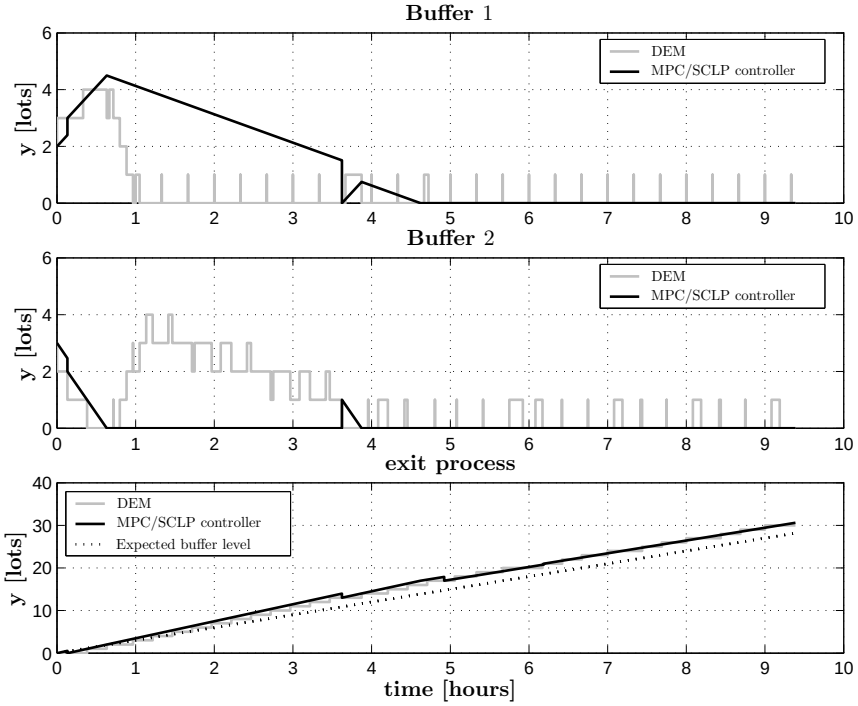


Figure 6.24: *Buffer levels of experiment* 2 *with sample scheme type* 4

Experiment 2

This experiment uses a fixed process time and the machines have different processing rates (see Table 6.1). The first machine produces with a rate of 12 $[lot/hour]$ (process time of 5 $[min/lot]$) and the second machine produces with a rate of 4 $[lots/hour]$ (process time of 15 $[min/lot]$). The results are presented in Figure 6.24 and Figure 6.25.

The second experiments show that the control actions are low, because there is a big difference in action between the controller and the DEM. It also shows that when the system has emptied its buffer the controller actions are very limited.

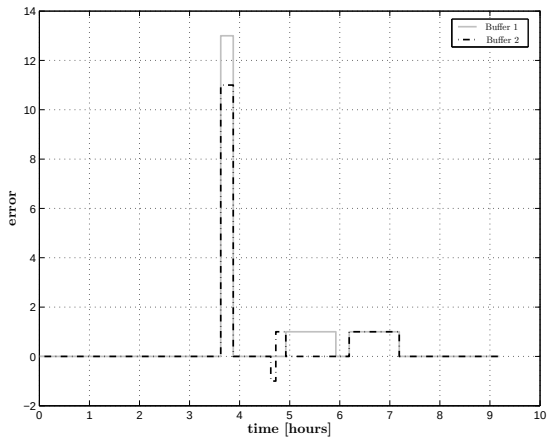


Figure 6.25: *Error between actual output and the estimated output of experiment* 2 *with scheme type* 4

Experiment 3

The final experiment for this sample scheme uses a stochastic process time and the machines have different processing rates (see Table 6.1). The mean of these stochastic process time is equal to the process time described in experiment 2. The results are presented in Figure 6.26 and Figure 6.27.

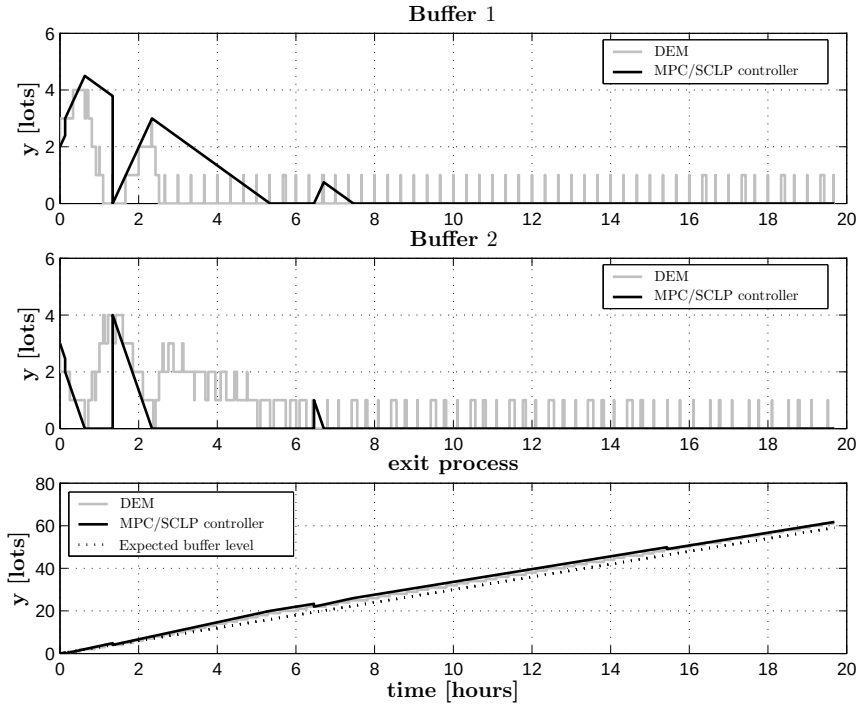


Figure 6.26: *Buffer levels of experiment* 3 *with sample scheme type* 4

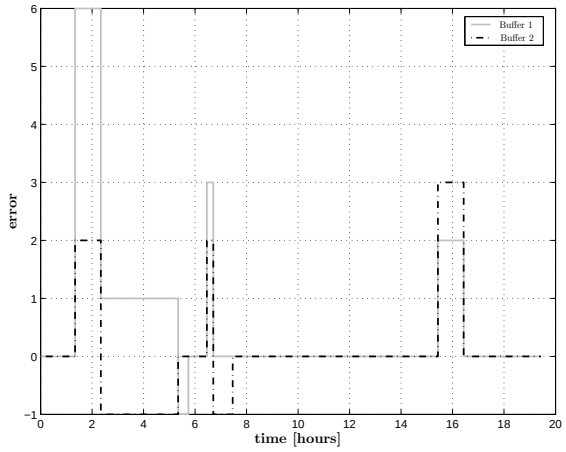


Figure 6.27: *Error between actual output and the estimated output of experiment* 3 *with scheme type* 4

Figure 6.26 shows the results of the buffer levels. The results are slightly different from the results found in experiment 3 with sample type 1, because during the period that the buffers are being emptied a breakdown occurs and the control actions are changed. The controller focuses the DEM to empty the buffers. However, the machine is not able to process all the lots that arrive in the buffer and the buffers increase.
The error plot, Figure 6.27, shows that when a breakdown happens, an error occurs.

Discussion

Comparing the sample types of experiment 4

The comparison of the sample types of experiment 4 is not possible, the problem is caused by the differences in length of breakdown intervals. However, the pattern of the results are similar compared to experiment 1.

Comparing the results with the expectations

For sample type 4 the expectations were that the buffers are empty with a limited number of control actions. The results show that the controller starts emptying the buffers and the DEM deviates from the signal of the controller except when a breakdown occurs. In that case the deviation is smaller.

6.6 Global discussion

The sample types of each experiments are already compared with each other. It is now time to compare the experiments with each other, this is done for the experiments 1–3. Experiments 4 is left out because of the difference in the durations of the simulations.
The first comparison is made between the low frequency and high frequency (re)optimization. When the controller (re)optimizes with al low frequency, the lots stay longer in the buffer until

a (re)optimization is performed. The signals of the DEM and the controller initially differ due to the low frequency of (re)optimization. With a high frequency of (re)optimizations this problem does not occur.
When the results of experiment 1 are compared with the results of experiment 2 and 3 for each sample type. The results show that in experiment 1 for all of the sample types the controller is able to bring the DEM into a steady state after approximately 1 hour. In case of experiment 2 the length of the transient times differ, sample type 1 has transient period of 4 $hours$, sample type 2 has a transient period of 7 $hours$, and sample type 3 has a transient time of 5 $hours$. With the experiment 3 the transient periods are 13 $hours$, 8 $hours$, and 5 $hours$ respectively. In all the sample types the initial buffer levels are higher then the other experiments.

The comparison of the error plots is difficult, because the way they are measures in the model might not be correct.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

The research objective of this project is to investigate the possibility of using an event-driven sample scheme in a hierarchical approach for inventory control and production optimization of a manufacturing line. In addition to the first objective it was desired to investigate the influence of the conversion of the continuous MPC into discrete-event decisions on the controller.

To achieve these objectives a hierarchical framework was build and an MPC controller was designed. In order to use the MPC controller it is important to understand the SCLP algorithm, created by Weiss [Wei04]. The principle of this algorithm is relatively simple, but the implementation of the algorithm is more complex and difficult to understand. In this report it was attempted to explain the principles of the algorithm and how the algorithm should be implemented. Due to the complexity of the algorithm there are still a few questions how the algorithm should be used and which limitations the algorithm has. But with the use of the SCLP solver designed by Weiss, it was possible to simulate the concept of an event-driven sample scheme with test cases of a manufacturing line.

In the investigation of the influence of the conversion of the continuous MPC into the discrete event decisions, several types of conversion were explored. The final conclusion is that when a controller is used to control a discrete-event system with variable process times, it is optimal to use a conversion method that consist of two parts. The first part is used to translate the throughput targets into the number of lots a machine has to produce during a time interval. The last part uses sequencing policies to decide which buffer should serve the machine in order to reach the estimated targets. The second part of the conversion is only needed when the line has machines with multiple buffers.

With the knowledge of the hierarchical approach, the MPC controller, the SCLP algorithm, and the conversions, a number of simulations were carried out with four different type of sampling schemes. The sample schemes investigated where: 1) every 12 hours, 2) after a product finishes on one machine, 3) a product leaves the line and 4) after a machine breaks down.

The sample schemes are tested by using a simple model of a manufacturing line with two buffers and two machines. Both machines are connected to one single buffer. The focus of these tests was to investigate the possibility of an event-driven sample scheme.
The simulations were carried out for three experiments, that used a model of a GBMBME-line. In the first experiment the process times are fixed at 5 $[min/lot]$, the second experiment also uses a fixed process times, but the process times are different (the first machine has a process time of 5 $[min/lot]$ and second machine has a process time of 15 $[min/lot]$). The final experiment uses a model that has stochastic process times with a mean of 5 $[min/lot]$ for the first machine and 15 $[min/lot]$ for the second machine.
In each experiment the four sample schemes are used. The results showed that the controller first starts with emptying the buffers and when the buffers are empty the machine produces with the same rate as the lots arrive, the controller and DEM ends in a steady state position. The results also showed that when the system is controlled using a sample scheme with high (re)optimization frequency, the system is still able reach steady state behavior. When as low frequency sample scheme is used a difference can occur between the DEM and control signal, but the system still reaches a steady state.
This means that when using a hierarchical approach for inventory control and production optimization of a manufacturing line, it is possible to use an event-driven sample scheme.

7.2 Discussion

The research objectives are obtained, but this path to obtain this objective should be critically reviewed.
The first part that should be reviewed is the SCLP algorithm. In the process of understanding the algorithm an attempt was made to program the algorithm, but due to the time limitations of this project this attempt had to be stopped. However, the programming of the algorithm gave a valuable inside in how the algorithm works exactly. So finishing a program of the algorithm is very valuable.
The next step was the translation of the MPC problem into a SCLP problem. This can be achieved in several ways, but not all the translations were solvable by the SCLP solver and the reason why the solver cannot find a solution is unknown. To find out why the translations do not work, a better understanding of the solver is needed. Another factor of the translation is that it is unknown whether this is the ideal way of representing an MPC problem.
The second part that should be reviewed is conversion from the manufacturing line to the controller. This conversion was chosen because it was the easiest to implement, but it is unknown whether this is the best conversion.
The simulation part also has some elements that should be reviewed. In this report a simple model of a manufacturing line with different type of process times was tested and the model was tested by using constant initial values. How the controller would react in more complex system and what the influences of the test strategy is, is unknown. To determine, this further research is needed.
The last part that should be evaluated are the error calculations made in Chapter 6. As mentioned in that chapter the error is caused by the way the measurement are done and the by the timing of the measurement.

7.3 Recommendations

The discussion has created a few questions, for which further research is needed to answer them. This leads to five recommendations.

The first recommendation is to program the algorithm for a better understanding of the algorithm.

The second recommendation is about the investigation of the limitations of the algorithm and an investigation into the best way to operate this algorithm, while keeping in mind the MPC principles. This might create a more transparent model.

The third recommendation is about the conversion from the manufacturing system to the controller. In the analysis of the conversion it is recommended that the conversion method applied in this report is compared to an alternative method using an observer.

The last two recommendations are related to the simulations. It is recommended to simulate a larger and a more complex line like a reentrant line, to find out how the controller would react. When simulating it is advisable that experiments are done by using different initial settings.
The last recommendation is about the use of error plots to determine how many targets are not reached. It is recommended to investigate the correctness of the error measurement used in this report.

Bibliography

[All00] F. Allgöwer and A. Zheng, editors. *Nonlinear Model Predictive Control*, volume 26. Birkhäuser, 2000.

[All01] F. Allgöwer and R. Findeisen, editors. *An Introduction to Nonlinear Model Predictive Control*. Dutch Institute of Systems and Control (DISC), Summerschool on 'The Impact of Optimization in Control', 2001. 3.1–3.45.

[Bla82] J.H. Blackstone, D.T. Philips, and G.L. Hogg. A state-of-the-art survey of dispatching rule for manufacturing job shop operations. *International Journal of Production Research*, 20(1):27–45, 1982.

[Cam99] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 1999.

[Dan68] G.B. Dantzig and M.N. Thapa. *Linear programming and extensions/ by George B. Dantzig*. Princeton University Press, 4 edition, 1968.

[Ess02] H.A. van Essen and M. Steinbuch. *Lecture notes on model predictive control for the course 'Capita Selecta in Control'*. Eindhoven University of Technology, Department of Mechanical Engineering, 2002.

[Fin00] R. Findeisen and F. Allgöwer. Nonlinear model predictive control for index-one DAE systems. In F. Allgower and A. Zheng, editors, *Nonlinear Model Predictive Control*, volume 26 of *Progress in systems and control theory*, pages 145–161. Birkhäuser, 2000.

