Markov based modeling of manufacturing systems dynamics

R.T.N.Beijsens

SE 420424

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, April 2005

FINAL ASSIGNMENT

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mechanical Engineering
Systems Engineering Group

April 2004

Student	R.T.N. Beijsens
Supervisor	Prof.dr.ir. J.E. Rooda
Advisor	Dr.ir. A.A.J. Lefeber
Start	April 2004
Finish	February 2005
Title	Properties of DEM and PDE models for manufacturing systems

Subject

In order to design control laws (that are used in the field of mechanical and electrical systems) for manufacturing systems suitable models are needed. Discrete event models (DEMs) are well-accepted for representing a manufacturing system, but are computationally expensive. Furthermore, no suitable control theory for real-life discrete event models is available.
Therefore, continuous time approximation models have been developed based on conservation laws resulting in Partial Differential Equations (PDEs). In recent research several PDE models have been developed. Currently, validation studies have shown that PDE models poorly describe transient behavior of manufacturing systems.

Assignment

Determine properties of DEMs to enable improvement of the behavior of PDE models. Thus, features of DEMs for several cases of manufacturing lines (e.g. identical machines in line, reentrant line) have to be worked out. Before starting with these cases averaging methods of simulations have to be studied. Then, cases can be started and for instance scaling aspects of performance are researched when machines in the line are doubled.
With for example scaling aspects DEM properties can be determined and herewith PDE models can be validated. In validating PDE models new conditions have to be formulated to which PDEs have to apply. Finally, the results of this study has to be presented in a report.

Prof.dr.ir. J.E. Rooda

Dr.ir. A.A.J. Lefeber

Systems Engineering

Department of Mechanical Engineering

Preface

Help, I am running out of days in my student life. Mayday. I am running low on days:

> 'Time is the best teacher; unfortunately it kills all its students'.

To avoid an early learning death, I have stretched the five year curriculum of my study of Mechanical Engineering with a couple of years. Most important though, I have enjoyed my student days, which began in an impoverished attic of the Edelweisstraat. Fortunately, I have looked for other housing and, moreover, other interests besides studying. For example, I have joined the student futsal organization of E.S.Z.V.V. Totelos and became a member of their committee for one year. In this year, I have learned a lot about non-study related activities.

After this (sabbatical) year, I have focussed on my study in which I had to choose a graduate group. From some interesting graduate groups, the Systems Engineering group applied to me the most, since the improvement of advanced manufacturing systems seemed very rewarding. Thanks to χ.
Furthermore, with this graduate group, I was able to follow mathematical courses like Process Algebra and Stochastic Performance Modeling, which amused me. As part of the graduate program, I went abroad to follow my internship in England at Hagemeyer UK Ltd. Logically, I would like to thank professor Udding for providing me with this internship and supporting me during this three month period.
Hereafter, I was ready for my master's project. For this project, I would like to thank Erjen Lefeber for coaching me and his support. Besides, I want to thank professor Rooda for supervising this project.

Some other thanks go out to all fellow-students and PhD students from the SE-lab, such as S.E.R.R. Tosserams, E.P.T. Kock, Miel and Wouter and Bas for the mini-golf competition. Moreover, I would like to thank my parents, who believed in me and gave me the genetic (and financial) power to study. Last but not least, a special thank goes out to my girlfriend Eefje who supported me during a big part of my student days.

Over and out.

Rolf Beijsens

Summary

The subject of this thesis finds its origin in a control framework developed by the Systems Engineering group of the department of Mechanical Engineering of the Technische Universiteit Eindhoven. The aim of this framework is to control a discrete manufacturing system with a continuous controller. For the development of the controller, a continuous model of the considered manufacturing system has to be made. The continuous model can be validated with a discrete-event model (DEM). After a successful validation, the designed controller can be tested when connected to a DEM. Finally, the designed controller can be implemented in a real-life plant.

The first objective of this thesis is to find and define properties of manufacturing systems. These properties have to be be used to validate manufacturing system models such as the Ordinary Differential Equation (ODE) and Partial Differential Equation (PDE) models. Second, the search for properties needs to lead to the development of alternative models and improvement of current models that meet these properties.

The search for properties of manufacturing systems started with an analysis of a single server queue with exponential arrival and process rates and an infinite buffer, the $M/M/1$ queue. The focus in this search has been put on three properties that can be used to scale performance measures of the model in time.
The first suggested property multiplies the process rate with time and is called the process rate property. With help of Markov theory, it has been proven that this property is exact for the $M/M/1$ queue. The process rate property can be used for scaling performance measures like the flow time, throughput and wip-level, the number of lots in the system. With help of the process rate property, $M/M/1$ queues with different process rates show identical behavior when scaled using this property. Besides single server queues, multiple $M/M/1$ queues in series have been considered as well. A Markov proof of a scaling factor for $M/M/1$ queues in series has not been found. Therefore, a validation has been performed for queues in series, which showed that the process rate property holds also for queues in series.
For the other two suggested scaling properties, a proof has not been made, since these are approximations. Consequently, validations have been made with DEMs. For the second suggested property, the possibility of scaling time-dependent performance measures has been investigated for queues with different utilizations, the utilization property. Given that an exact approach with the time-dependent Markov solution has not

succeeded, the relaxation time has been used as an approximation to determine the utilization property. The effect of the relaxation time on the utilization property has been validated with a DEM. From the validation, it appeared that the approximation for the utilization property corresponds better for the asymptote to steady state than the initial transient of time-dependent performance measures.
Finally, the regarded scaling property has been the workstation property in which the number of identical workstations in series has been studied. This property should connect performance measures of queues with a different number of identical workstations. Since the determination of the workstation property has not been determined exact, an approximation has been used. In this approximation, the influence of a single workstation on the whole system has been defined with a time factor. This time factor approximation shows that a system with two identical workstations finds itself in the same situation $1\frac{1}{2}$ times later than a system with a similar single workstation.
In all, the effect of scaling properties on validation methods has not been considered in this research, but the focus has been put on the definition of three scaling properties. These three properties have been investigated. For the process rate property, a proof has been made. The other two properties, the utilization and the workstation property, have been approximated.

Markov theory has not been used for the proof of the process rate property only. In this proof, Markov theory has been used to obtain the time-dependent solution of the single server $M/M/1$ queue. Markov theory has been used for the development of transfer function model as well.
The transfer function model has been developed to increase the applicability of the time-dependent solution, since the derivation of this solution is very complex and has a limiting applicability. The transfer function model uses an intermediate solution of the time-dependent derivation to act as a transfer function. This intermediate solution has been adjusted with Taylor and Padé approximations to make it suitable for a transfer function. The result is an approximated transfer functions, which has a desired applicability increase with respect to the original time-dependent solution of the $M/M/1$ queue.
The applicability has been increased, because transfer functions for single server queues can be extended easily to transfer functions for queues in series. A disadvantage of the approximated transfer functions is a reduced accuracy for high utilizations. Besides the utilization accuracy loss, an accuracy also loss occurs when more servers in series are considered.
In short, transfer function models are not perfect, since accuracy loss arises from Padé and Taylor approximations. However, the developed transfer function is a new alternative which can be used to describe the behavior of $M/M/1$ queues. Furthermore, this transfer function can certainly be used for systems which do not have high utilizations.