[Gar89] C.E. Garcia, D.M. Prett, and M. Morari. Model predicitve control: theory an practice — a survey. *Automatica*, 25(3):335–347, 1989.

[Gre93] M.S. Grewal and A.P. Andrews. *Kalman Filtering Theory and Practice*. Prentice Hall, 1993.

[Hof02] A.T. Hofkamp and J.E. Rooda. χ *reference manual*. Eindhoven University of Technology, Department of Mechanical Engineering, System Engineering Group, November 2002. http://se.wtb.tue.nl/documentation.

[Hof03] A.T. Hofkamp. *Python from* χ. Eindhoven University of Technology, Department of Mechanical Engineering, System Engineering Group, χ-0.8 edition, December 2003.

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

[Roo01] J.E. Rooda and J.J.T. Kleijn. *χ Manual*. Eindhoven University of Technology, Department of Mechanical Engineering, System Engineering Group, October 2001. http://se.wtb.tue.nl/documentation.

[Roo03] J.E. Rooda and J. Vervoort. *Analysis of Manufacturing Systems*. Eindhoven University of Technology, Department of Mechanical Engineering, System Engineering Group, February 2003.

[Ste99] A. Sterian. Pymat - an interface between python and matlab, July 1999. http://claymore.engineer.gvsu.edu/~steriana/Python/pymat.html.

[Tsa97] K.S. Tsakalis, J.J. Flores Godoy, and A.A Rodriguez. Hierarchical modeling and control for re-entrant semiconductor industry fabrication lines: A mini-fab benchmark. *In 6^{th} IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA'97),Los Angles, CA*, 1997.

[Van96] R.J. Vanderbei. *Linear Programming foundations and extensions*. Kluwer Academic Publishers, 1996. http://www.princeton.edu/~rvdb/LPbook/index.html.

[Var03] 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):pp. 578–587, July 2003.

[Ver03] J. Vervoort and J.E. Rooda. *Learning χ-0.8*. Eindhoven University of Technology, Department of Mechanical Engineering, System Engineering Group, October 2003. http://se.wtb.tue.nl/documentation/.

[Wei82] L.M. Wein. Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 1(3):115–130, 1982.

[Wei04] G. Weiss. A simplex based algorithm to solve separated continuous linear programs. *submitted to Mathematical Programming*, 2004. http://stat.haifa.ac.il/~gweiss/.

Appendix A

Translation to a primal SCLP

In Chapter 3 a continuous-time MPC problem is formulated. For solving this problem an algorithm is presented in Chapter 4, the SCLP algorithm. The MPC formulation (3.6a) is different from the SCLP formulation. This means that the MPC problem has to be translated, to be able to solve it with the SCLP algorithm. In this translation a choice is made not to include all the parameters of the SCLP-algorithm. The choice is based on the examples Weiss uses about a flowshop and a reentrant line. The translation of the MPC problem is shown below.

The continuous-time MPC problem as described in Chapter 3:

$$\min_{u,x} J = \int_0^T (T-t)\vec{r}'\ \vec{u}(t) + \vec{p}'\ \vec{x}(t) d\tau, \tag{A.1a}$$

s.t. model constraint

$$\begin{aligned} \dot{\vec{x}}(t) &= \mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda}, \\ \vec{y}(t) &= \mathbf{I}\vec{x}(t), \end{aligned} \tag{A.1b}$$

s.t. inequality constraints

$$\begin{aligned} \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \leq \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \geq \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \\ \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \leq \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \geq \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} \end{aligned} \tag{A.1c}$$

The translation can be divided into three parts. The translation of the cost function, the translation of the buffer constraints and finally the translation of the capacity constraints.

A.1 Cost function

The desired cost function is:

$$\max_u J = \int_0^T (T-t)\vec{c}'\ \vec{u}(t)d\tau, \tag{A.2}$$

The continuous-time MPC problem has the following function:

$$\min_{u,x} J = \int_0^T (T-t)\vec{r}'\ \vec{u}(t) + \vec{p}'\ \vec{x}(t)\ d\tau. \tag{A.3}$$

The first difference between the cost functions is that (A.3) minimize the cost and (A.2) is maximization. This is translated by maximizing the negative of the cost function (A.3).

$$\max_{u,x} J = \int_0^T -(T-t)\vec{r'}\ \vec{u}(t) - \vec{p'}\ \vec{x}(t)\ d\tau. \tag{A.4}$$

The next phase of the translation of the cost function is to write (A.4) as a function of $u(t)$. This means that the second term in (A.4) needs to be translated.

$$\int_0^T -\vec{p'}\ \vec{x}(t)d\tau. \tag{A.5}$$

By using partial integration (A.5) can be translated to a function of $u(t)$.

$$\int_0^T -\vec{p'}\ \vec{x}(t)\cdot 1\ d\tau = \tag{A.6a}$$

$$-\vec{p}'\ \vec{x}(t)\cdot t\big|_0^T - \int_0^T -\vec{p}'\ \dot{\vec{x}}(t)\cdot t\ dt =$$

$$-\vec{p'}\ \vec{x}(T)\ T - \int_0^T -t\ \vec{p}'\ (\mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\ \vec{\lambda})dt =$$

$$-\vec{p}'\ \vec{x}(T)\ T - \int_0^T (T-t)\vec{p}'\ (\mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\vec{\lambda})\ dt - \int_0^T T\vec{p}'\ (\mathbf{B}\vec{u}(t) + \bar{\mathbf{B}}\ \vec{\lambda})\ dt =$$

$$-\vec{p'}\ \vec{x}(T)\ T - \int_0^T (T-t)\vec{p'}\ \mathbf{B}\vec{u}(t)\ dt + \int_0^T (T-t)\vec{p'}\ \bar{\mathbf{B}}\ \vec{\lambda}\ dt - T\vec{p'}\ \vec{x}(T) =$$

$$-\int_0^T (T-t)\vec{p'}\ \mathbf{B}\vec{u}(t)\ dt - 2\ T\vec{p'}\ \vec{x}(T). \tag{A.6b}$$

The second term $(-2\ T\vec{p}'\ \vec{x}(T))$ in (A.6b) has a constant value during the optimization. This means that it can be neglected in the cost function. Which result into the following cost function:

$$\begin{aligned}\max_u J &= \int_0^T -(T-t)\vec{r}'\ \vec{u}(t) - \int_0^T (T-t)\vec{p}'\ \mathbf{B}\vec{u}(t)\ d\tau \\ &= \int_0^T (T-t)(-\vec{r'} - \vec{p}'\ \mathbf{B})\vec{u}(t)\ d\tau.\end{aligned} \tag{A.7}$$

With the desired cost function:

$$\max_u J = \int_0^T (T-t)\vec{c}'\ \vec{u}(t)d\tau, \tag{A.8}$$

$$c' = (-\vec{r}' - \vec{p}'\ \mathbf{B}).$$

Constraints

In Chapter 3 a buffer and a capacity constraint are defined. The buffer constraint needs to be rewritten into a constraint that is a function of $u(t)$. This is done by using the model constraints to rewrite $y(t)$:

$$\begin{aligned} \int_0^t \dot{x}(s)ds &= \int_0^t (Bu(s) \ + \ \bar{\mathbf{B}} \ \vec{\lambda}) \ ds, \\ x(t) &= \int_0^t Bu(s) \ ds + x(0) \ + \ \bar{\mathbf{B}} \ \vec{\lambda} \ t, \end{aligned} \tag{A.9a}$$

$$\begin{aligned} \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix} &= \int_0^t \mathbf{IB}\vec{u}(s) \ ds \ + \ \bar{\mathbf{B}} \ \vec{\lambda} \ t \ + \ \vec{x}(0), \\ &= \int_0^t \mathbf{B}\vec{u}(s) \ ds \ + \ \bar{\mathbf{B}} \ \vec{\lambda} \ t \ + \ \vec{x}(0). \end{aligned}$$

$$\vec{0} \leq \int_0^t \mathbf{B}\vec{u}(s) \ ds \ + \ \bar{\mathbf{B}} \ \vec{\lambda} \ t \ + \ \vec{x}(0) \leq \vec{\mathcal{Y}}_{max} \tag{A.9b}$$

Equation (A.9b) can also be written as:

$$\begin{aligned} \int_0^t -\mathbf{B}\vec{u}(s) \ ds \ &\leq \ \vec{x}(0) \ + \ \bar{\mathbf{B}} \ \vec{\lambda} \ t \\ \int_0^t \ \mathbf{B}\vec{u}(s) \ ds \ &\leq (\vec{\mathcal{Y}}_{max} - \ \vec{x}(0)) \ - \ \bar{\mathbf{B}} \ \vec{\lambda} \ t \end{aligned} \tag{A.10a}$$

This creates the desired buffer constraint function.

$$\begin{aligned} &\int_0^t \mathbf{G}\vec{u}(s) \ ds \ \leq \ \vec{\alpha} \ + \ \vec{a} \ t \\ &\mathbf{G} = \begin{pmatrix} -\mathbf{B} \\ \mathbf{B} \end{pmatrix} \qquad \vec{\alpha} = \begin{pmatrix} \vec{x}(0) \\ \vec{\mathcal{Y}}_{max} - \ \vec{x}(0) \end{pmatrix} \qquad \vec{a} = \begin{pmatrix} \bar{\mathbf{B}} \ \vec{\lambda} \\ -\bar{\mathbf{B}} \ \vec{\lambda} \end{pmatrix} \end{aligned} \tag{A.10b}$$

The capacity constraint can be rewritten into, with the notion that the SCLP algorithm states that $\vec{u}(t) \geq 0$ and $\mathbf{H} \geq 0$:

$$\begin{aligned} &\mathbf{H}\vec{u}(t) \leq \vec{b}. \\ &\vec{b} = \vec{\mathcal{M}} \\ &\mathbf{H} = \mathbf{R} \end{aligned} \tag{A.11}$$

All these translations together result into an SCLP format as presented in Chapter 4 (in these equation $\vec{x}(t)$ is replaced by $\vec{z}(t)$).

$$\begin{aligned} \max \quad & \int_0^T (\vec{\gamma}\,' + (T-t)\vec{c}\,'\vec{u}(t) + \vec{d}\,'\vec{z}(t))dt, \\ s.t. \quad & \int_0^t G\vec{u}(s)ds + F\vec{z}(t) \leq \vec{\alpha} + \vec{a}t, \\ & H\vec{u}(t) \leq \vec{b}, \\ & \vec{z}(t), \vec{u}(t) \geq \vec{0}, \qquad t \in [0,T]. \end{aligned} \tag{A.12}$$

$$\begin{aligned} \min \quad & \int_0^T (\vec{\alpha}\,' + (T-t)\vec{a}\,')\vec{u}(t) + \vec{b}\,'\vec{z}(t))dt, \\ s.t. \quad & \int_0^t G'\vec{u}(s)ds + H'\vec{z}(t) \leq \vec{\gamma} + \vec{c}t, \\ & F'\vec{u}(t) \leq \vec{d}, \\ & \vec{z}(t), \vec{u}(t) \geq \vec{0}, \qquad t \in [0,T]. \end{aligned} \tag{A.13}$$

With:

$$\begin{aligned} &\vec{\gamma} = \vec{0}, && \vec{c} = (-\vec{r}\,' - \vec{p}\,'\ \mathbf{B}), && \vec{d} = (\quad), \\ &G = \begin{pmatrix} \mathbf{B} \\ -\mathbf{B} \end{pmatrix}, && F = (\quad), && \vec{\alpha} = \begin{pmatrix} \vec{x}(0) \\ \vec{x}_m - \vec{x}(0) \end{pmatrix}, && \vec{a} = \begin{pmatrix} \bar{\mathbf{B}}\ \vec{\lambda} \\ -\bar{\mathbf{B}}\ \vec{\lambda} \end{pmatrix}, \\ &\mathbf{H} = \mathbf{R}, && \vec{b} = \vec{\mathcal{M}}. \end{aligned}$$

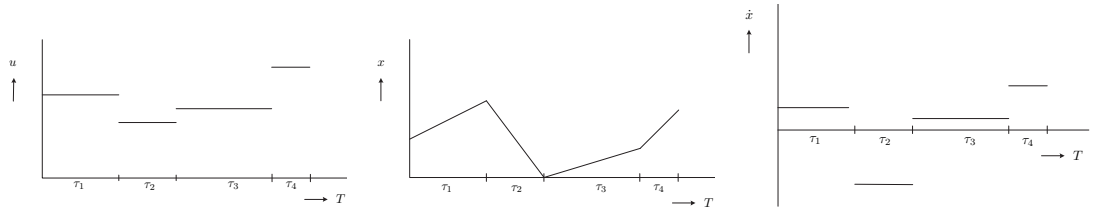
Appendix B

Explaining the SCLP algorithm

In Chapter 4 a algorithm was presented to solve a continuous-time MPC problem, the SCLP algorithm [Wei04]. It showed the rules of the algorithm. This appendix explains the algorithm graphically by showing some of the steps the algorithm makes. By doing so it should create a better understanding of the algorithm.

B.1 The SCLP Algorithm

To understanding how the algorithm finds the optimal solution it is important to understand the mean principle of this algorithm. This principle is stated in Theorem 1. It says that the solutions should be feasible and bounded and that the SCLP/SCLP* problems have a complementary slack optimal primal and dual solution. This means that there is no duality gap between the solutions of SCLP and SCLP*, strong duality hold. The theorem also states that the control variables $(\vec{u}(t), \vec{p}(t))$ are piecewise constant and that the state variables $(\vec{x}(t), \vec{q}(t))$ are continuous piecewise linear. This is shown in Figure B.1, which shows the variables $\vec{u}(t), \vec{x}(t), \dot{\vec{x}}$ as a function of time.



(a) *u piecewise constant,* (b) *x piecewise linear.* (c) *ẋ piecewise constant.*

Figure B.1: *The piecewise constant property of an SCLP problem*

By using this property, that the variables $\vec{u}(t), \vec{p}(t), \dot{\vec{x}}, \dot{\vec{q}}$ of an SCLP/SCLP* are piecewise constant, the time horizon can be divided into N intervals. Which means that when an optimal solution is found at the boundary of the problem it is also the solution for the first interval. And because the variables are piecewise constant only one optimal solution can be

found per interval. This means that the problem can be solved $0 \dots T$ by starting at $T = 0$ and letting T increase. As T is increasing the algorithm is able to move iteratively from one optimal solutions to another in the next interval.

The process of solving an SCLP problem is shown in Figure B.2. Vertically it shows the time horizon, from $T = 0$ to $T = \inf$, of the solution to the SCLP problem and horizontally the increase of time, going from $t = 0$ to $t = \inf$.

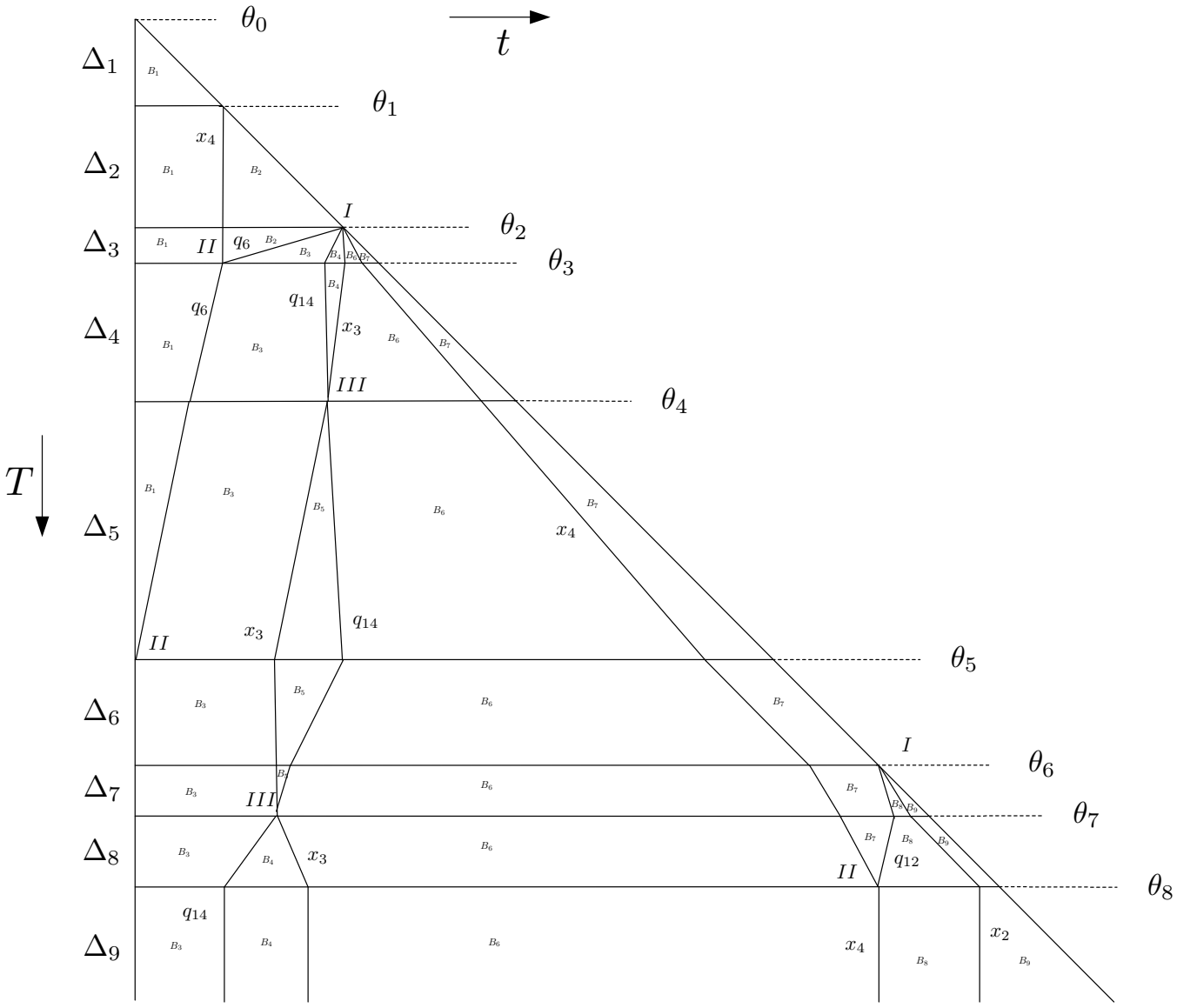


Figure B.2: *An example of the iterations of the SCLP algorithm*

The figure shows, a problem that has eight intervals, $\theta_0 - \theta_8$, with the length of each intervals represented by Δ_i $i = 1, \dots, 9$. The last interval Δ_9 is from $\theta_8 \rightarrow \inf$.

At θ_0 ($T = 0$) the SCLP solver starts with the optimal solution of the boundary of the SCLP problem. This is found by solving the LP optimization, Boundary-LP/LP* (4.5,4.6) and rates-LP/LP*($\mathcal{K}_0, \mathcal{J}_N$) (4.9,4.10). The solution to these optimizations are the base-sequence (B_1 or B_1*). These base-sequences are sets of variables consisting out of the variables u_j, x_k respectively p_k, q_j or a set of their indexes. The base-sequences have a range from $B_1, \ldots, B_N$ or $B_N^*, \ldots, B_1^*$. The complementary slackness (see C.2) makes it possible to find the complementary dual basis $B*$ if the primal basis is known or the other way around, find de primal basis when the dual is known. This is done by using the following rules $\dot{x} \in B \Rightarrow p_k \notin B^*$, $\dot{q}_j \in B^* \Rightarrow u_j \notin B$.

The first base-sequences is valid for the neighborhood around $T = 0$ and $T = N$. At θ_1 the base-sequences are no longer optimal, which ends the first interval. The new optimal solution is found by performing a pivot operation, which changes the bounds of the rates-LP/LP* and so creating new base-sequences D. The pivot operation makes it possible to go from one optimal base-sequence to another optimal base-sequence. This is done by removing one variable for the first base-sequences B' (the leaving variable) and to place the a new variable into the second base-sequences B'' (the entering variable). In the Figure B.2 the pivot operations are represented by the vertical lines with x_k or q_j next to it. It should be noted that when a variable leaves a base-sequences it can enter an other neighboring base-sequence and at a later point in time.

When the new base-sequence D is adjacent to B' (adjacent is when $|B' \setminus D| = 1$) the neighboring base-sequences is found. But when $|B' \setminus D| > 1$ a subproblem is need to find the optimal sequences of basis. The situations were a subproblem are needed are represented by points I in the Figure B.2.

Figure B.2 shows that the set of base-sequences changes al long the time horizon. Base-sequences can disappear from the optimal solution, when $|B| = 0$ (see point II). And in point III one base-sequences disappears and another appears. The Base-sequences that appears has been empty up till that point in time.

B.2 Validity regions

A base-sequence is valid for interval of Δ, this intervals is also called the validity region. In Figure B.2 shows that the length of an interval varies. To determine the length of a validity region, one need to check when a solution is no longer feasible. This is done by checking when a constraint is violated. Since the control variables are constant, the state variable determines which variable causes a constraint violation. This means that only the constraints $F\vec{x}(t) \leq \vec{\alpha}$ and $H'\vec{q}(t) \leq \vec{\gamma}$ and these are the same as the boundary-LP/LP*. So the constraints are violated when the values for $\vec{x}, \vec{q}$ are larger than the boundary values ($\vec{x}^{\,0}, \vec{q}^{\,n}$). This means that the boundary values minus the values for $\vec{x}(t)$ and $\vec{q}(t)$ have to be larger than zero. This is graphically shown in Figure B.3. Where the course of the control (u) and state (x) variables is shown (Figure B.3(b)) and the position on the time horizon (Figure B.3(a)).

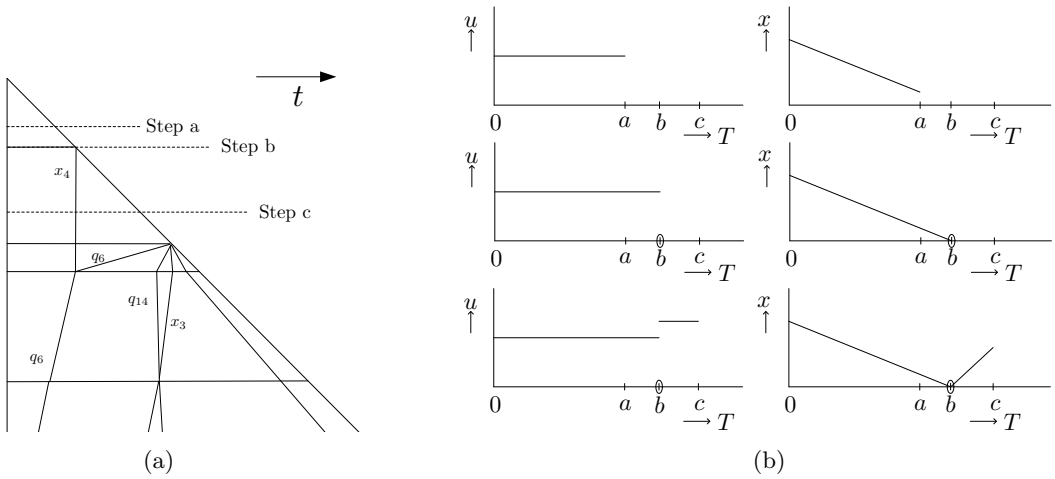


Figure B.3: *Determining the interval length*

After the validity region has been calculated a pivot step is performed and a new optimal base-sequence is determined. The next step is to determine how long this solution is optimal. The value of x or q at the start of new base-sequence is the end values of the old base-sequence. This means that the value of the state variables can be plotted as shown in Figure B.4. The rates calculated for a base-sequence do not change as the time horizon is increased. But it is possible that the length of an old interval changes due to the new base-sequence, because a different state causes a violation of the boundary constraints at an earlier point in time.

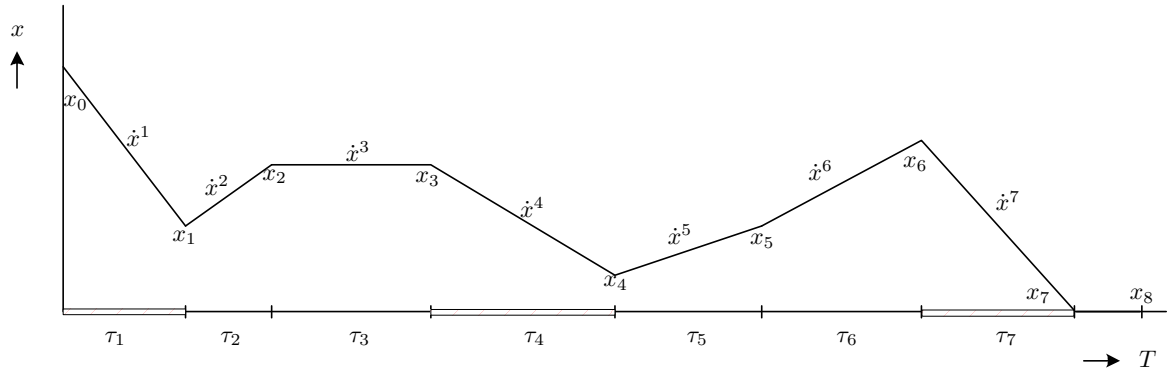


Figure B.4: *The path of a state variable*

Reduction of the number validity regions of importance

The equations used to determine the length of the validity region use all regions, but not all of them are needed. In fact at every breakpoint (the intersection of two regions) exactly

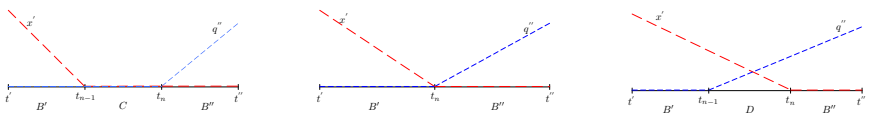
one event happens, either x_k hits zero or q_j hits zero. This means to determine the length of an interval, not all the validity regions are needed. Only the region(s) where $\dot{x}, \dot{q} < 0$ are of interest for determining the length of the validity region. In Figure B.4 the intervals of interest are marked. By only checking the regions where $\dot{x}, \dot{q} < 0$ the size of the problem (4.14) can be reduced.

B.3 Pivoting

Pivoting is one the most import elements of the SCLP algorithm. As mentioned earlier the pivoting operation is needed to move from one optimal base-sequence to the next neighboring optimal base. The common boundary between these neighboring basis is called $\theta = 0$. The optimal base-sequence on the left side of the boundary is valid for $\theta < 0$ and the base-sequence on the right side is valid for $\theta > 0$. The SCLP algorithm has four mean types of pivoting operations called case i-iv, with some having a sub groups. These cases are graphically shown in the Figures B.5-B.8.

Figure B.5 shows the three situations ($\theta < 0$, $\theta = 0$ and $\theta > 0$) for case ii, with a simple pivot. The first situation is Figure B.5(a), it shows the optimal base-sequences is B', C, B'' with the intervals (t', t_{n-1}),(t_{n-1}, t_n) and (t_n, t''). The variable that leave B' is $B' \setminus C = \dot{x}_{,}$, with $x_{,}(t) > 0$ for $t' < t < t_{n-1}$ and $x_{,}(t) = 0$ during $t_{n-1} < t < t''$. And for C the leaving variable is $C \setminus B'' = u_{,,}$, with $q_{,,}(T - t) > 0$ for $t_n < t < t''$ and $q_{,,}(T - t) = 0$ during $t' < t < t_{n-1}$. The second situation is for $\theta = 0$, shown in Figure B.7(b). It shows the boundary between the validity region of B' and B''. Where the time interval $t_{n-1} < t < t_n$ ($\tau_n = t_n - t_{n-1}$) has shrunk to zero. The final situation is for the optimal base-sequence B', D, B'', which is valid for $\theta > 0$, shown in Figure B.5(c). The intervals are the same as in B.5(a), but $B' \setminus D = u_{,,}$ and $D \setminus B'' = \dot{x}_{,}$. Such that $x_{,}(t) > 0$ for $t' < t < t_n$ and $q_{,,}(T - t) = 0$ for $t_{n-1} < t < t''$. And in the interval (t_{n-1}, t_n) both are > 0.

For this pivot operations the collision occurred at time t_n, the collision is classified as a case ii simple collision. This is valid for both $\theta \nearrow 0$ and $\theta \searrow 0$. This means that when a optimal solution is known for the region $\theta < 0$ the solution for $\theta > 0$ can be constructed, by discarding C and inserting D.