Alongside the consideration of infinite queues, finite queues have been considered as well. These finite $M/M/1/N$ queues have been modeled with transfer functions derived again from Markov theory. The modeling of these transfer functions is a lot easier for finite than infinite queues. This modeling simplicity originates from the finite number of

Ordinary Differential Equations (ODEs), while infinite queues have an infinite number of ODEs. Next to the simplicity advantage, the finite queue transfer function is exact. However, the finite queueing transfer function also has some disadvantages, since the applicability has been reduced in comparison with the infinite queue. First, an extension from a single server queue to multiple queues in series is not straightforward anymore. Therefore, a new model has to be made for every number of workstations. Second, the applicability reduces, since a new model has to be made for every number of buffer places. These two disadvantages can be avoided using a mathematical model to generate the transfer function. The third disadvantage occurs, because the computation of the transfer requires an inverse of the transition state matrix. Inverting this matrix can take a long time as buffer places and the number of workstations increase.
In brief, the only remaining disadvantage is the applicability of finite queueing transfer functions for large systems. The useability for large systems has not been considered in this thesis. The advantages of the transfer function are the accuracy and the suitability to describe a finite queue with a continuous model.

Finally, the design and DEM implementation of a controller has been discussed in this thesis. The design of the controller is done with state feedback. With state feedback, the controller uses the continuous model to generate all system states from the output signal. With this state information, the controller provides the system with a new input signal. Now, the designed controller can be used in a DEM implementation. This implementation has been performed for two $M/M/1/2$ queues in series. In this implementation, the wip-level has been used as output signal and the arrival rate as input signal. The DEM implementation should show that the control signal depends on actual wip-levels compared to steady state wip-levels. So, when the actual wip-level is lower than its steady state value, the controller should increase the input rate. Otherwise, a higher wip-level than steady state should lead to a decrease of the input rate.
When an implementation of a single simulation run is considered, the controlled model meets its desired behavior. However, averages of multiple simulations reveal that the steady state wip-level of the controlled model exceeds the reference wip-level of the uncontrolled model.
Although, the reference steady state wip-level is exceeded, the controller can be used to absorb fluctuations in the process by varying the input signal.

Samenvatting

Het onderwerp van deze afstudeeropdracht ligt binnen het ontwikkelde regelconcept van de groep Systems Engineering van de faculteit Werktuigbouwkunde van de Technische Universiteit Eindhoven. Dit regelconcept richt zich op het regelen van fabricage systemen met een continue regelaar. Voor het ontwerpen van een dergelijke regelaar moet het fabricage systeem beschreven worden met een continu model. Hierna kan de ontworpen regelaar getest worden door een implementatie uit te voeren op een Discrete Event Model (DEM), waarna de ontworpen regelaar geïmplementeerd kan worden in een fabriek.

Dit onderzoek richt zich op het vinden en definiëren van eigenschappen van fabricagesystemen. Deze eigenschappen kunnen gebruikt worden om continue modellen zoals die van partiële differentiaal vergelijkingen te valideren. Bovendien wordt de zoektocht naar eigenschappen gebruikt om alternatieve modellen te ontwikkelen en deze te controleren op juistheid.

De start van de zoektocht naar eigenschappen is begonnen met een model van een enkele server wachtrij met exponentieel verdeelde aankomst en bewerkingssnelheden. Samen met een oneindige buffer resulteert het in de $M/M/1$ wachtrij. De nadruk bij het zoeken naar eigenschappen ligt op het schalen van parameters van een model in de tijd. In dit verslag worden enkele suggesties gedaan voor schalingseigenschappen.
Voor één van de voorgestelde eigenschappen is met behulp van de Markov theorie bewezen dat de eigenschap geldt. Deze schalingseigenschap vermenigvuldigt de bewerkingssnelheden met de tijd, resulterend in de bewerkingssnelheid eigenschap. Met de bewerkingssnelheid eigenschap kunnen prestatie indicatoren voor doorzet, doorlooptijd en wip-niveau geschaald worden. Het wip-niveau komt overeen met het aantal producten in het productieproces. Dit resulteert in onafhankelijkheid van de bewerkingssnelheid in een dimensieloos tijdsdomein voor de enkele server $M/M/1$ wachtrij. Bovendien worden meerdere $M/M/1$ wachtrijen in serie beschouwd. Hiervoor is geen Markov bewijs geleverd, maar er is een validatie uitgevoerd met een DEM. Hieruit blijkt dat voor wachtrijen in serie de bewerkingssnelheid eigenschap ook van kracht is wanneer er meerdere wachtrijen in serie staan.
Vervolgens zijn er ook voor andere voorgestelde schalingseigenschappen DEMs validaties gemaakt. Zo is de mogelijkheid onderzocht om tijdsafhankelijke prestatie-indicatoren van modellen met verschillende bezettingsgraden naar elkaar schalen. Aangezien een

exacte benadering met behulp van de tijdsafhankelijke Markov oplossing niet is geslaagd, is de relaxatietijd gebruikt als benadering om de utilisatie eigenschap te bepalen. Om het effect van de benadering te testen, is er een validatie verricht met een DEM. In deze validatie is het effect van de relaxatietijd als schalingsparameter bekeken. Hierin is de schaling voor de asymptoot beter dan voor het begin van het transiënte gedrag van de prestatie indicatoren.
Tot slot is het aantal identieke werkstations in serie bekeken ofwel de werkstations eigenschap. Deze eigenschap probeert een verband te leggen tussen de tijdsafhankelijke prestatie indicatoren van modellen, die een verschillend aantal werkstations hebben. Aangezien deze eigenschap niet exact bepaald is, wordt deze eigenschap benaderd. De gebruikte benadering geeft de invloed aan van het werkstation op tijdsafhankelijke prestatie indicatoren. Zo heeft een systeem met twee werkstations in serie anderhalf keer zoveel tijdsinvloed op een prestatie indicator in vergelijking met een systeem met een enkel werkstation.
Het effect voor het gebruik van deze schalingseigenschappen als validatie methode is niet beschouwd in dit onderzoek. Het is echter wel aan te raden om dit effect te testen, waarna deze wel of niet gebruikt kunnen worden voor validatie.

Het bewijs van de eerder genoemde bewerkingssnelheid eigenschappen is geleverd met behulp van Markov ketens en processen voor de enkele server $M/M/1$ wachtrij. Markov theorie kan gebruikt worden om de tijdsafhankelijke oplossing voor het gedrag van fabricagesystemen te verkrijgen. In deze analytische oplossing is tijd evenredig met de bewerkingssnelheid. De afleiding van de tijdsafhankelijke oplossing is erg complex en de tijdsafhankelijke oplossing heeft maar een beperkte toepasbaarheid.
Om dit op te lossen is er gezocht naar een meer algemeen model, waarin een tussenoplossing van de tijdsafhankelijke $M/M/1$ oplossing gebruikt wordt om als overdrachtsfunctie te fungeren. Om die tussenoplossing geschikt te maken, is er een Taylor en Padé benadering uitgevoerd met als gevolg de gewenste toegenomen toepasbaarheid in de vorm van een overdrachtsfunctie. Deze overdrachtsfuncties hebben het voordeel dat ze gemakkelijk kunnen worden omgezet om systemen met meer werkstations in serie te beschrijven door machtsverheffen. Hiermee wordt de toepasbaarheid vergroot. Uit een validatie met een DEM is gebleken dat het vergroten van de toepasbaarheid heeft geleid tot een lagere nauwkeurigheid van de overdrachtsfuncties voor hoge utilisaties. Daarnaast neemt de nauwkeurigheid van de benaderde overdrachtsfunctie af wanneer deze uitgerekt wordt voor het beschrijven van meerdere werkstations.
Kortom, het model van de overdrachtsfuncties is niet perfect door een lagere nauwkeurigheid, omdat er een Taylor en Padé benadering nodig waren. Het gevolg en het vermijden van deze lagere nauwkeurigheid is niet onderzocht. Toch is het beschrijven van het gedrag van $M/M/1$ wachtrijen met behulp van overdrachtsfuncties geslaagd. Deze benaderingsmethode kan zeker gebruikt worden voor systemen met minder hoge utilisaties.

Naast het beschouwen van oneindige wachtrijen, zijn er in dit onderzoek ook eindige wachtrij meegenomen, de $M/M/1/N$. Voor eindige wachtrijen is eveneens gebruik gemaakt van Markov modellen waarvan wederom een overdrachtsfunctie is afgeleid. Het

verkrijgen van een overdrachtsfunctie voor een eindige wachtrij is een stuk eenvoudiger dan in het geval van oneindige wachtrijen. De eenvoud ontstaat doordat een eindige wachtrij beschreven kan worden met een eindig aantal differentiaal vergelijkingen. Naast het voordeel van een eenvoudigere afleiding, is de overdrachtsfunctie van een eindige wachtrij exact. Hierdoor is er ook bij hoge utilisaties een goede nauwkeurigheid van het model. De overdrachtsfunctie van de eindige wachtrij heeft nadelen. Zo kan de overdrachtsfunctie niet meer gebruikt worden om een wachtrij te beschrijven met meerdere werkstations. Daarnaast moet er voor elke bufferinhoud een nieuw model gegenereerd worden. Dit probleem kan vermeden worden door een algoritme te ontwerpen dat de overdrachtsfunctie uitrekent. Het uitreken van de overdrachtsfunctie heeft echter wel een beperking, omdat hiervoor het inverteren van een toestandsmatrix vereist is. Het inverteren van deze matrix kan erg lang duren zeker als het aantal bufferplaatsen en werkstations groter wordt.
Het beschrijven van eindige wachtrijen met overdrachtsfuncties heeft als enige nadeel dat de inzetbaarheid voor het beschreven van grote systemen niet is getoond. Verder heeft de overdrachtsfunctie geen beperkingen en is uitermate geschikt om een continue model te maken van ieder systeem met eindige wachtrij.

Tot slot wordt er in dit onderzoek nog een stap gemaakt om een regelaar te implementeren op een DEM. Als regelwet wordt er een toestandsterugkoppeling gemaakt. De regelaar voedt het fabricagesysteem met een nieuwe invoersnelheid van producten, de input. De implementatie van de regelaar is uitgevoerd voor twee $M/M/1/2$ wachtrijen in serie. De implementatie is verricht met het wip-niveau als uitgang en regelsignaal. De regelaar zorgt ervoor dat een wip-niveau onder de steady state waarde wordt gecorrigeerd door een hogere invoersnelheid en met een wip-niveau boven de steady state waarde wordt de invoer gecorrigeerd met een lagere input snelheid.
Uit een validatie met een DEM is gebleken dat de regelaar goed functioneert voor een individuele simulatie. De input is hoger naarmate het wip-niveau lager wordt en vice versa. Er is echter wel een nadeel wanneer het gedrag van meerdere simulaties gemiddeld wordt. Na het middelen van meerdere simulaties wordt een andere steady state waarde dan de gewenste waarde verkregen.
Ondanks het steady state verschil kan de regelaar wel degelijk gebruikt worden om de schommelingen in de processen op te vangen door de invoersnelheid aan te sturen.

Contents

Abbreviations and Symbols

Abbreviations

CONWIP	Constanst Work-In-Process
DEM	Discrete Event Model
$GBME$	Generator Buffer Machine Exit
GB_NME	Generator N place Buffer Machine Exit
$GBME$	Generator Buffer Machine Buffer Machine Exit
GW_NE	Generator N place Workstation Exit
$GWWE$	Generator Workstation Workstation Exit
$M/M/1$	Queue with Memoryless arrivals and process rates
$M/M/1/N$	$M/M/1$ queue with N places
LQR	Linear Quadratic Regulator
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
WIP-level	Work-In-Process level

Symbols

A	State space matrix
B	State space matrix
C	State space matrix
$\mathfrak{C}$	Controllability matrix
$H(s)$	Transfer function
I_ν	Modified Bessel function of the first kind
i	Initial value of a system
K	Control gain
k	State number in a Markov chain
L	Observability gain
$\mathcal{L}$	Laplace operator
$P_k(t)$	Probability of being in state k at time t

$p_k(t)$	Probability of being in state k at time t
p_k	Steady state probability of state k
p_n	Padé coefficient n of numerator
$P_n(s)$	Numerator of Padé approximation of order n
$P_0^*(s)$	Laplace transform of $P_0(t)$
$P^*(z,s)$	Laplace transform and z-transform of $P_k(t)$
$\mathfrak{O}$	Observability matrix
Q	Rate matrix of an ODE relation
Q_m	Denominator of Padé approximation of order m
Q_{lqr}	Q matrix for the LQR method
q_n	Padé coefficient n of denominator
R_{lqr}	R element for the LQR method
s	Frequency domain variable
$\mathcal{T}_n$	Taylor series of order n
$U(s)$	Input relation of transfer function
u	(Feedback) input of transfer function
u_{ss}	(Feedback) steady state input of transfer function
w	WIP
$w(t)$	Time-dependent WIP
$\hat{w}$	Estimated WIP
$\overline{w}$	Steady state WIP
$\overline{w}(t)$	Average time-dependent WIP
X	Random variable
x	State vector of state space
x_{ss}	State vector of state space for steady state
$Y(s)$	Output relation of transfer function
y	Output of transfer function
t	Time
z	Discrete z-transforation parameter
$\hat{x}$	Estimate state vector of state space
Γ	Γ function
δ	Throughput
$\delta(t)$	Time-dependent throughput
$\overline{\delta}$	Steady state throughput
$\overline{\delta}(t)$	Average time-dependent throughput
ζ_i	Root i of the $P^*(z,s)$
λ	Arrival rate
μ	Process rate

ν	Non-integer variable of the modified Bessel equation
φ	Flow time
$\varphi(t)$	Time-dependent flow time
$\overline{\varphi}$	Steady state flow time
$\overline{\varphi}(t)$	Average time-dependent flow time
ρ	Utilization
τ	Time
τ_r	Relaxation time
τ_n	Workstation influence parameter

Chapter 1

Introduction

In manufacturing systems research, a lot of interesting fields come to mind, such as design, analysis, modeling, optimization and control. These topics are coherent and, consequently, most research contains several aspects of them. In this thesis, the research area can be characterized by model-based control and aspects involved are analysis, modeling and control.

1.1 Manufacturing systems

According to Hopp and Spearmann [Hop01], a manufacturing system is an objective-oriented network of processes through which entities flow. The objective of a manufacturing system can be improving throughput or flow time. In addition, it contains processes that are not only physical, but can include support of direct manufacturing (e.g., order entry, maintenance). Entities include the parts being manufactured and the information that is used to control the system. Moreover, the flow of entities describes how materials and information are processed. Finally, a manufacturing system is a network of interacting parts. Managing the network of interacting parts is as important as managing individual parts, if not so more.