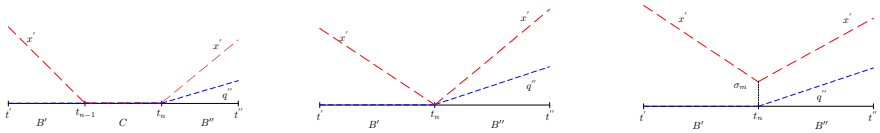


(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.5: *A simple collision case ii, with a simple pivot*

The second type of pivot operation is a Simple collision case i/iii, shown in Figure B.6. It is similar to the previous case, for $\theta < 0$ (Figure B.6(a)) the optimal basis are B', C, B'' in the intervals (t', t_{n-1}),(t_{n-1}, t_n) and (t_n, t''). Where $B' \setminus C = \dot{x}_{,}$ and $C \setminus B'' = u_{,,}$, but in this collision case the variable that leaves during the pivot operation $B' \longrightarrow C$ ($\dot{x}_{,}$) enters the

basis again in the pivot $C \longrightarrow B''$. This means that $x_{'}(t) > 0$ for $t' < t < t_{n-1}$, $x_{,}(t) = 0$ for $t_{n-1} < t < t_n$ and finally $x_{,}(t) > 0$ for $t_n < t < t''$. For the situation that $\theta = 0$ (Figure B.6(c)) the interval τ_n also shrinks to zero, where $B' \setminus B'' = u_{,,}$ (B' and B'' are adjacent) and the state variable $x_{,}$ has a strict local minimum of 0 at t_n. And the final situation $\theta > 0$, the interval (t_{n-1}, t_n) remains of length zero, with the optimal basis B',B'' for (t', t_n),(t_n, t''). And the state variable $x_{,}(t)$ has a strict local minimum at t_n, with a value $\sigma = x_{,}(t_n) > 0$. The collisions in these cases occur at t_n and are classified as a case i, simple collision, when $\theta \nearrow 0$ and as a case iii, simple collision, when $\theta \searrow 0$. This means that when the optimal solution for the region $\theta < 0$ is known the solution for $\theta > 0$ can be constructed by discarding the basis C. In case the optimal solution is known for the region $\theta > 0$, the solution for $\theta < 0$ can be constructed by inserting the basis C between B', B''.

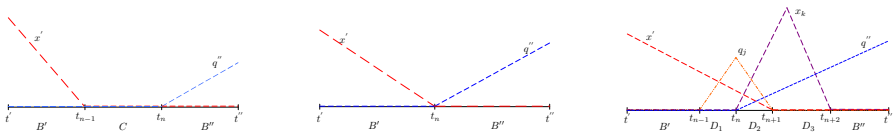


(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.6: *Simple collisions case i/iii, with a simple pivot*

The third type of pivot operations is described in Figure B.7. This type of collision is more complicated. The first two parts ($\theta < 0, \theta = 0$) are the same as in Figure (B.5(a),B.7(b)), but the solution for the region $\theta > 0$ (see Figure B.7(c)) is different. The optimal base-sequence is $B', D1, D2, D3, B''$, with $B' \setminus B'' = \{\dot{x}_{'}, u_{,,}\}$, but the basis B' and B'' are separated by several intervals. These intervals all shrink to zero when $\theta \searrow 0$. This means that $x_{,}(t) > 0$ for $t' < t < t_{n+1}$ and $q_{,,}(T-t) > 0$ for $t_n < t < t''$, while the other state variables $x_k(t), q_j(T-t))$ that appear in this figure are zero outside the region of (t_{n-1}, t_{n+2}). The base-sequences are link as follow: $B' \setminus D_1 = u_j$, $D_1 \setminus D_2 = u_{,,}$, $D_2 \setminus D_3 = \dot{x}_{,}$ and $B_3 \setminus B'' = \dot{x}_k$.

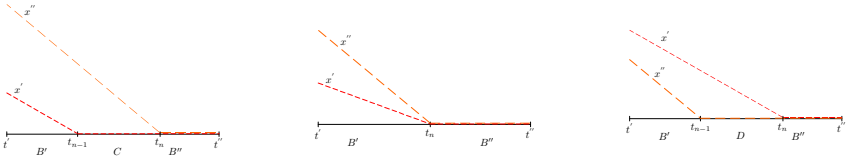
When $\theta \searrow 0$ this collision is called a compound collision (case ii). So if the solution of the region $\theta > 0$ is known, the solution for $\theta < 0$ can be constructed by erasing the bases $D1, D2, D3$ and inserting basis C. But in opposite directions, the solution of $\theta > 0$ is known and the solution of $\theta < 0$ is needed, the basis C can be erased and the bases $D1, D2, D3$ need to be constructed and inserted between the bases B' and B''. The bases $D1, D2, D3$ is obtained by solving a subproblem, the solution to this subproblem is part of the pivot operation.



(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

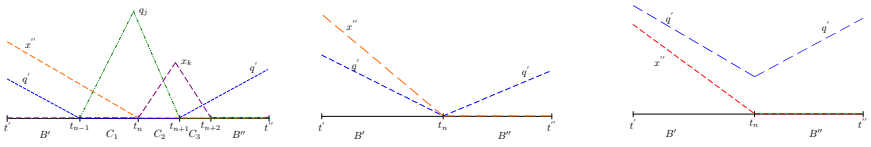
Figure B.7: *Simple collisions case ii, pivot requires solution of a subproblem*

In the next three figures (Figure B.8-B.10), different examples of a single collision and pivot at $0 < t < T$ are shown. They can be interpreted in the same way as Figures B.5-B.7.



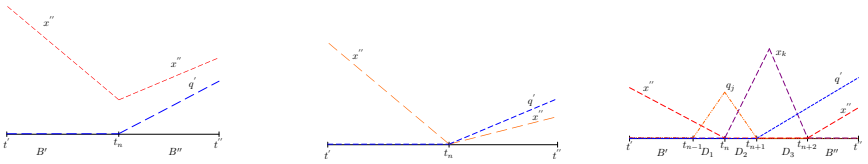
(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.8: *Simple collisions case ii, simple pivot* $v' = \dot{x}', v'' = \dot{x}''$



(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

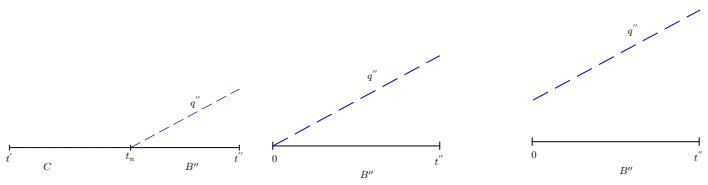
Figure B.9: *Collisions case i, serval intervals shrink to* 0, $v' = u', v'' = u''$



(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

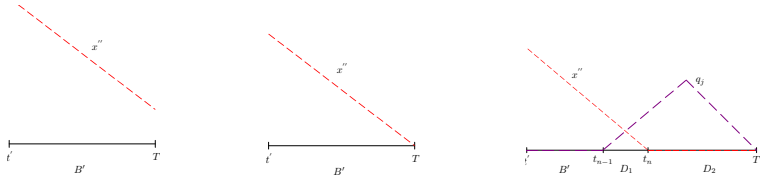
Figure B.10: *Collisions case iii, with a subproblem* $v' = u', v'' = \dot{x}''$

There are also single collisions at $t = 0$ or $t = T$, related to the cases $\mathrm{i}_a, \mathrm{iii}_a, \mathrm{i}_b, \mathrm{iii}_b, \mathrm{iv}_a$ and iv_b. These are shown in Figures B.11-B.13.



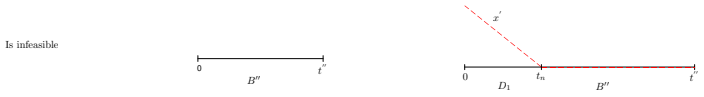
(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.11: *Collisions case* i_a *interval shrinks to zero at time 0*



(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.12: *Collisions case iii_b slack hits zero at time T, with subproblem $v'' = \dot{x}''$*



(a) $\theta < 0$ (b) $\theta = 0$ (c) $\theta > 0$

Figure B.13: *Collisions case iii_b slack hits zero at time T, with subproblem $v'' = \dot{x}''$*

B.4 Subproblem

In case of a Compound collision $B' \setminus B'' > 1$, this means that B' is not adjacent to B'', a subproblem is needed to obtain the sequence of bases. The subproblem is actually a smaller SCLP problem, with a subset of the variables of the main problem. By removing the variables which are zeros from primal and dual set of variables. The subproblem is solved for the boundary values $l(\theta)$, with $0 < \theta < 1$. At $\theta = 0$ the optimal base-sequences D, this is the base-sequences found by the rates-LP/LP*, and at $\theta = 1$ the optimal base-sequences are B', B''. This is graphically described in Figure B.14.

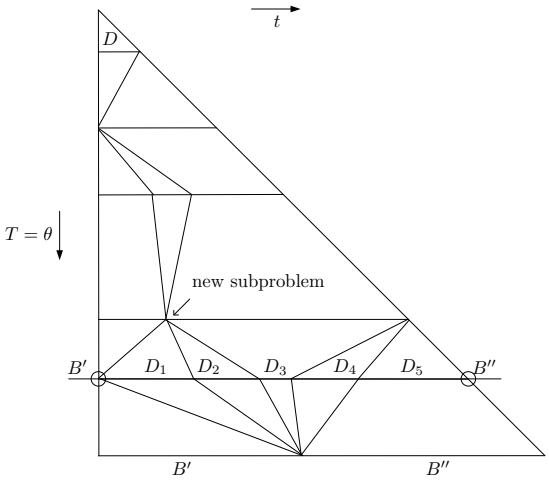


Figure B.14: *The iteration of a subproblem*

The first step in solving a subproblem is to make an initial pivot. At $\theta = 0$ the initial base-sequences is D. By performing a pivot operation according to case iii_a or iv_a to obtain the basis D_-. To obtain the basis D_+ the pivot operation according to case iii_b or iv_b has to be performed. When D_- or D_+ are not adjacent to D, a new subproblem needs to be solved to find the sequence of bases $D_{-M'}, \ldots, D_{-1}$ respectively $D_1, \ldots, D_{M''}$.

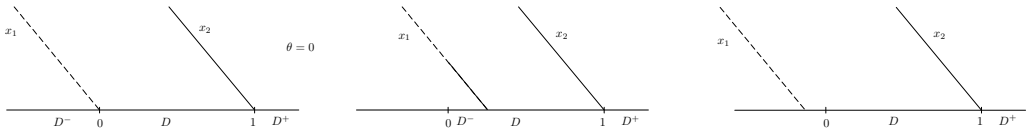


Figure B.15: *Initial solutions of the subproblem, for* $\theta > 0$

The next step is to determine the initial solution. Depending on the situation (D_-, D), (D_-, D, D_+) or (D, D_+) a valid adjacent base is chosen for a neighborhood of $\theta > 0$. If the new base are not adjacent a new subproblem(s) need to be defined. When a neighboring base is found. The validity region of the base-sequence needs to be determined. The next step is to check whether the base-sequence contains all the variable of the main problem, if not, a pivot step is made and a new base-sequence is determined. When all variables of the main problem are part of the base-sequence the excluded variable are re-introduced and the algorithm returns to the main problem.

Appendix C

LP theory

Linear Programming theory is a theory that is widely used. In this report this theory is used to solve an SCLP problem. For better understanding the SCLP algorithm, an appendix is made that contains some of the basics of Linear Programming theory. This appendix contains the theory and a small example. For a more detailed explanation of the LP theory see [Van96, Dan68].

To explain the LP-theory the following standard form of a linear programming problem is used:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j, \\ s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \ \leq \ b_i, \\ & x_j \ \geq \ 0. \qquad j = 1, 2, \ldots, n \qquad i = 1, 2, \ldots, m \end{aligned} \tag{C.1}$$

C.1 The Simplex Method

The simplex method is a method to find the optimal solution of the LP problem (C.1)(or it determines whether or not the optimal solution exists). The simplex method does this by rewriting the inequality constraints into equality constraints (C.2) and by introducing extra variables (the slack variables). (The notation of the slack variables can be less indistinguishable by just adding them to the end of the list of x-variables. ($\{1, 2, \ldots, n + M\}$))

$$\begin{aligned} \zeta &= \sum_{j=1}^{n} c_j x_j, \\ w_i &= b_i - \sum_{j=1}^{n} a_{ij} x_j, \\ & x_j, \ w_1 \geq 0. \qquad j = 1, 2, \ldots, n \qquad i = 1, 2, \ldots, m \end{aligned} \tag{C.2}$$

The system of equations in (C.2) is called a dictionary. The variables on the left are called the basic variables, with the exception of ζ and the variables on the right are the nonbasic variables. The solution found by setting the nonbasic variables to zero is called the basic feasible solution.

Pivoting

To solve the LP problem (C.1), the equations are transformed into the starting dictionary as mentioned in (C.2). The optimal solution is found by searching iteratively for it. This is done by starting with a feasible solution, for example the basic feasible solution, and determining the value for the objective function ζ. The next step is to change the dictionary in such a way that the value of ζ is increased. In each iteration only one variable changes from basic to a nonbasic variable. This is called pivoting. The rules for making this choice are called pivot rules.

Each dictionary has m basic variables and n nonbasic variables. B denotes the indices of basic variables and $\mathcal{N}$ the nonbasic variables. For the first dictionary $\mathcal{N} = \{1, 2, \ldots, n\}$ and $B = \{n+1, n+2, \ldots, n+m\}$. When a variable goes from a nonbasic to basic variable it is called an *entering variable*. The other way around it is called a *leaving variable*.

The choice of which variable enters the dictionary is determined with the aim to increase ζ. This means that $\{j \in \mathcal{N} : c_j > 0\}$ has to be valid. If this set is empty then the current solution is optimal. If the set is larger than one, it suffices to say that the largest value of c_j is usually picked. When the values are equal the variable which creates the largest one-step increase of the objective function is chosen.

After the entering variable is selected (x_k), the leaving variable is chosen by changing x_k from zero to a positive value. This increases the values of the basic variables.

$$x_i = b_i - a_{ik}x_k, \qquad i \in B \tag{C.3}$$

The basic value have to be nonnegative, so it is required that

$$b_i - a_{ik}x_k \geq 0, \qquad i \in B. \tag{C.4}$$

As x_k increases the expression in (C.4) can only go negative, when the values of a_{ik} are positive. The remaining expression remain fixed or increase. To determine which value leaves, the attention can be restricted to those i for which a_{ik} is positive. The next step is to check for which values of x_k the expressions becomes zero

$$b_i - a_{ik}x_k = 0, \tag{C.5}$$

$$x_k = \frac{b_i}{a_{ik}}. \tag{C.6}$$

Since non of the basic variable may become negative, x_k can only be raised to the smallest of the values found in (C.6)

$$x_k = \min_{i \in B : a_{ik} > 0} \frac{b_i}{a_{ik}}. \tag{C.7}$$

Another method for finding the leaving variable is by rewriting (C.4) to:

$$\frac{1}{x_k} \geq \frac{a_{ik}}{b_i}, \qquad i \in B. \tag{C.8}$$

This method use the convention that $\frac{0}{0} = 0$. To find the leaving variable, the largest possible increase of x_k is needed. This is equivalent to:

$$x_k = (\max_{i \in B} \frac{a_{ik}}{b_i})^{-1}. \tag{C.9}$$

Degeneracy and Non-degeneracy

A dictionary is called *degenerate* if b_i vanishes for some $i \in \mathcal{B}$. (If $b_i = 0$ for at least one i.) When a dictionary is degenerate it produces a degenerate pivot. A pivot is degenerate, when one of the ratios of the leaving variables is $+\infty$. In case that a dictionary is *non-degenerate* it means that non of the pivot operations become ∞.

When a degenerate dictionary is reached, one or more of the subsequent pivots are degenerate, but eventually a non-degenerate pivot breaks away from the degenerate dictionary. This does not always happen. It is also possible that after a sequence of degenerate pivots the dictionary returns to a dictionary that has appeared before. In this case the simplex method enters into an infinite loop and never finds the optimal solution. This behavior is called cycling.

C.2 Duality theory

Duality theory states that for every linear program (primal), another program exists, called its dual. This aspect is very import in the LP theory, because by determining the dual of the primal problem it is possible to find the optimal solution faster. The feasible solution of the dual gives a bound on the optimal objective function value of the primal problem and the primal feasible solution is a bound for the dual problem.
The standard form for primal and dual LP problem is:

Primal

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j, && \text{(C.10)} \\ s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \\ & x_j \geq 0, \quad j = 1, 2, \ldots, n \\ & \qquad\quad i = 1, 2, \ldots, m \end{aligned}$$

Dual

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} b_i x_j, && \text{(C.11)} \\ s.t. \quad & \sum_{j=1}^{m} y_i a_{ij} \geq c_j, \\ & y_i \geq 0, \quad j = 1, 2, \ldots, n \\ & \qquad\quad i = 1, 2, \ldots, m \end{aligned}$$

For finding the dual problem, the primal problem is rewritten as mentioned in C.1. And by taking the negative transpose of the primal dictionary the dual dictionary can be found:

Primal LP problem

$$\begin{aligned} \max \quad & x_1 + x_2, \qquad (C.12)\\ s.t. \quad & x_1 \leq 1,\\ & x_2 \leq 2,\\ & 3x_1 + x_2 \leq 6,\\ & x_1,\ x_2 \geq 0, \end{aligned}$$

Primal dictionary

$$\begin{aligned} \zeta &= x_1 + x_2, \qquad (C.13)\\ w_1 &= 1 - x_1,\\ w_2 &= 2 - x_2,\\ w_3 &= 6 - 3x_1 - x_2,\\ & x_1, x_2, w_1, w_2, w_3 \geq 0, \end{aligned}$$

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & -1 & 0 \\ 2 & 0 & -2 \\ 6 & -3 & -1 \end{pmatrix} \overset{Negative\ transpose}{\Longleftrightarrow} \begin{pmatrix} 0 & -1 & -2 & -6 \\ -1 & 1 & 2 & 3 \\ -1 & 0 & 2 & 1 \end{pmatrix}, \qquad (C.14)$$

Dual dictionary

$$\begin{aligned} -\xi &= -y_1 - 2y_2 - 6y_3, \qquad (C.15)\\ z_1 &= -1 + y_1 + 2y_2 + 3y_1,\\ z_2 &= -1 + 2y_2 + y_1,\\ & y_1,\ y_2,\ y_3,\ z_1,\ z_2 \geq 0. \end{aligned}$$

By using this quality of the duality theory, it is possible to solve both primal and dual problem at the same time. This is done by using the simplex method on the primal problem and at the same time perform the analogous pivot on the dual problem. When this technique is used it does not mean that both primal and dual problem have to be feasible dictionaries. Only when the primal optimal dictionary is found, the corresponding dual dictionary is also be feasible.

As mentioned above, the duality theory determines the bound of the primal/dual LP problem. The quality of the bound can be determined by the *weak* and *strong duality theorems*. The weak duality theorem states that when the primal feasible solution is $(x_1, x_2, \ldots, x_n)$ and the dual feasible solution is $(y_1, y_2, \ldots, y_m)$ and

$$\sum_j c_j x_j \leq \sum_i b_i y_i \qquad (C.16)$$

holds, it is a *weak duality* problem. And that the set of the primal values lies completely to the left of the set of dual values. When the right endpoint of the primal set is next to the left endpoint of dual set, there is no gap between the optimal objective functions values. Otherwise there is a duality gap, which means that the optimal solution cannot be verified.

If the optimal set of the primal $(x_1^*, x_2^*, \ldots, x_n^*)$ and dual $(y_1^*, y_2^*, \ldots, y_m^*)$ problem are:

$$\sum_j c_j x_j^* = \sum_i b_i y_i^*. \qquad (C.17)$$

The solution of the problem is then referred as a *strong duality* problem, which means that there is no duality gap and the optimal solution can be verified.

In some case the dual optimal solution is needed, but only the primal optimal solution is know. In that case the *Complementary Slackness* Theorem can be used (cf. [**?**, Chapter 5.6]). The *Complementary Slackness* Theorem states that when $x = (x_1, x_2, \ldots, x_n)$ is primal feasible and $y = (y_1, y_2, \ldots, y_m)$ is dual feasible. $(w_1, w_2, \ldots, w_m)$ denotes the primal slack variables and $(z_1, z_2, \ldots, z_n)$ are the dual slack variables, then x and y are optimal for their respective problems if and only if

$$x_j z_j = 0 \qquad \text{for } j = 1, 2, \ldots, n, \tag{C.18}$$

$$w_i y_i = 0 \qquad \text{for } i = 1, 2, \ldots, m.$$

C.3 An example of how to solve an LP problem

The first step in solving an LP problem is by introducing slack variables and by determining the dual problem.

Primal

$$\begin{array}{lll} \max & x_1 + x_2, & (C.19) \\ s.t. & x_1 + 2x_2 \leq 2, & \\ & 2x_1 + x_2 \leq 3, & \\ & x_1,\ x_2 \geq 0. & \end{array}$$

Primal dictionary

$$\begin{array}{ll} \zeta = x_1 + x_2, & (C.20) \\ w_1 = 2 - x_1 - 2x_2, & \\ w_2 = 3 - 2x_1 - x_2, & \\ x_1,\ x_2,\ w_1,\ w_2 \geq 0. & \end{array}$$

Dual problem

$$\begin{array}{lll} \min & 2y_1 + 3y_2, & (C.21) \\ s.t. & y_1 + 2y_2 \geq 1, & \\ & 2y_1 + y_2 \geq 1, & \\ & y_1,\ y_2 \geq 0. & \end{array}$$

Dual Dictionary

$$\begin{array}{ll} -\xi = -2y_1 - 3y_2, & (C.22) \\ z_1 = -1 + y_1 + 2y_2, & \\ z_2 = -1 + 2y_1 + y_2, & \\ y_1,\ y_2,\ z_1,\ z_2 \geq 0. & \end{array}$$

Step 1:

The primal basic solution is: $x_1 = 0,\ x_2 = 0,\ w_1 = 2,\ w_2 = 3 \rightarrow \zeta = 0$
The dual basic solution is: $y_1 = 0,\ y_2 = 0,\ z_1 = -1,\ z_2 = -1 \rightarrow -\xi = 0$

In Figure C.1 the constraints of the primal problem are shown graphically and the primal basic solution is represented by a $*$.

The solution of the primal basic problem is feasible.

$$\begin{array}{ll} B_1 = \{w_1, w_2\} & i = 1,\ 2, \\ \mathcal{N}_1 = \{x_1, x_2\} & j = 1,\ 2. \end{array}$$

For making the choice which variable leaves or enter the dictionary, the following equations are needed:

$$\{j \in \mathcal{N} : c_j > 0\} \qquad \Rightarrow \{x_1, x_2\} \qquad k = 1,\ 2, \tag{C.23}$$

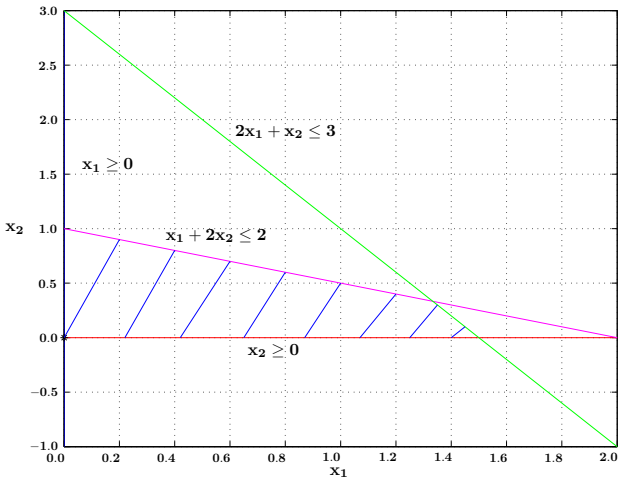


Figure C.1: *A graphical representation of the constraints of the primal problem, with solution ($x_1 = 0, x_2 = 0$)*

In this case (C.23) has two possible variables. And the values (c_j) are equal, so the leaving variable is determined by checking which variable creates the largest one-step increase of the objective value. The answer to that question is x_1. This means that $x_1(k = 1)$ is the leaving variable for the primal problem and z_1 for the dual problem.

The entering variable is determined by calculating (C.6):

$$x_1 \;=\; \min_{i \in B: a_{ik} > 0} = \{\frac{b_1}{a_{11}} \;=\; \frac{2}{1}\;,\; \frac{b_2}{a_{21}} \;=\; \frac{3}{2}\} \;=\; \min\{2\;,\; 1\frac{1}{2}\} \qquad \Rightarrow i = 2, \tag{C.24}$$

The entering variable is w_2 for the primal problem and y_2 for the dual problem.

Rewriting w_2 results in:

$$x_1 \;=\; \frac{3}{2} - \frac{1}{2}w_2 - \frac{1}{2}x_2,$$

The new primal and dual dictionary are:

$$\text{(P)} \qquad \begin{aligned} \zeta &= \frac{3}{2} - \frac{1}{2}w_2 - \frac{1}{2}x_2, \qquad \text{(C.25)} \\ w_1 &= \frac{1}{2} + \frac{1}{2}w_2 - 1\frac{1}{2}x_2, \\ x_1 &= \frac{3}{2} - \frac{1}{2}w_2 - \frac{1}{2}x_2, \end{aligned} \qquad \text{(D)} \qquad \begin{aligned} -\xi &= -\frac{3}{2} - \frac{1}{2}y_1 - \frac{3}{2}z_2, \qquad \text{(C.26)} \\ y_2 &= \frac{1}{2} - \frac{1}{2}y_1 + \frac{1}{2}z_2, \\ z_2 &= -\frac{1}{2} + 1\frac{1}{2}y_1 + \frac{1}{2}z_2. \end{aligned}$$

Step 2:

The primal basic solution is: $x_1 = \frac{3}{2},\ x_2 = 0,\ w_1 = \frac{1}{2},\ w_2 = 0 \rightarrow \quad \zeta = \frac{3}{2}$
The dual basic solution is: $\quad y_1 = 0,\ y_2 = \frac{1}{2},\ z_1 = 0,\ z_2 = -\frac{1}{2} \rightarrow -\xi = -\frac{3}{2}$

In Figure C.2 the basic solution of step 2 is shown in combination with the constraints

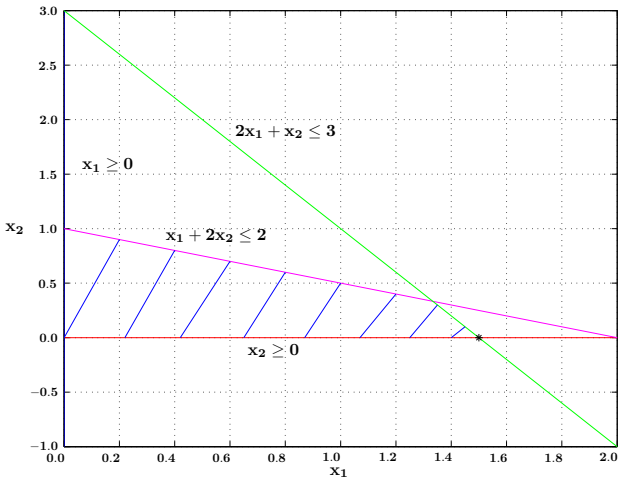


Figure C.2: *A graphical representation of the constraints of the primal problem, with solution* ($x_1 = \frac{3}{2}, x_2 = 0$)

of the primal problem.

The solution of the primal basic problem is feasible and the basic dual problem is infeasible. This means that for the second step a pivot is performed on the primal problem.

$$B_2 = \{w_1, x_1\} \qquad i \;=\; 1,\; 2,$$
$$\mathcal{N}_2 = \{w_2, x_2\} \qquad j \;=\; 1,\; 2,$$

$$\{j \in \mathcal{N} : c_j > 0\} \qquad \Rightarrow \{x_2\} \qquad k \;=\; 2,$$

$$\min_{i \in B : a_{ik} > 0} = \{\frac{b_1}{a_{12}} \;=\; \frac{\frac{1}{2}}{\frac{3}{2}}\;,\; \frac{b_2}{a_{22}} \;=\; \frac{\frac{3}{2}}{\frac{1}{2}}\} \;=\; \min\{\frac{1}{3}\;,\; 3\} \qquad \Rightarrow i = 1, \tag{C.27}$$

The leaving and entering variables of the primal problem are x_2 and w_1. For the dual problem z_2 and y_1 are the entering and leaving variables.

Rewriting w_1 results in:

$$x_2 = \frac{1}{3} - \frac{1}{3}w_2 - \frac{2}{3}w_1, \tag{C.28}$$

The new primal and dual dictionary are:

(P)
$$\zeta = \frac{4}{3} - \frac{1}{3}w_2 + \frac{1}{3}w_1, \tag{C.29}$$
$$x_2 = \frac{1}{3} + \frac{1}{3}w_2 - \frac{2}{3}w_1,$$
$$x_1 = \frac{4}{3} - \frac{1}{3}w_2 + \frac{1}{3}w_1,$$

(D)
$$-\xi = -\frac{4}{3} - \frac{1}{3}z_2 - \frac{4}{3}z_1, \tag{C.30}$$
$$y_2 = \frac{1}{3} - \frac{1}{3}z_2 + \frac{1}{3}z_1,$$
$$y_1 = \frac{1}{3} + \frac{2}{3}z_2 - \frac{1}{3}z_1.$$

Step 3:

The primal basic solution is: $x_1 = \frac{4}{3},\ x_2 = \frac{1}{3},\ w_1 = 0,\ w_2 = 0 \rightarrow \quad \zeta = \frac{4}{3}$
The dual basic solution is: $\quad y_1 = \frac{1}{3},\ y_2 = \frac{1}{3},\ z_1 = 0,\ z_2 = 0 \rightarrow -\xi = -\frac{4}{3}$

In Figure C.3 the optimal solution of the primal problem is shown by $*$, in combination with the constraints of the primal problem.

$$B_3 = \{x_2, x_1\} \qquad i \ = \ 1,\ 2,$$
$$\mathcal{N}_3 = \{w_2, w_1\} \qquad j \ = \ 1,\ 2,$$

$$\{j \in \mathcal{N} : c_j > 0\} \qquad \Rightarrow \{\}. \tag{C.31}$$

The set of (C.31) is empty. This means that the solution is optimal. In the optimal solution the primal and dual dictionary are both feasible.
The optimal solution to this LP problem is:

$$x_1 = \frac{4}{3},\ x_2 = \frac{1}{3}$$

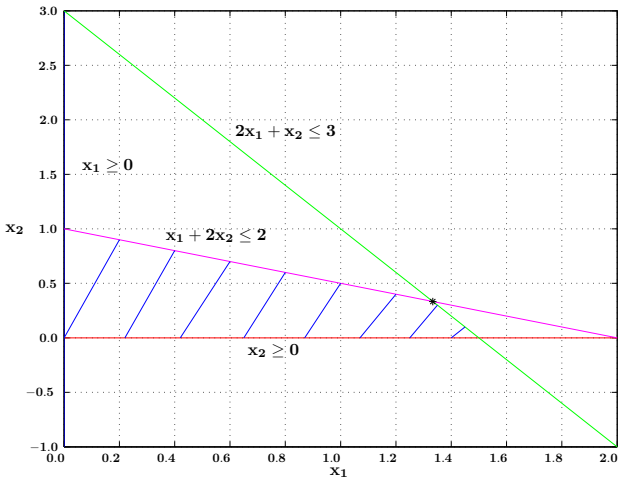


Figure C.3: *A graphical representation of the constraints of the primal problem, with solution* $(x_1 = \frac{4}{3}, x_2 = \frac{1}{3})$

C.4 Discussion

This appendix showed some of the basic theories of the Linear Programming theory and some examples were presented to illustrated the use of LP theory.