In this thesis, manufacturing systems are also considered as a discrete event system. Here, the state of a system is instantaneously changed by an event at a certain point in time. Only arrival and departure times at processes are regarded as events, so the action of a process itself is not an event. An example of arriving and departing events at a process is shown in Figure 1.1. One can imagine that events in Figure 1.1 can correspond with arriving and departing products or lots at for example a workbench of an engineering factory. Furthermore, Figure 1.1 can be seen as a starting point for evaluating and analyzing performance of a process or manufacturing system. In manufacturing system analysis, commonly used parameters are flow time, throughput,

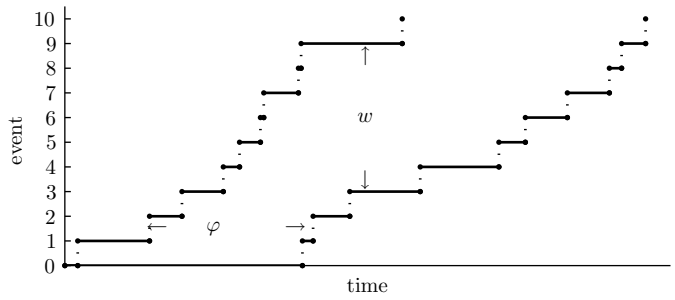


Figure 1.1: *Example of arrival and departure times at a process*

utilization and the Work-In-Process. In Rooda and Vervoort [Roo03], these are defined as:

- flow time (φ), the time it takes to travel through the system;
- throughput (δ), the number of lots per time unit that leave the system;
- utilization (ρ), the fraction of time lots possess a machine;
- Work-In-Process or wip-level (w), the number of lots in the manufacturing system.

From Figure 1.1, a couple of these indicators can be derived. The flow time is the horizontal difference and the wip-level is the vertical difference between arrivals and departures. The throughput can be computed from the departures. The utilization is not determined with graph data; it is predetermined by the quotient of arrival rate (λ) and processing rate (μ). In this study, λ and μ will be considered constant and therefore the utilization is also a constant. The expression of the utilization results in the following relation:

$$\rho = \frac{\lambda}{\mu} < 1. \tag{1.1}$$

Thus, in the models discussed later the utilization is a fixed input parameter, which in general has to be less than '1' one for infinite buffers. Otherwise, an instable system results. Accordingly, the performance measures concern flow time, throughput and wip-level with a settled utilization as a starting point. Another remarkable observation of Figure 1.1 can be the difference or variability between duration of events. Variability is very common in all kinds of processes. Due to variability in manufacturing systems, values of performance measures fluctuate, resulting in complexity. Therefore, models are required to imitate behavior of manufacturing systems.

1.2 Modeling and control of manufacturing systems

During the last century, a lot of activity has occurred in the manufacturing environment. Designing and improving manufacturing systems has become an important discipline in

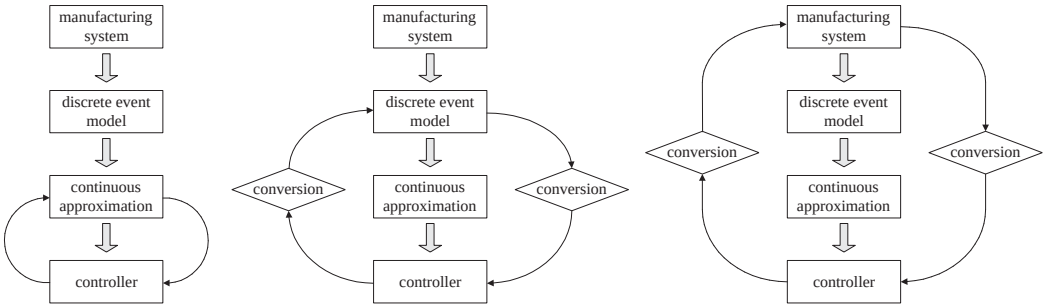
daily life. Together with variability, the evolution of manufacturing systems leads to a need for predicting behavior of the manufacturing systems. Consequently, models are developed to describe manufacturing behavior. In this thesis, three types of models can be distinguished:

- queueing models, based on queueing theory;
- discrete-event models, performed by computer simulations;
- continuous models, using e.g. differential equations.

The first one contains models that are derived from queueing theory, for example Markov theory, the Pollazcek and Khinchin formula [Pol30, Khi32] and Jackson networks of R.R.P. Jackson and J.R. Jackson [Jac54, Jac63]. The application of these models is for relatively simple systems. However, the development of manufacturing systems in the 20^{th} century asked for a better representation of system behavior. With the aid of computers, simulation models have been developed like the Discrete Event Model, abbreviated as DEM. The DEM is part of the second approach. Currently, DEMs are well-accepted for representing manufacturing systems. The last approach originates from traffic modeling ideas. Traffic modeling usually consists of continuous models that are based on for example flows using conservation laws. Traffic modeling results in ordinary or partial differential equations. Unfortunately traffic models cannot describe manufacturing systems as accurate as DEMs, but using ordinary or partial differential equations creates new possibilities, since these equations are continuous.

One of these new possibilities is developing controllers for manufacturing systems based on standard control theory instead of working with heuristics. CONstant Work-In-Process, CONWIP is one of the simplest heuristics. When a product leaves the manufacturing system, the CONWIP controller releases a new product into the system. The concept of using controllers from standard control theory forms one of the fundamentals for the developed research framework of which this study is part of. In Figure 1.2, the research framework is split in three parts. The first part contains modeling and design. Modeling concerns development of a DEM and a continuous approximation model for a manufacturing system. With a continuous model, a controller has to be designed using standard control theory. The second part consists of testing the controller on the DEM. Logically, the signals to and from the controller have to be transformed to continuous or discrete values. Consequently, two conversion steps are necessary for controlling the system and therefore have to be developed. Finally, in the third part the controller is implemented on the manufacturing system.

In order to be able to create a controller for a system, the models have to accurately describe time-dependent behavior. Especially the dynamics, or transient behavior, are of great importance for the controller. Transient behavior occurs in the start-up period of a system. Herein, the system starts empty and is fed with a certain rate. Therefore, the useability of queueing models is limited, since they describe mostly limiting behavior



(a) *modeling and design* (b) *conversion and test* (c) *implementation*

Figure 1.2: *Research framework*

and not dynamical behavior. DEMs are less suitable to design a controller, because current design techniques result in highly complex computation efforts for even the most simple practical situations. Consequently, a continuous model is desired to apply control theory.

1.3 Objective

In previous research on the framework of Figure 1.2 validation studies have been performed. Van den Berg and Platschorre [Ber04, Pla04] have validated continuous PDE models using the output of a DEM as a validation model. From the validation study, it appeared that in accurately describing transient behavior PDE models, 'leave much to be desired'. Therefore, improvement of continuous models is required within the framework of Figure 1.2. The objective of this research can be distinguished from the desire to improve continuous models and can be described as:

- determine properties of manufacturing systems;
- develop alternatives that satisfy these properties.

The first objective can be interpreted in the sense of the possibility of making general characteristics to which models have to fulfill. The second objective uses properties to develop new or adjust existing models. For example, Daganzo [Dag95] shows that with a simple observation an important modeling property can be found with respect to backward flow. In higher order PDE models, the flow of a manufacturing model can run backward. In manufacturing, products cannot run back, since backward flow corresponds with de-processing of products. Therefore, models in which backward flow might occur, can be rejected as suitable models for describing manufacturing systems.

1.4 Outline

The two part objective will be performed by the following approach. In Chapter 2, the basis of queueing theory is treated, namely stochastic processes. The next two sections refine these processes by considering Markov chains and the birth-death process. In the birth-death process, the transition is made to the first queueing model, the Markovian single server queue. The following two sections examine the limiting and time-dependent behavior for the single server queue. Furthermore, the first property is defined in Section 2.4.

Chapter 3 deals with the discrete event modeling. Herein, averaging of and performance measures of simulations are regarded in the first section. In the next section, the simulation itself is discussed. The last section has been used to investigate some properties of DEMs with respect to time scaling possibilities.

In the fourth chapter, a continuous model is presented in which the dynamics will be described by a transfer function. In the first section, an analytical approach has been made to approximate a transfer function of a queueing model.

Until Chapter 5, only infinite buffer queues have been discussed. From Chapter 5 on, the step towards finite buffers has been made to show the differences between finite and infinite buffers. Section 5.1 makes some changes in the birth-death process of Chapter 2 to introduce the finite buffer. Sections 5.2 and 5.3 consider and compare the Markov models with the DEM and its applicabilities for the single server queue as well as two queues in series. Moreover, the Markov models have been used to obtain a transfer function for some performance measures.

In Chapter 6, an earlier derived model has been used for designing a control law. The derived control law will be used to implement a controller on a DEM. Hereafter, results of the implementation will be discussed with the use of the control input and the model output.

Finally, Chapter 7 presents conclusions of this study. After the conclusions, some recommendations have been made for future research.

Chapter 2

Stochastic Processes

In the previous chapter, terms such as manufacturing systems have been defined. Furthermore, the chapter describes performance measures which can be used to evaluate manufacturing systems. Besides, the evaluation itself has been treated by introducing three different modeling types.
In this chapter, the first modeling type, queueing models, will be discussed. Evidently, these queueing models are based on queueing theory. The fundamentals of queueing theory are formed by stochastic processes. Within the area of stochastic processes, one can specify the class of Markov chains and Markov processes. In Section 2.1, Markov theory has been treated in order to define a first queueing model in Section 2.2 with the specification of birth-death processes. The last two sections consider the behavior of queueing models to achieve more insight in the manufacturing and to gain more information about properties of manufacturing systems.

2.1 Markov chains and Markov processes

In daily life, queues occur when customers cannot be served immediately due to limited capacities of service facilities. In manufacturing, queues happen as well. Now, customers represent products or lots, service facilities become machines and queues are formed in buffers. Both cases result in a queueing system, which can be seen as a stochastic process. Stochastic processes are processes of systems that evolve randomly, see e.g. Kleinrock [Kle75]. Such a process is a collection of states of random variables $\{X(t), t \in T\}$ where for each $t \in T$, $X(t)$ is a random variable.

Stochastic processes can be divided into several classes. Markov chains constitute an important class of discrete time stochastic processes. Moreover, using Markov chains forms the essential first step to compose a model for a manufacturing system. These Markov chains have a discrete state space containing a set of random variables $\{X_n\}$ which form a chain where the next state $\{X_{n+1}\}$ depends upon the current state $\{X_n\}$

only. The continuous time variant of a Markov chain is called a Markov process, where transitions between (discrete) states take place at a certain time. In this research, the continuous time variant is of most interest, since this variant is most suitable to describe a manufacturing system. In this description, the discrete state of the Markov chain can correspond with a product that stays in a machine (process) for a certain timespan before 'jumping' to the next process.

As mentioned above, the next state of a Markov chain depends on the current state only. Therefore, the specification of the current state contains the past history completely. This lack of history results in a strict constraint on the distribution function of a process. An exponential distribution satisfies the lack of history constraint, the so called 'memoryless property', and is specified below:

$$P(X > t + x | X > t) = P(X > x) = \mathrm{e}^{-\mu t}. \tag{2.1}$$

The memoryless property is required for all Markov chains and restricts the generality of processes to concern. In terms of states, it is called the 'Markov property' and can be expressed as,

$$P\Big(X(t_{n+1}) = x_{n+1} | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \ldots, X(t_1) = x_1\Big) = P\Big(X(t_{n+1}) = x_{n+1} | X(t_n) = x_n\Big). \tag{2.2}$$

Unfortunately, the presence of exponential arrival and process rates is not the most realistic distribution in manufacturing systems, especially for process rates. However, other distributions can be approximated with a sum of a finite sequence of exponential distributions. Therefore, the consideration of Markov processes is used.

2.2 Birth-death process

The second important step towards a queueing model is to define the birth and death process. The class of the birth-death process forms a part of the Markov chains and processes. Then, the birth-death process can be characterized by Markov chains and process that allow transitions between neighboring states only. These process state changes correspond with a population size that will either increase or decrease through birth or death. In the state changes, births occur with birth rate λ_k and deaths occur with death rate μ_k. Both rates of the birth-death process are exponentially distributed, because the considered class lies within the area of the Markov theory. Therefore, the birth-death process can be represented with a Markov chain, as is shown in Figure 2.1. In the birth-death process, one can assume a pure birth process when μ_k=0. Moreover, if one considers a birth process with constant coefficients ($\lambda_k = \lambda$), the famous Poisson process is obtained. The state changes of the poisson process can be characterized by

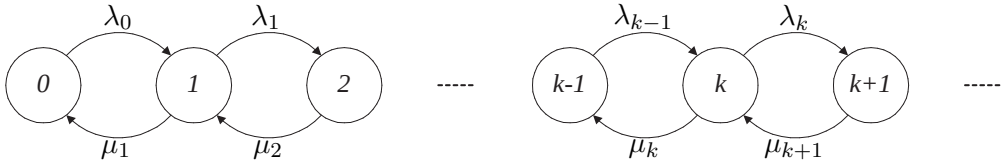


Figure 2.1: *Transition state diagram of birth and death process*

the following set of ordinary difference equations:

$$\frac{dp_k(t)}{dt} = -\lambda p_k(t) + \lambda p_{k-1}(t) \quad \forall\ k = 1, 2, \ldots \tag{2.3a}$$

$$\frac{dp_0(t)}{dt} = -\lambda p_0(t). \tag{2.3b}$$

In Cohen [Coh82], these equations are further analyzed. In this analysis, the set of equations (2.3) have been solved via induction, which results in the time-dependent Poisson distribution:

$$p_k(t) = \frac{(\lambda t)^k}{k!} \mathrm{e}^{-\lambda t}. \tag{2.4}$$

Thus, the time-dependent probability distribution of Poisson process has been derived from a birth process with constant birth rates. According to [Kle75], a birth process is relevant in queueing theory for two reasons. First, analytical or probabilistic properties can be simplified by e.g. the PASTA property. PASTA stands for 'Poisson Arrivals See Time Averages', see Wolff [Wol82]. Second, numerous natural physical and organic processes exhibit behavior that is probably meaningfully modeled by Poisson processes. Therefore, Poisson processes are used to describe arrivals of customers or lots to a service facility or machine.

Another possibility in the birth-death process is to assume a pure death situation when $\lambda_k = 0$. In queueing systems, a death situation may correspond to a service completion. Consequently, discussing a pure death situation is not relevant, since a service completion is not possible without arriving customers, products or events.

However, combining the pure birth and pure death processes with constant coefficients $\mu_k = \mu$ and $\lambda_k = \lambda$ is relevant, because the combination results in perhaps one of the simplest queueing model: the single server queue with exponential arrivals and departures. In Kendall [Ken53], a quite popular notation has been introduced for the Markov single server queue namely the $M/M/1$. Here, the 'M's stand for Memoryless or Markov arrivals and departures respectively and these occur at '1' present server. The $M/M/1$ forms the basis of the queueing models considered in this thesis. The time-dependent notation of an $M/M/1$ has been expressed by the forward Chapman-

Kolmogorov's equations:

$$\frac{dP_k(t)}{dt} = -(\lambda + \mu)P_k(t) + \lambda P_{k-1}(t) + \mu P_{k+1}(t) \quad \forall\ k = 1, 2, \ldots \tag{2.5a}$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t). \tag{2.5b}$$

These equations can be derived using the flow diagram in Figure 2.1. [Kle75] refers to (2.5) as a set of differential-difference equations. In this thesis, (2.5) is seen an infinite number of ordinary differential equations (ODEs).

2.3 Limiting behavior

It is very complicated to obtain the time-dependent solution of the forward Chapman-Kolmogorov's equations. Therefore, the much easier to analyze limiting behavior is discussed first. Limiting behavior is also be referred as steady state, or equilibrium, behavior. Logically, equilibrium behavior holds for $t \to \infty$:

$$p_k = \lim_{t\to\infty} P(X(t) = k) = \lim_{t\to\infty} P_k(t). \tag{2.6}$$

Thus, limiting behavior isolates the system dynamics and basically, limiting behavior is the opposite of transient behavior.