Appendix D

Programs

As mentioned this project uses a hierarchical approach to control the manufacturing system. In the simulation the manufacturing system is modeled by using χ-language, a language which was specially designed for modelling, simulation and control of concurrent manufacturing systems. This is the heart of the simulation. The controller actions needed in the hierarchical approach are calculated by using an optimization solver in MATLAB. The communication between χ and MATLAB is done by using pytmath [Ste99, Hof03] interface in Python language. In this appendix the programs is explained in combination with the codes used.

D.1 MATLAB

In the hierarchical approach used in this project the control actions are calculated by the SCLP algorithm described in Chapter 4. This algorithm is translated to a solver and was supplied to use by Weiss. The solver is made for MATLAB as pcode files, which means that the solver should be seen as a black box, where the translation of the MPC problem is the input for the solver and it calculates the optimal solution.

This section explains how the pcode files should be used. An explanation of the way the input for the solver has to be generated and a file that processes the data to compare the calculated targets with the actual output of the discreet event model.

SCLP algorithm

The SCLP solver contains a set of pcode files that perform the steps of the SCLP algorithm. The main file is an m-file that combines all these files into one solver. In order that the solver can operate properly the subdirectories with the pcode need to be added to the path definition of MATLAB. To do so the following script is created.

```
% Creating path definitions in a Linux environment
%
%
cd /mnt/hit/s444792/Afstuderen/Programs/Reprog-SCLP/Reprog-MATLAB-Pcode
pa = pwd;                       % finding the path
% Setting the directories and subdirectories into the path
addpath(pa)
addpath([pa '/DisplayPlot'])
addpath([pa '/MainRoutines'])
addpath([pa '/Scripts'])
addpath([pa '/Subroutines'])
```

The script first sets the directory of the SCLP solver and then adds the separated directories to the path of MATLAB.

When the directories are added to the path, the input parameters of the SCLP algorithm are needed. These parameters are the Matrix G, H, F and the vectors $a, b, c, d, \gamma, \alpha, T$. To formulate the input matrix and vectors the following function is created. This function needs the arrival rate of the system (ar), the number of buffer (K), the number of machines (I), a vector with the process times (te) and the routing of the machines ($route$).

```
% Creating input of the simulation model

function [G,H,F,a,b,c,d,gamma,T,K,B,y_buf,pp,pu,tt]=inputSCLP(ar,K,I,te,route)   %

% K is number of buffers
% I is number of machines
% p are the cost of having product in the buffer (p < 0, because the
% problem is a Max and the buffer level should be as low as possible).
% r are the cost of producing
% Ym is the buffer capacity
% ta is the speed with which the product should leave the line.
% b is the machine capacity

p = -(10:10:10*K);
r = -0.2.*ones(1,K);
T = inf;

b = ones(I,1);

B = -eye(K)+diag(ones(1,K-1),-1);

G = [-B ;B];
% Cost function
gamma =  zeros(1,K);
c =  (r+p*B);
d = [];

% Buffer constraint
F = [];
l = [1; zeros((K-1),1)];
a = ar.*[l; -l];

% Machine constraint
H = zeros(I,K);
for i=1:size(te,1)
    H(route(i,:),i)= te(i,:);
end

% Inisiliting the datalogging
y_buf.lev = [];
y_buf.lt = [];
y_buf.mul = [];
y_buf.mut = [];
```

```
y_buf.levSCLP = [];
y_buf.ltSCLP = [];
y_buf.uSCLP =  [];
y_buf.utSCLP =  [];
pp = zeros(K+1,1);
pu = zeros(I+1,1);
tt = zeros(K+1,1);
```

The vector α is not defined in this function because this vector changes each run. This means that this vector is determined before each run of the SCLP solver. Appendix A shows that the vector α is determined by using the maxim buffer level and the buffer level at the moment of the calculation. When the vector is determined the SCLP solver can calculate the optimal solution.

The SCLP solver is called by using the functions SCLP and the parameters mentioned above. It also needs the parameters $messages, graph, taxis, xaxis$ and $qaxis$. The messages level is needed to determine what type output is needed. There are five type of message levels. The first message level is 1, in this case the solver only shows the end results (the objective, the error, the number of intervals, the number of pivot, the number of flops and the cpu usage). When the messages level is set to 2, the solver also shows the start setting of the mean problem and the subproblems. In case of Massages level 3 the solver shows all the iterations. The solution to each iteration is shown in nine columns. The meaning of each is described as follows:

1. The first column is the step count,
2. The second column is the depth of a problem (0 main problem, 1 subproblem),
3. The third column is the iteration within a subproblem,
4. The fourth column shows the dimensions of the problem (K, J, I, L),
5. The fifth column are the number of intervals, dictionaries, bases,
6. The sixth column is the validity region of that interval,
7. The seventh column shows the collision case,
8. The eighth column represents the indices of the interval preceding and succeeding the collision and finally
9. The ninth column shows the collision variable with $+$ for the x variables and $-$ for the q variables.

The parameter graph is used to define the graphical solution of the solver, with graph level 1 no graphical results are shown and level 2 shows the results of the main problem and a graph with the result of the subproblem. The last three parameters are needed to decrease the length of the axis to improve the graphical results.

```
function [Output,t,u] = weissSCLP(G,H,F,a,b,c,d,gamma,Ym,y,T)

x0 = y(1:(size(y,1)-1));
alpha = [Ym ; Ym - x0];
Output = [];
messages = 1;
graphs = 1;

[t,x,q,u,p,Obj,Err,NN,stepcount,flopss,ecpu] = ...
        SCLP(G,H,F,a,b,c,d,alpha,gamma,T,messages,graphs,Inf,Inf,Inf);%

% transforming the matrix into a vector. Needed for the communication
% with pytmat.
for i=1:size(G,2)
    Output = [Output u(i,:)];
end

t=t';
```

After the solver has found a set of optimal solutions and the durations the solution are optimal. These optimal solutions are the target throughputs ($u[lot/hour]$). These targets are sent to the discrete event model, to simulate what would happen in a manufacturing line when these targets are the input for a manufacturing line.

To compare the result of the discrete event model with calculated the targets, targets need to be translated into buffer levels or the buffer levels of the discrete event model into throughputs. In this project the calculated targets are translated into buffer levels, by using the duration a solution is optimal and the dynamics of the system. With the notion that not the entire range of each optimal solution is taken into account. Only the duration that the discrete event model operates between two calculations is taken into account.

```
y_buf.lev = [y_buf.lev; zeros(max(sl),size(sl,1))];
y_buf.lt = [y_buf.lt; zeros(max(sl),size(sl,1))];
y_buf.mul = [y_buf.mul; zeros(max(su),size(su,1))];
y_buf.mut = [y_buf.mut; zeros(max(su),size(su,1))];

% Transforming the collum with all the buffer level and de collum
% with all time registration into two matrix.
for i=1:size(sl,1)
    if i == 1
        bg = 0;
    else
        bg = bg + sl(i-1,:);
    end
    lev = zeros(sl(i,:),1) ;
    lt = zeros(sl(i,:),1);
    for j=1:sl(i,:)
        k = bg + j;
        lev(j,1) = mlev(k,:);
        lt(j,1) = lmt(k,:);
    end
    y_buf.lev(((pp(i,:)+1):(pp(i,:)+ size(lev,1))),i)= lev;
    y_buf.lt(((pp(i,:)+1):(pp(i,:)+ size(lt,1))),i)= lt;
    pp(i,:) = pp(i,:) + sl(i,:);
end

% Transforming the collum with all the utilisation level and de collum
% with all time registration into two matrix.
for m=1:size(su,1)
```

```
    if m == 1
        bu = 0;
    else
        bu = bu + su(m-1,:);
    end
    mul = zeros(su(m,:),1) ;
    mut = zeros(su(m,:),1);
    for n=1:su(m,:)
        h = bu + n;
        mul(n,1) = mull(h,:);
        mut(n,1) = mutl(h,:);
    end
    y_buf.mul(((pu(m,:)+1):(pu(m,:)+ size(mut,1))),m)= mul;
    y_buf.mut(((pu(m,:)+1):(pu(m,:)+ size(mut,1))),m)= mut;
    pu(m,:) = pu(m,:) + su(m,:);
end

if size(y_buf.ltSCLP,1) ==  0
   ltend = zeros(1,size(y,1));
else
   ltend = y_buf.ltSCLP(end,:);
end

ten = lmt(end,:);
tt = lmt(end,:)-ltend(1,1);
BB = [B ; zeros(1,(size(B,2)-1)) 1];
aa = [a((1:2),:); 0];
tm = [];
uu = [];
fi = zeros(1,size(y,1));

% translating the calculated throughput's into buffer levels.
for w=1:size(y,1)
    tme = [];
    for v= 1:size(t,2)
        if tt >= t(1,v) & (v <= size(u,2))
            tme(:,v) = (ltend(1,w) + t(1,v));
        elseif (tt < t(1,v)) &  (fi(:,w) == 0)
                tme(:,v) = ten;
                fi(:,w) = 1;
        end
    end
    if tt > t(1,end)
        tme = [tme ten];
    end
    tm = [tm ; tme];
end

for z= 1:size(t,2)
        if tt >= t(1,z) & (z <= size(u,2))
            uu(:,z) = u(:,z);
        elseif (tt >= t) & (z > size(u,2))
            uu(:,z) = u(:,end);
        elseif (tt < t(1,z)) &  (fi(:,w) == 0)
                uu(:,z) = u(:,(n-1))
                fi(:,w) = 1;
        end
end

y_buf.uSCLP = [y_buf.uSCLP; uu(1:K,:)'];
y_buf.utSCLP = [y_buf.utSCLP; tm(1,:)'];

Y = y;
for n = 1:size(y,1)
    for l= 1:size(uu,2)
        o = l + 1;
```

```
        if o <= size(tm,2)
            xx= (BB(n,:)*uu(1:K,l).*(tm(n,o)-tm(n,o-1)));
            ra= (aa(n,:).*(tm(n,o)-tm(n,o-1)));
            Y(n,o)= round((Y(n,l)+ xx + ra)*100)/100;
        end
    end
end

y_buf.levSCLP = [y_buf.levSCLP; Y'];
y_buf.ltSCLP = [y_buf.ltSCLP; tm'];
```

With the following script the results are plotted.

```
% plot function
close all
clear all
clc
for m = 1:12
    type = m;
    if m == 1
        load sim12
        nam =['sim12'];
    elseif m == 2
        load sim12zvdp
        nam =['sim12zvdp'];
    elseif m == 3
        load sim12zv
        nam =['sim12zv'];
    elseif m == 4
        load sim2150
        nam =['sim2150'];
    elseif m == 5
        load sim2150zvdp
        nam =['sim2150zvdp'];
    elseif m == 6
        load sim2150zv
        nam =['sim2150zv'];
    elseif m == 7
        load sim478
        nam =['sim478'];
    elseif m == 8
        load sim478zv
        nam =['sim478zv'];
    elseif m == 9
        load sim478zvdp
        nam =['sim478zvdp'];
    elseif m == 10
        load sim35
        nam =['sim35'];
    elseif m == 11
        load sim35zv
        nam =['sim35zv'];
    elseif m == 12
        load sim35zvdp
        nam =['sim35zvdp'];
    end
    figure
    for i=1:size(y,1)
        subplot(size(y,1),1,i)
        plev = plot(y_buf.lt(1:pp(i,:),i),y_buf.lev(1:pp(i,:),i),'linewidth',1.5);
        set(plev,'Color',[ 0.753 0.753 0.753 ]);
        hold on
        plot(y_buf.ltSCLP(:,i),y_buf.levSCLP(:,i),'k','linewidth',1.5);
        grid
        ylabel('y [lots]')
```

```
        title(['Buffer ' num2str(i)])

        if i == size(y,1)
            act= y_buf.ltSCLP.*ar;
            plot(y_buf.ltSCLP,act,'k:','linewidth',1.5)
            title(['exit process'])
            legend('DEM','MPC/SCLP controller','expected buffer level',2)
            xlabel('time [hours]')
        else
            legend('DEM','MPC/SCLP controller',0)
            if type < 10
                axis([0 25 0 6])
            else
                axis([0 ceil(max(y_buf.ltSCLP(:,1))) 0 6])
            end
        end
    end
    saveas(gcf, name, 'fig')
    print('-depsc2', name)
    figure
    name= [name 'er'];
    for j=1:I
        if j == 1
            pmut = plot(y_buf.mut(1:pu(j,:),j),y_buf.mul(1:pu(j,:),j),'linewidth',1.5);
            set(pmut,'Color',[ 0.753 0.753 0.753 ]);
            hold on
        else
            plot(y_buf.mut(1:pu(j,:),j),y_buf.mul(1:pu(j,:),j),'k-.','linewidth',1.5)
        end
        if type < 10
            axis([0 25 -2 2])
        end
    end
    grid
    ylabel('error')
    xlabel('time [hours]')
    legend('Buffer 1','Buffer 2',0)

    saveas(gcf, name, 'fig')
    print('-depsc2', name)
end
```

D.2 Python

With the optimal solution it, is not important to get this solution into the discrete event simulation. This is done by using the pymath interface in python. For an explanation of the use of pymath see [Ste99, Hof03].