Definition 2.1. Transient and recurrent
The Markov process is transient if the state can only be visited a finite number of times. Otherwise, the state is recurrent.

The above stated definition results in time-independency for limiting behavior, while transient behavior only occurs during a certain time. Transient behavior stops when the time of steady state has been reached. Although transient behavior is of most interest in this research, limiting behavior is important for several reasons. Limiting behavior shows to which equilibrium transient behavior leads. Besides, research on limiting behavior is not as complicated as time-dependent behavior. Finally, steady state behavior gives insights of transient behavior. The limiting distribution can be computed with the forward Chapman-Kolmogorov's equations and (2.6). However, a limiting distribution only exists when certain conditions have been fulfilled. First, the Markov process has to be irreducible.

Definition 2.2. Irreducible
In Kulkarni [Kul99], a Markov process $\{X(t), t \geq 0\}$ is said to be irreducible if the corresponding Markov chain is irreducible. A Markov chain $\{X_n(t), n \geq 0\}$ on state space $S = \{1, 2, \ldots, N\}$ is said to be irreducible if $\forall\ i, j$

$$P(X_k = j | X_0 = i) > 0.$$

The birth-death process as shown in Figure 2.1, shows that all states are adjacent and transitions are possible between neighbors only. Thus, all states can be reached via each other, which means that no absorption takes place in a certain state. Therefore, the birth-death process is an irreducible Markov process. Second, the Markov process has to be positive recurrent.

Definition 2.3. Positive recurrent
A Markov process is said to be positive recurrent, if the corresponding Markov chain has a finite mean return time to a recurrent state.

The positive recurrence condition affects the quotient between input rate and output rate. Intuitively, when the input rate is larger than the output rate, the system expands. A continuously expanding system results in an increase in states to infinity. After some time, the system never runs empty anymore and not all states can be reached. Not being able to reach all states results in the following condition, $\lambda > 0$ and $\mu > \lambda$. Obviously, positive recurrence restricts the process to utilizations less than '1', i.e. $\rho < 1$.

Given that an $M/M/1$ system is irreducible and that $\rho < 1$, the equilibrium behavior can be computed. An approach to compute the limiting distribution is to take (2.5) and set $\frac{dP_k(t)}{dt} = \frac{dP_0(t)}{dt} = 0$, which leads to the following stationary expression:

$$\begin{aligned} 0 &= -(\lambda+\mu)P_k(t) + \lambda P_{k-1}(t) + \mu P_{k+1}(t) \quad \forall\ k = 1, 2, \ldots \\ 0 &= -\lambda P_0(t) + \mu P_1(t). \end{aligned}$$

Along with the sum of all probabilities,

$$\sum_{k=0}^{\infty} P_k(t) = \sum_{k=0}^{\infty} p_k = 1, \tag{2.7}$$

the limiting distribution can be derived when the mean transition rates have been be computed. After some rearranging, a limiting distribution can be specified as:

$$p_k = (1-\rho)\rho^k \ \forall\ k \geq 0.$$

The limiting distribution function p_k can be used to obtain relevant performance measures as treated in Section 1.1. First, the steady state wip-level can be computed using the sum over all probabilities p_k times the number of the state in that probability:

$$\overline{w} = \sum_{k=0}^{\infty} kP_k = \sum_{k=0}^{\infty} k(1-\rho)\rho^k = \frac{\rho}{1-\rho} \qquad \forall\ 0 \leq \rho < 1. \tag{2.8}$$

The flow time can be computed with the use of Little's Law [Lit61]. Little's Law ($\overline{w} = \overline{\delta}\ \overline{\varphi}$) links the WIP-level, throughput and flow time for steady state. The wip-level has been stated above and the throughput in steady state is the same as the input rate. So, with the aid of Little's law the equilibrium flow time results in,

$$\overline{\varphi} = \frac{1}{\mu(1-\rho)}.$$

The denominator contains the $(1-\rho)$ term implying that flow time goes to infinity when the utilization approaches '1'. Therefore, in the models of this thesis, utilizations will not exceed '0.9'.

These relations can be derived also via the Pollazcek-Khinchin formula [Khi32, Pol30]. The Pollazcek-Khinchin formula is an approximation of the $G/G/1$ queue, which is a queue with generally distributed arrival and process rates G:

$$\overline{\varphi} = \left(\frac{c_a^2 + c_p^2}{2}\right)\left(\frac{\rho}{1-\rho}\right)\frac{1}{\mu} + \frac{1}{\mu}. \tag{2.9}$$

The generally distribution uses coefficients of variation of arrival and process rates, c_a and c_p respectively. The variability coefficient is defined as: $c = \sigma\mu$. From this formula, the exact solution can be determined for a $M/G/1$ queue, Thus, the exact solution for the memoryless system can be derived using $c_a^2 = c_p^2 = 1$. In case of deterministic arrival and process rates $(D/D/1)$ the exact behavior can found with $c_a^2 = c_p^2 = 0$ and common sense, of course.

There are even more ways to determine limiting behavior. For example, limiting behavior characteristics have been obtained by applying the 'mean value approach'. This approach uses the PASTA property, some other approaches can be found in Adan [Ada02].

2.4 Time-dependent behavior

Instead of looking for limiting behavior, this section considers transient behavior to capture the dynamics of the $M/M/1$ queue. Gathering time-dependent relations is very complicated for even the most simple systems. However, continuous relations are necessary for using controllers and this Markov theory forms the basis. Therefore, studying the basics of deriving transient behavior can be important to give insight in the $M/M/1$ process.

The derived of time-dependent behavior has been treated shortly in Takacs [Tak62]. In this thesis, the time-dependent behavior derivation will be treated more extensively. The first step contains rewriting the set of ordinary differential equations (2.5) using Z-transformation. Basically, Z-transformation is a domain transition from discrete to continuous in which $|z| < 1$ holds. The Z-transform is defined as,

$$P(z,t) = \sum_{k=0}^{\infty} P_k(t) z^k. \tag{2.10}$$

Of (2.5), the k^{th} differential equation is multiplied by z^k. Now, (2.7) has been used together with properties of the Z-transform, see [Kle75]. After some rearranging, the result leads to the following expression:

$$z\frac{\partial}{\partial t}P(z,t) = (1-z)\big[(\mu - \lambda z)P(z,t) - \mu P_0(t)\big]. \tag{2.11}$$

On this equation a Laplace transformation has been applied, to lose the time-derivative. A Laplace transform makes a conversion from the time domain to the frequency domain. Besides, an initial condition has been used, which is defined as $P(z,0^+) = z^i$. The initial condition specifies the initial number of events i in the queue. With the formulation in the Laplace domain and the initial condition, the following important expression can be obtained:

$$P^*(z,s) = \frac{z^{i+1} - \mu(1-z)P_0^*(s)}{zs - (1-z)(\mu - \lambda z)}. \tag{2.12}$$

The expression of $P^*(z,s)$ will be used later, in Chapter 4. Sofar, Equation (2.12) cannot be used, since $P_0^*(s)$ is unknown. Therefore, Rouche's theorem has been applied to find the relation of $P_0^*(s)$. In the unit circle $P^*(z,s)$ has to be finite, since the sum of all probabilities is '1'. Therefore, the root of the denominator has to be root in the numerator, leading to the same zeros. Consequently, with Rouche's Theorem the following relations have been obtained:

$$P_0^*(s) = \frac{\zeta_1(s)^{i+1}}{\mu(1-\zeta_1(s))} \tag{2.13}$$

where,

$$\zeta_1(s) = \frac{(\lambda + \mu + s) - \sqrt{(\lambda+\mu+s)^2 - 4\lambda\mu}}{2\lambda}.$$

With (2.12) and (2.13), the time-dependent relation can be completed for an $M/M/1$ queue by applying inverse Laplace and Z-transformations. However, from this point on, the calculations of inverting Z-transformation and Laplace transformation get very complex. Only the final solution is relevant. Therefore, more detailed calculations can be found in in Appendix A. This leaves the final time-dependent solution of Equation (2.5) as,

$$P_k(t) = \mathrm{e}^{-(\lambda+\mu)t}\Bigg[\rho^{(k-i)/2} I_k(2t\sqrt{\lambda\mu}) + \rho^{(k-i-1)/2} I_{k+i+1}(2t\sqrt{\lambda\mu}) + (1-\rho)\rho^k \sum_{j=k+i+2}^{\infty} \rho^{-j/2} I_j(2t\sqrt{\lambda\mu})\Bigg]. \tag{2.14}$$

Herein, I_ν is the modified Bessel function of the first kind. Solutions of the modified Bessel equation have been formed by the modified Bessel function. The modified Bessel equation has been expressed as:

$$z^2\frac{d^2y}{dz^2} + z\frac{dy}{dz} - (z^2 + \nu^2)y = 0.$$

The fundamental set of solutions for non-integer ν are defined as:

$$I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{z^2}{4}\right)^k}{k!\Gamma(\nu+k+1)}.$$

Furthermore, in (2.14) an initial buffer occupancy is defined by i. With $P_k(t)$, the wip-level can be computed by taking an infinite sum over all k, just like the steady state:

$$\overline{w}(t) = \sum_{k=0}^{\infty} k\, P_k(t) = \sum_{k=1}^{\infty} k\, P_k(t).$$

The full relation has not been expressed, since it leads to a formulation similar to Equation (2.14) see Appendix A. However, when the derivative of (2.4) is taken, more understandable relations appear. After some calculations and rearranging, it results in:

$$\begin{aligned} \frac{\partial \overline{w}(t)}{\partial t} &= \sum_{k=1}^{\infty} k \frac{\partial P_k(t)}{\partial t} \\ &= \lambda - \mu(1 - P_0(t)). \end{aligned} \tag{2.15}$$

With this relation, the throughput can be computed, since the input rate is a Poisson process with rate λ. Consequently, the throughput can be expressed in (2.17). The throughput expression seems very logical, since the output rate is the processing rate times the probability of being non-idle. However, $P_0(t)$ has not been determined yet. Therefore, when (2.13) is rearranged to the following expression the solution can be determined:

$$P_0^*(s) = \frac{\zeta_1(s)^{i+1}}{\mu(1 - \zeta_1(s))} = \frac{1}{\mu} \sum_{k=1+i}^{\infty} \zeta_1(s)^k = \frac{1}{\mu} \sum_{k=1+i}^{\infty} \rho^{-k} \zeta_2(s)^{-k}.$$

In Erdelyi, Magnus, Oberhettinger and Tricomi [Erd54], the inverse Laplace transform for $\zeta_2(s)^{-k}$ has been expressed. The following equation is the result of applying inverse Laplace:

$$P_0(t) = \frac{\mathrm{e}^{-(\lambda+\mu)t}}{\mu t} \sum_{k=i+1}^{\infty} k \rho^{-k/2} I_k(2\mu t \sqrt{\rho}). \tag{2.16}$$

Now, all the required behavior of the manufacturing system has been obtained for the $M/M/1$ queue. The time-dependent probability distribution function $P_k(t)$ has been derived for every state k in (2.14). Furthermore, (2.16) presents an easier relation for $k = 0$. Both equations consist of an $\mathrm{e}^{-(\lambda+\mu)t}$ term and term (s) with a Bessel's function, $I_k(2\mu t\sqrt{\rho})$. These term keep each other in balance when steady state has been reached for $t \to \infty$.

When the relation of $P_0(t)$ is examined further, a remarkable property can be found. If λ is substituted with $\rho\,\mu$, all terms of time are directly linked with the process rate; leading to the process rate independency. Basically, the relation between process rate and time can be seen as a general property. When the process rate is doubled, the number of lots can be processed in half the time. Herein, $\mu\, t$ has the same value. Therefore, the investigation of theory of stochastic processes and especially Markov chains and processes have become of interest, since it can be used to derive the process rate property.

Property 2.1 (Process rate)**.**
For every state k, the probability distribution $P_k(t)$ can be formed such that the process rate μ is proportional with time. Consequently, the curve of $P_k(t)$ depends on $\mu\, t$. The proportionality of the process rate and time lead to process rate property. Herein, time can be scaled to every process rate without extra computation effort.

For example, processes with different process rates can have identical wip-levels in dimensionless time, when time has been multiplied with process rate under the condition of a similar utilization. If process B has a process rate that is twice as much as the other, then the wip-levels will be similar when the time of process B is multiplied with 2.

When other models have been formulated in this thesis, it is possible to use Property 2.1 to approve $M/M/1$ models. In Section 3.3, other properties have looked for in area of scaling.

(2.16) can be used also for expressing performance measures. So, the following time-dependent relation can be derived for the throughput:

$$\overline{\delta}(t) = \mu\big(1 - P_0(t)\big) = \mu - \frac{\mathrm{e}^{-(\lambda+\mu)t}}{t} \sum_{k=i+1}^{\infty} k\, \rho^{-k/2} I_k(2\mu t\sqrt{\rho}). \tag{2.17}$$

Moreover, another relation for the wip-level and a relation for the flow time can been obtained. These can be derived from the throughput using Figure 1.1. If the input-event are seen as $\lambda\, t$ and the output events can be seen as the integral of the throughput, then the following relations can be derived:

$$\overline{w}(t) = \lambda\, t - \int_0^t \overline{\delta}(\tau) d\tau \tag{2.18}$$

$$\overline{\varphi}(t) = t - \frac{1}{\lambda} \int_0^t \overline{\delta}(\tau) d\tau. \tag{2.19}$$

For these performance measures, time-dependent behavior can be visualized. To test the applicability of these performance measures, Example 2.1 has been performed.

Example 2.1 ($M/M/1$ throughput for $\rho = 0.5$)**.**
A single server queue with exponential arrival and process rate with infinite buffer places or the $M/M/1$ queue has been considered for this example.
The used time-dependent probability relations are stated in (2.14) and (2.16). However, time-dependent probability functions are not of interest. Therefore, from (2.16), performance measures have been derived for the throughput (2.17), wip-level (2.18) and flow time (2.19). The wip-level and flow time are left behind in this example, since the throughput forms a big part of the wip-level and flow time. For the $M/M/1$ queue, the chosen arrival rate is $\lambda = 0.5$ and the chosen process rate is $\mu = 1.0$. With (1.1), the utilization becomes $\rho = 0.5$. Moreover, the system is initial empty.
An initial empty system leads to an initial empty throughput. The expected limiting

behavior can be computed with Section 2.3. Herewith, the steady state throughput can be derived, which results in the input rate, $\delta = \lambda$. Therefore, the example should show an initial zero throughput, which evolves towards $\delta = 0.5$. With Mathematica, (2.17) has been plotted as is done for the throughput in Figure 2.2.