The interface works by creating function that can be called upon in χ. In this project there are five different functions. The two basic functions, opening MATLAB and closing it. The other three functions are the systemInitialization, this function creates the input parameters for the SCLP solver (see InputSCLP). The function WeissSCLP calculates the targets and the function datalogging is used to log all the calculated targets and the buffer levels of the discrete event system.

```
// com_SCLP.ext
//
language "python"
file "ComSCLP"

ext systemInitialization(depr: real,K,I: nat, Ym,root: nat*, te: real*) -> bool
ext weissSCLP(y: real*) -> (real*)^2
```

```
ext datalogging(let,lt,mull,mutl: real*, sl,su: nat*) -> bool
ext openMatlab(na: string) -> bool
ext closeMatlab(confirm: bool,na: string) -> bool
```

```
# Com_SCLP.py
# Version 1
# Com_SCLP is a function that communicates with chi and matlab
# using the pymat interface.
#

from Numeric import *
import pymat

# With this function matlab 6.0 is opened.
# And Setting the path to Matlab path, which enables the use of SCLP pcode Weiss created.
def openMatlab(name):
    global H
    H = pymat.open("matlab60 -nosplash")
    pymat.put(H,"name",name)
    pymat.eval(H,"diary(name)")
    pymat.eval(H,"cd /mnt/hit/s444792/Afstuderen/Programs/Event")
    pymat.eval(H,"Path_creationL")
    return 1

# This function initialize the parameter of the optimization problem
def systemInitialization(ar,K,I,Ym,route,te):
    pymat.put(H,"ar",[ar])
    pymat.put(H,"K",[K])
    pymat.put(H,"I",[I])
    pymat.put(H,"Ym",Ym)
    pymat.put(H,"route",route)
    pymat.put(H,"te",te)
    pymat.eval(H,"[G,H,F,a,b,c,d,gamma,T,K,B,y_buf,pp,pu,tt]=inputSCLP(ar,K,I,te,route);")
    return 1

# Solving the optimization problem.
def weissSCLP(y):
    pymat.put(H,"y",y)
    pymat.eval(H,"[Output,t,u] = weissSCLP(G,H,F,a,b,c,d,gamma,Ym,y,T);")
    Out = pymat.get(H,"Output")
    ti = pymat.get(H,"t")
    t = list(ti)
    O = list(Out)
    ou = tuple([O,t])
    return ou
def datalogging(mlev,lmt,mull,mutl,sl,su):
    pymat.put(H,"mlev", mlev)
    pymat.put(H,"lmt",lmt)
    pymat.put(H,"sl",sl)
    pymat.put(H,"mull", mull)
    pymat.put(H,"mutl",mutl)
    pymat.put(H,"su",su)
    pymat.eval(H,"Data_registration")
    return 1

# With this function matlab 6.0 is closed.
def closeMatlab(confirm,name):
    pymat.put(H,"name",name)
    if confirm==1:
        confirmed=raw_input ("Hit enter to close Matlab and all figures.")
    pymat.eval(H,"save(name)")
    pymat.close(H)
    return 1
```

D.3 χ model discription

Model description GBMBME line

In this section the model description of GBMBME line is given and the codes used to describe this model. The model is built in such a way that it should be possible to create a model for a larger line, by just changing the declaration and the cluster.

Declaration

The declaration is the first part of a χ model. In this part the standard and random libraries are imported and the file needed to communicate (the pymath file) is also imported.
After these imports, the constants and the type definition are declared. These constants and type definition are used to clarify the model. The constants declared in this model are $tm, nb, n, m, xe, Maxbuf, Y, ihs, route$.

- The constant tm is the unit index of time to create one time unit for the entire hierarchical approach,
- The constant $tmin$ is the minimum process time
- nb, n are constants related to the number of buffer and the number of machines,
- m is the number of buffer plus the exit process,
- xe is the last machine of the line,
- $Maxbuf, Y$ are the maximum buffer size and the initial buffer level,
- ihs is the lot history,
- $route$ is the routing of a lot in the line.

There is one type declaration, this is the representation of a lot in a line. The type contains the lot identification, the type of product, the machine step, the lot history and the start time of a lot.

Functions

The next part of the model are the definition of functions. In this model the following functions are used: bufsize, transformation, sequence, seq, updHis, upd, start, collect, conversionCD and type_event. The function bufsize is used to determine the buffer level and warn when it has reached its maximum level. Transformation is a function that translates the vector receive by the controller into targets for each machine. Sequence, seq, seqw are functions to create a tuple of booling values. The functions updHis, upd are used to update the lot info. The start function creates the initial buffer list, these lists are the lots that are in the system at the beginning of a simulation. The next function is the function collect, which is

used to collect the buffer level and its time instant. The function conversionCD is a very important function, this function converts the optimal throughput into machine targets. The final function, type_event, is created to set the right type of handles for each type of event that needs to be simulated.

Initialization

The Initialization process opens the MATLAB program and initiates the SCLP problem. It sends the initial buffer lists to the buffers and the info about the type of event and number of simulation are send to the machines and the control dispatcher (DCP).

DCP

DCP is the control dispatcher, it regulates and initiates the control optimization and it sends the targets to the machines. Before every new optimization it also collects the buffer levels and their time instances and these are processed by the function datalogging. When requesting the buffer levels, the DCP also request the fraction of time a lot has been processed. This time is added to the buffer after the machine and the remaining part of the lot is added to the buffer in front of the machine.

G

Process G is the generator of the system it send lots into the line with the arrival rate of the line. This process in only stopped at the moment a new target is calculated and when the first buffer has reached its maximum size.

B

The process B is the buffer. The buffer can receive lot until it has reached its maximum size, and sends a lot when the machine can receive a lot. At the end of one optimization run the DCP process asks the buffer the current buffer level and the list with the changes of the buffer levels and the time instant these change happened.

M

M stands for the machine. The machine receives its targets from DCP, the machine can always receive new targets. The targets are converted at the beginning of the period that these targets are optimal. When the target is larger then zero, and the machine is not processing a lot and the machine in not down. It can receive a lot and start to process it. The variable process time is determined by taking standard process time and adding a sample of normal distribution, with mean zero and a variance of 1.0. When using a fixed process time this distribution is left out.
The breakdown of a machine happens on a average of once a week. The moment depends on the distribution, it is a negative exponential distribution with a mean of 2 shift of 12 hours.

In case the DCP requests the buffer levels for new optimization and the machine is process a lot the machine determines the fraction of time the lot has been processed and sends this fraction to the DCP.

E

The process E is the exit process of a lot. When a lot leaves the line it is logged by the exit process and the process E communicates the changes of the exit process to the DCP when a new optimization run is started.

exit

The exit process is created to shut down MATLAB.

cluster

The cluster connects all the channels between the processes as described in Figure D.1

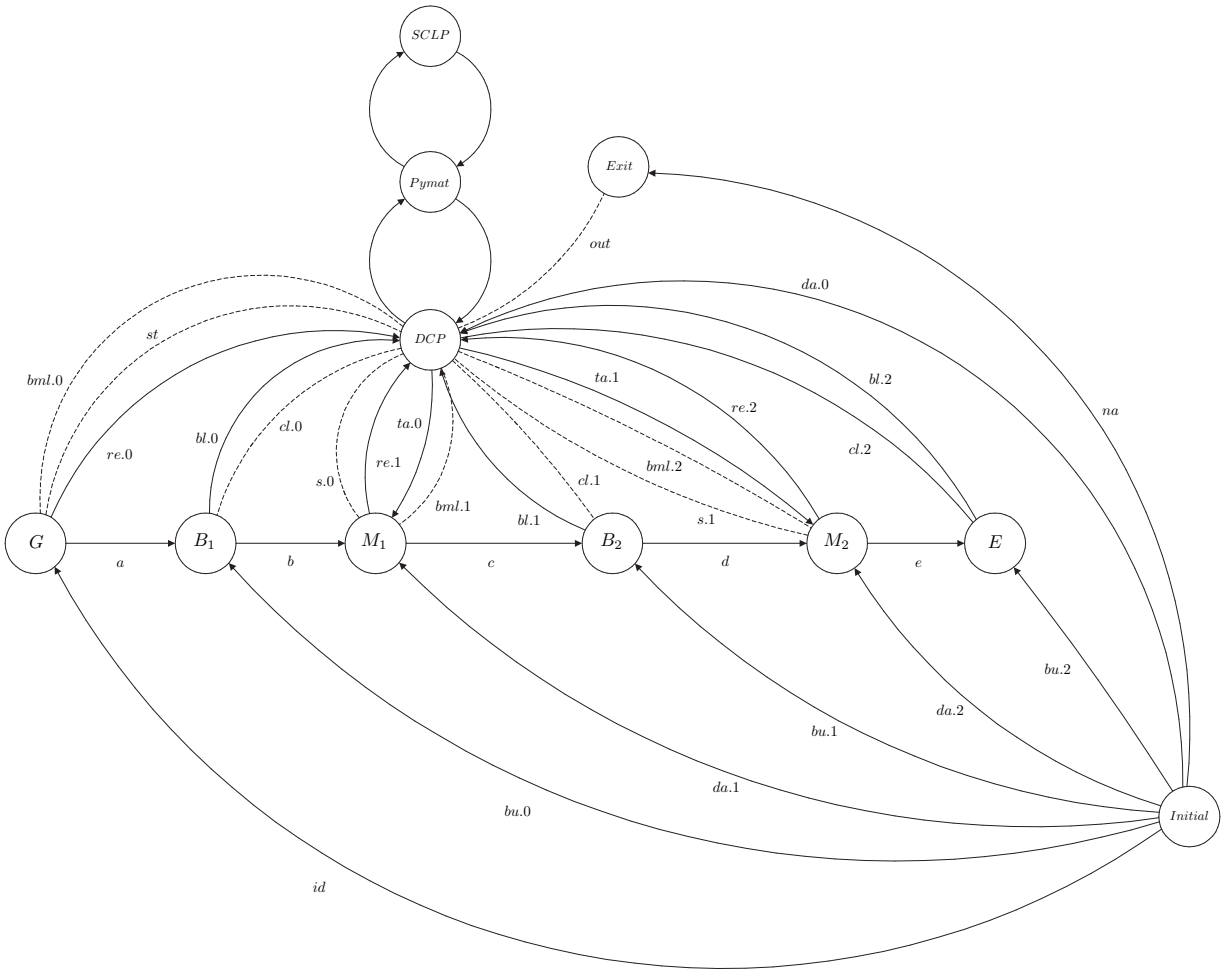


Figure D.1: χ *model structure.*

D.4 χ-code

```
from std import *
from random import *
from ComSCLP import *

// Type of event possibilities
// -1- Every 12 hours
// -2- After every finished product
// -3- After a breakdown
// -4- After the product is finished on the bottleneck station

const  tm:     real   = 60.0              // Unit index of the time (60 min)
     , tmin:   real   = 0.0               // The lowest stochactic process time
     , nb:     nat    = 2                 // Number of buffers
     , n:      nat    = 2                 // Number of machines
     , m:      nat    = 3                 // Number of buffers + exit process
     , w:      nat    = 3                 // Number of machines + Genarator
     , xe:     nat    = 1                 // Number of the machine in front of the exit process (n-1)
```

```
     , Maxbuf: nat^2  = <|10,10|>           // Maximum buffer size (B1,B2)
     , Y:      real*  = [2.0,3.0,0.0]       // Initial buffer level (B1,B2,E)
     , ihs:    nat^2  = <|1,1|>             // Lot History (M1,..,M_n)
     , route:  nat^2  = <|1,2|>             // which buffer service which machine (B1,..,B_nb)


type lot = id.nat#tp.nat#sp.nat#hs.nat^n#st.real
      // identification#type(0,1)#step(0...n)#history#starttime

// Bufsize gives a warning when the buffer exceed there maximum
func bufsize(sbf: nat,sp: nat) -> nat =
|[ [sbf < Maxbuf.sp
     -> skip
   |sbf >=  Maxbuf.sp
     -> !"Buffer = ", sp," is has reached the maximum buffer size.\n"
   ]; ret sbf
]|

// Transformation assigns the targets of each optimal solution, to a machines.
func transformation(SCLP: (real*)^2) -> (real*#real*)^n =
|[ s,i,j,k: nat, targ: (real*#real*)^n, Output: real*
 | s:=len(SCLP.1)-1; i:= 1; j:=0;k:=0; Output:= SCLP.0
 ;*[len(Output)> 0
     -> [i<= s
         -> targ.j.0:= targ.j.0 ++ [hd(Output)]
           ; Output:= tl(Output); i:= i + 1
        |i> s
          -> j:= j + 1; i:= 1
        ]
   |len(Output) = 0
     -> *[k<nb -> targ.k.1:= SCLP.1; k:= k + 1]; ret targ
   ]
]|

// sequence create a booling tuple size n
func sequence(send: bool^n) -> (bool^n)=
|[ k: nat
 | k:=0
 ;*[k<n -> send.k:=true; k:= k + 1]
 ; ret send
]|

// seq create a booling tuple size m
func seq(nu: bool^m) -> (bool^m)=
|[k: nat
 |k:= 0
 ;*[k<m -> nu.k:= true; k:= k + 1]
 ; ret nu
]|

// seq create a booling tuple size m
func seqw(nu: bool^w) -> (bool^w)=
|[k: nat
 |k:= 0
 ;*[k<w -> nu.k:= true; k:= k + 1]
 ; ret nu
]|

// updating history
func updHis(hs: nat^n, bn: nat) -> nat^n =
|[ i,j: nat
 | i:= 0
 ; *[i<bn
      -> j:= (route.i-1); hs.j:= hs.j - 1; i:= i + 1
    ]
 ; ret hs
```

```
]|

// Update changes the lot parameters.
func upd(x:lot, j: nat) -> lot = |[ x.sp:=x.sp + 1; x.hs.j:= x.hs.j - 1; ret x ]|

// Start function creates the lots that are in the system at the beginning of the simulation.
func start(y: real*, t: real) -> (lot*)^m#nat=
|[i: real^m, j,k,l: nat, buf: (lot*)^m, ih: nat^n
 |j:=0; k:=0; l:= 0;  ih:= ihs
 ;*[k<m -> i.k:= 0.0; buf.k:=[]; k:=k+1]
 ; *[l<m -> [ i.l<hd(y)
                -> buf.l:=buf.l ++ [<j,0,l,updHis(ih,l),t>]; j:=j+1
                  ; i.l:=i.l + 1.0
            | i.l>= hd(y)
                -> y:= tl(y); l:= l + 1
            ]
    ]
   ; ret <buf,j>
]|

// Function collect, gathers the buffer level en times. This is done to compare the
// error between the target and the actual production.
func collect(bu,t: real, lt, lev: real*) -> (real*#real*)=
|[ lev:= lev ++ [hr(lev)];lt:= lt ++ [round(t*1000)/1000]
 ; lev:= lev ++ [bu]; lt:= lt ++ [round(t*1000)/1000]
 ; ret <lev,lt>
]|

// The function conversionCD translates the calculated speeds
// into machine targets.
func conversionCD(targ,opt: (real*),t: real) -> (int#real*#real*) =
|[ tra: int, ta,op,t1,t2,tar: real, s: bool, tt: real*
 | s:= false
 ; *[len(targ) > 0 and not s
      -> [t <= hd(opt) or t = 0.0
            -> tar:= hd(targ); s:=true
         |t >  hd(opt) and t /= 0.0
            -> ta:= hd(targ); op:= hd(opt); targ:= tl(targ); opt:= tl(opt)
         ]
    |len(targ) = 0 and not s
      -> targ:= [ta]; opt:= [op];tar:= hd(targ); s:=true
    | s
      -> tt:= take(opt,2)
        ;[len(tt) > 1
           -> t1:= hd(tt); t2:=hr(tt)
         |len(tt) = 1
           -> t1:= hd(tt); t2:=hd(tt)+ 1
         ]
        ; [(t2-t1) > 0 -> tra:=  round(tar*(t2-t1))
          |(t2-t1) = 0 -> tra:=  round(tar*1)
          ]
        ; ret <tra,targ,opt>
    ]
]|

func type_event(event: nat,t,bt: real) -> (bool#bool#bool#real)=
|[ea,br,ex: bool, brt: real
 |ea:= false; ex:= false; br:= false; brt:= 0.0
 ;[event = 1                                          // Type of sample scheme.
    -> skip
  |event = 2
    -> ea:= true
  |event = 3
    -> br:= true; brt:= t + bt; !"brt= ",brt,nl()
  |event = 4
    -> ex:= true
```

```
  ]
 ; ret <ea,br,ex,brt>
]|


// Initialization the input of the system
proc Initialization(da: (!nat^2)^m, bu: (!lot*)^m, iden: !nat, na: !string
 , type_sym, nr_sym: nat, ar: real,tml: real*, name: string)=
|[open, iniM: bool, bb: bool^m, sim: (nat^2)
 , buf: (lot*)^m, id,g,i,k: nat, Ym,rout: nat*, tb: real*
 | g:=0; i:=0; k:=0; Ym:= []; tb:= [];rout:=[]
 ; na!name
 ; name:= name ++ ".txt"
 ; sim.0:= type_sym
 ; sim.1:= nr_sym
 ; open:= openMatlab(name)                              // Opening Matlab using Pymat interface.
 ; *[g < nb -> Ym:= Ym ++ [Maxbuf.g]
             ; tb:= tb ++ [hd(tml)/tm]; tml:= tl(tml)
             ; rout:= rout ++ [route.g]
             ;g:= g + 1
   ]
 ; iniM:= systemInitialization(ar,nb,n,Ym,rout,tb)    // Initialization of the SCLP problem.
 ; bb:= seq(bb)
 ; *[j:nat <-0..m: bb.j                                // Sending type of sampling scheme and number
     -> da.j!sim; bb.j:=false                          // of simulation to DCP and machines.
   ]
 ; <buf,id>:= start(Y,time)                            // Creating the initial buffer levels.
 ; iden!id
 ; *[l:nat <-0..m: not bb.l                           // Sending initial buffer level to the buffers.
     -> bu.l!buf.l; bb.l:=true
   ]
]|

//
proc DCP(bl: (?real#real*#real*)^m, da: (?nat^2), s: (?void)^n, p: (?real#real*#real*)^n
         , ta: (!(real*#real*))^n, cl,bml: (!void)^m, fi: (!void)^n, out,st: !void)=
|[q: nat,sl,su: nat*, sim: (nat^2), SCLP: (real*)^2
, targ: (real*#real*)^n, y,lmt,mlev,mutl,mull: real*, lt,lev: (real*)^m, mut,mul: (real*)^n, yj: real^m
, re: real^n, send,receiveM: (bool)^n, receive: (bool)^m, signal: (bool)^w, data: bool
| q:=0; lmt:= []; mlev:= []; y:=[]; sl:=[]; mutl:= []; mull:= []; su:=[]
; da?sim
; y:=Y
; *[q<sim.1
     -> send:= sequence(send)
       ; SCLP:= weissSCLP(y)                                  // Calculating new target with SCLP-algorithm.
       ; targ:= transformation(SCLP)
       ; *[i:nat <- 0..n: send.i; ta.i!targ.i        // Sending target to machines.
            -> send.i:=false
          ]
       ;  st!
       ;[ sim.0 = 1                                          // Event sampling.
          -> delta 720
        | sim.0 = 2 or sim.0 = 3
          -> [j:nat <- 0..n: true; s.j? -> !"s.",j,nl(); *[ii:nat <- 0..n: not send.ii; fi.ii! ->  send.ii:= true]]
        | sim.0 = 4
          -> [true; s.xe? -> skip]
        ]
       ; q:= q + 1
       ; y:= []
       ; receive:= seq(receive)
       ; *[j:nat <- 0..m: receive.j; cl.j!                  // Gathering buffer levels.
            -> bl.j?<yj.j,lev.j,lt.j>; receive.j:=false
          ]
       ; *[x:nat <- 0..m: not receive.x -> y:= y ++ [yj.x]
                                         ; lmt:= lmt ++ lt.x
                                         ; mlev:= mlev ++ lev.x
```

```
                                        ; sl:= sl ++ [len(lev.x)]
                                        ; receive.x:= true
          ]                                                  // Sorting the buffer levels.
       ; signal:= seqw(signal)
       ; *[v:nat <- 0..w: signal.v; bml.v! -> [v = 0
                                                  -> skip
                                              |v > 0
                                                  -> p.(v-1)?<re.(v-1),mut.(v-1),mul.(v-1)>
                                              ]
                                              ; signal.v:= false
          ]
       ; receiveM:= sequence(receiveM)
       ; *[l:nat <- 0..n: receiveM.l ->[ re.l = 0.0  -> skip
                                       | re.l > 0.0  -> yj.l:= yj.l+ (1-re.l)
                                                        ; yj.(l+1):= yj.(l+1) + re.l
                                       | re.l >= 1.0 -> yj.(l+1):= yj.(l+1) + 1.0
                                        // In this case the lot is waiting to be send to the buffer.
                                       ]
                                       ; !"mul= ", mul.l,nl()
                                       ; mutl:= mutl ++ mut.l
                                       ; mull:= mull ++ mul.l
                                       ; su:= su ++ [len(mul.l)]
                                       ; receiveM.l:= false
          ]
       ; data:= datalogging(mlev,lmt,mull,mutl,sl,su)                  // Registering the data.
       ; sl:= []; mlev:=[]; lmt:=[]; su:= []; mull:=[]; mutl:=[]
  ]
  ; out!
]|

//
proc G(st,bml: (?void),iden: (?nat), a: (!lot*),ar: real) =
|[ i: nat, t,rem: real, rq,tar: bool
 | iden?i; tar:= false; rq:= false; rem:= 0.0
 ; *[true; st?                                    // receiving start time.
       ->[rem >  0.0 -> tar:= true
         |rem <= 0.0 -> rq:= true
         ]
        ; t:= time + rem
        ;!"G receives target at t= ",time," min",nl()
   |rq; a![<i,0,0,ihs,time>]                       // Sending lot into the line.
       -> i:= i + 1
         ; rq:= false; tar:= true; t:= time + (tm/ar)
   |tar; delta (t-time)
       -> rq:= true; tar:= false
   |true; bml?
       -> rem:= (t-time); rq:= false; tar:=false
   ]
]|


proc B(a,bu: (?lot*), cl: (?void), bl: (!(real#real*#real*)), b: (!lot*), stat: nat) =
|[ bufs: nat, xs,buf: lot*, Bf: real, lt,lev: real*
 | bu?buf; bufs:= bufsize(len(buf),stat);lev:=[n2r(bufs)]; lt:=[time/tm]
 ; *[bufs < Maxbuf.stat; a?xs                          // Receiving lot.
       -> buf:= buf ++ xs; bufs:= bufsize(len(buf),stat)
        ; <lev,lt>:= collect(n2r(bufs),time/tm,lt,lev)
        ; !"B ",stat," receives lot ",hd(xs).id," at t= ",time," min\tlevel= ",bufs,nl()
   |true; cl?                                          // Receiving request for the buffer level.
      -> Bf:= n2r(bufs)
        ; <lev,lt>:= collect(n2r(bufs),time/tm,lt,lev); !"Bf= ",Bf,nl()
        ; bl!<Bf,lev,lt>; lev:= [hr(lev)]; lt:= [hr(lt)]
        ; !"DCP sends Request to receive buffer level\n"
   |bufs>0; b![hd(buf)]                                // sending lot to the machine.
      -> !"B ",stat," sends lot ",hd(buf).id,"\tlevel= ",bufs,nl()
       ; buf:= tl(buf); bufs:= bufsize(len(buf),stat)
```

```
        ; <lev,lt>:= collect(n2r(bufs),time/tm,lt,lev)
    ]
]|

//
proc M(ta: (?(real*#real*)), a: (?lot*), da: (?nat^2), bml,fi: ?void
       , b: (!lot*), p: (!real#real*#real*),s: (!void), stat: nat, tml: real* )=
|[ xs: lot*, ti: (real*#real*), wa,ea,ex,br,rq,st,fp,se: bool
 , wait,t,tle,re,di,brt,tem,v,te,op,opr,sam: real, opt,mut,mul: real*, l: nat
 , event: nat^2, tra,targ,i: int, bt: -> real, tel: -> real
 | wa:= false ; st:= true; rq:= false; fp:= false; se:= true
 ; t:=time; i:=+0; l:= 0; tra:=+0; mut:= [0.0];  mul:= [time/tm]
 ; *[l = stat
       -> te:= hd(tml); l:= n
    |l /= stat and l < n
       -> tml:= tl(tml); l:= l + 1
    ]
 ; bt:= negexp(720.0); da?event; tel:= normal(0.0,1.0)
 ; <ea,br,ex,brt>:=type_event(event.0,time,sample bt)
 ;*[true; ta?ti                                          // receiving target.
      -> t:= time; rq:= true; st:= true
       ; !"ti",stat,"= ",ti,nl()
       ; v:= time
       ; targ:= +0
       ; op:= hd(ti.1); opt:= tl(ti.1)
       ; <ea,br,ex,brt>:=type_event(event.0,time,sample bt)
       ; !"Machine ",stat," receives new target at t= ",time," min",nl()
  |not wa and tra > 0 and st; a?xs                      // receiving lot.and v > time
     -> wa:= true; sam:= sample tel
      ; *[sam >= tmin and se -> se:= false
         |sam <  tmin and se -> sam:= sample tel
         ]
      ; se:= true; tem:= te + sam
      ; wait:= time + tem; tle:= time
      ;!"Machine ",stat," receives lot ",hd(xs).id," at t= ",time
      ;!" min and the process time= ",tem,nl()
  |true; fi?
     -> fp:= false
  |true; bml?                                            // requesting wip level.
     -> [not wa -> tle:= time
        |    wa -> skip
        ]
      ; di:= (time-tle); re:= (di/te); p!<re,mut,mul>; rq:= false; mut:= [hr(mut)];  mul:=[hr(mul)]
      ; !"request wip level ",stat," machine is processing t= ",time," min",nl()
  |wa;   delta (wait - time)                          // processing lot.
     -> xs:=[upd(x,(route.stat-1))| x:lot <- xs]; b!xs
       ; wa:= false; i:= i + 1
       ; tra:= tra - 1
       ; !"\nMachine ",stat," finished its product \tt= ",(time-t)/tm," hours\nnumber lots made= ",i,nl()
       ; [ ea
            -> fp:= true; rq:= false     // Request to calculate new targets.
         | ex
            -> [hd(xs).hs.xe = 0 -> fp:= true; rq:= false
               |hd(xs).hs.xe > 0 -> skip
               ]
         | not ea  or not ex
            -> skip
         ]
  |fp; s!
     -> fp:= false
  |br; delta (brt - time)                             // Machine Breakdown.
     -> fp:= true; !"machine breakdown at t= ", time," min",nl()
       ; rq:= false; wa:= false; br:= false; st:= false
  |rq; delta (v - time)
     -> <mul,mut>:= collect(i2r(i-targ),time/tm,mut,mul)          // collecting the new error value
       ; !"io",stat,"= ",(i-targ)," targ= ",targ,nl()
```

```
        ; <tra,ti.0,ti.1>:= conversionCD(ti.0,ti.1,(time -t)/tm)
        ; targ:= tra
        ; [len(opt) >= 1 -> [ (hd(opt)- op) > 0 -> opr:= ((hd(opt)- op)*tm)
                           | (hd(opt)- op) = 0 -> opr:= tm
                           ]
                         ; v:= time + opr; op:= hd(opt); opt:= tl(opt)
          |len(opt) = 0  -> v:= time + tm
          ]
        ; i:= +0
        ; !"Machine ",stat," Calculated target at t= ",time," min \ttarget= ",tra,nl()
   ]
]|

// E is the exit process.
proc E(a,bu: (?lot*), cl: ?void, bl: !real#real*#real*) =
|[  i: real, xs: lot*, lev,lt: real*
 | bu?xs; i:= n2r(len(xs)); lev:= [i]; lt:= [time/tm]
 ; *[true; a?xs
      -> i:= i + 1.0; <lev,lt>:= collect(i,time/tm,lt,lev)
         ;!"\nexit at t= ",time," min\nLot info= ",hd(xs)," \nNumber lot produced= ",i ,"\n",nl()
    |true; cl?
      -> <lev,lt>:= collect(i,time/tm,lt,lev)
         ; bl!<i,lev,lt>            // The u is the work in progress of the Generator
                                    // and ut is it's time registration.
         ; lev:= [hr(lev)]; lt:= [hr(lt)]
    ]
]|

proc exit(out: ?void,na: ?string)=
|[close: bool, name: string
 |na? name; out?; close:=closeMatlab(false,name)
]|

//
clus S(type_sym, nr_sym: nat,ar: real, te: real*,name: string)=
|[ bl: (-real#real*#real*)^m
 , a,b,c,d,e: -lot*
 , bu: (-lot*)^m
 , re: (-(real#real*#real*))^n
 , ta: (-(real*#real*))^n
 , da: (-nat^2)^m
 , id: (-nat)
 , s,fi:  (-void)^n
 , cl,bml: (-void)^m
 , out,st: -void
 , na: -string
 | Initialization(da,bu,id,na,type_sym,nr_sym,ar,te,name)
|| DCP(bl,da.0,s,re,ta,cl,bml,fi,out,st)
|| G(st,bml.0,id,a,ar)
|| B(a,bu.0,cl.0,bl.0,b,0)
|| M(ta.0,b,da.1,bml.1,fi.0,c,re.0,s.0,0,te)
|| B(c,bu.1,cl.1,bl.1,d,1)
|| M(ta.1,d,da.2,bml.2,fi.1,e,re.1,s.1,1,te)
|| E(e,bu.2,cl.2,bl.2)
|| exit(out,na)
]|

xper(type_sym, nr_sym: nat,ar: real, te: real*,name: string)=
|[ S(type_sym, nr_sym, ar, te, name) ]|
```