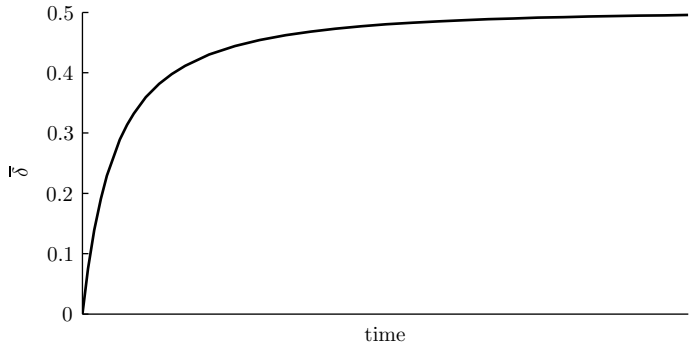


Figure 2.2: *Throughput for* $\rho = 0.5$ *and* $\mu = 1.0$

Herein, the dimensionless time versus the throughput has been illustrated. As expected, the throughput starts in zero and reaches the steady state value of $\delta = \lambda = 0.5$. In all, (2.17) gives a throughput relation which is satisfying and when integration of the throughput has been performed, the flow time and wip-level can be obtained.

2.5 Résumé

The derivation of the $M/M/1$ queueing model has been treated in this chapter. The derivation of the $M/M/1$ model started with the basics of stochastic processes. Within the stochastic processes, Markov chains and processes have been distinguished, which restrict the involved processes to exponential distributions. Furthermore, Markov chains and processes have been detailed even more with the birth-death process, which restrict transition changes to neighboring states only. Finally, constant birth and death rates are introduced in the birth-death process to obtain the $M/M/1$ model.

The developed $M/M/1$ queueing model includes a distribution function from state probabilities only. In manufacturing, other measures are of great interest. Therefore, performance measures have been derived from the $M/M/1$ probability distribution functions. These performance measures have been derived with probability distribution functions of limiting and time-dependent behavior. The achievement of limiting distribution functions is quite straightforward, while the derivation of time-dependent behavior is very complex. This complex derivation has been treated entirely to gain more insight of the $M/M/1$ queue. Finally, the solution of the time-dependent derivation has been used to define the process rate property, in which time is proportional with the process rate.

In all, an analytical model of the $M/M/1$ queue has been presented and derived in this chapter. The $M/M/1$ model will be used as a validation model later in this thesis. Furthermore, the achieved insight during the derivation of the $M/M/1$ model will be used to determine a transfer function in Chapter 4. Finally, Chapter 5 uses the $M/M/1$ model to arrange ODE relations of the finite buffer.

Chapter 3

Discrete Event Modeling

In the previous chapter, Markov theory has been used to develop a queueing model of a manufacturing system. The considered manufacturing system is an $M/M/1$ queue. The behavior of the $M/M/1$ queue has been distinguished in two types, limiting and time-dependent behavior. The relation of time-dependent behavior showed an interesting proportionality between time and the process rate. The proportionality of time and the process rate resulted in the specification of the process rate property, see Property 2.1. The realization of the process rate property showed that deriving a time-dependent relation via queueing theory is very complicated for even the most simple systems.
To avoid complex derivations, this chapter considers computer simulation models and especially the Discrete Event Model (DEM). The DEM is very suitable for simulating manufacturing systems. After simulating, it is common to perform an analysis of the simulation output data to obtain the desired data. The data of interest is transient and therefore Section 3.1 treats transient analysis of DEMs. After a proper transfer analysis, the simulations itself can be set-up, see Section 3.2. Finally, Section 3.3 will use the transient analysis and simulation knowledge to discuss scaling properties of manufacturing systems.

3.1 Transient analysis

In Law and Kelton [Law00], two types of DEM evaluation methods are described. The first method deals with transient analysis, which is used for time-dependent behavior. The second method is steady state analysis, which is independent of time and the initial transient may not participate. The focus will be put on the first method, because the considered control strategy requires transient analysis. Transient analysis excludes equilibrium behavior of manufacturing systems. However, the start time of equilibrium behavior cannot be determined explicitly. The hardness to obtain this start time is illustrated in Figure 3.1 , where an example of a transient analysis is shown for the wip-level of a DEM output. The DEM output has been averaged and stochastic behavior

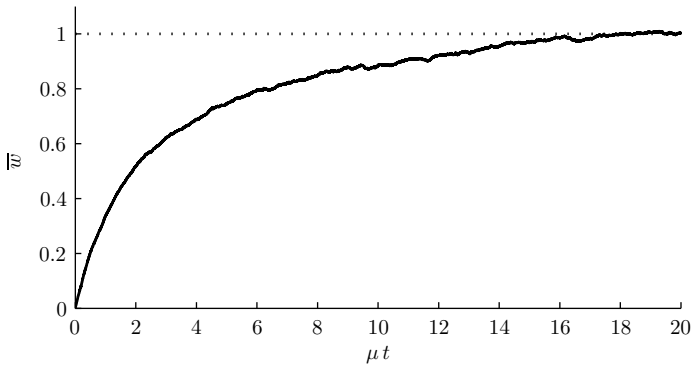


Figure 3.1: *Initial transient and steady state*

restricts the determination of the start time of steady state and the end time of transient state. Consequently, a certain end time has been chosen close the steady state indication to avoid superfluous computation time.

Thus, transient behavior ends when the slope reaches 0. In theory, the horizontal asymptote holds for $t \to \infty$ in practice though, this time cannot be observed explicitly. Consequently, simulations are run to a certain time when it can be visually observed that performance measures flatten.

Performance measures of simulation output

A sample of input and output events is shown from a single DEM simulation of an $M/M/1$ queue in Figure 3.2. This figure indicates the flow time and wip-level, which can

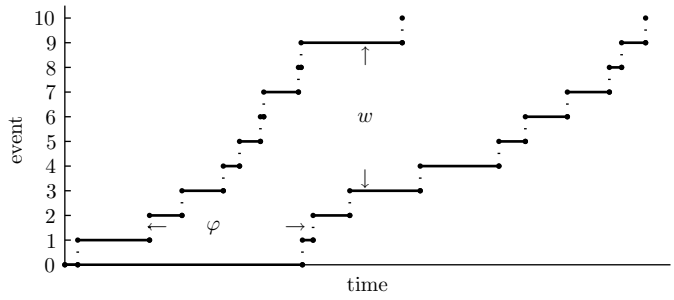


Figure 3.2: *Sample of simulated events*

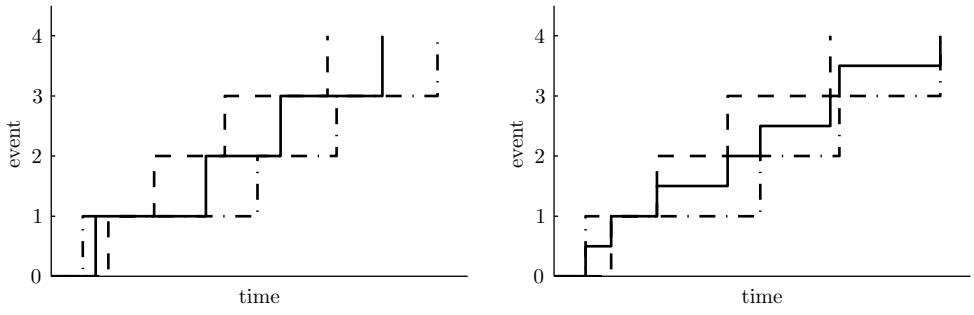
be computed with figure data. Furthermore, the figure data can be used to determine the throughput by computing the output events derivative. One complication though, the derivative of output events results in an infinite throughput at an event change time. All other times have a zero throughput which result in a throughput that becomes '0' almost everywhere. Consequently, a throughput computation requires a sample time and multiple simulations, since variability can give complete different outputs of two

single simulations. When multiple simulations are taken a correlation occurs. This correlation can be obtained by taking averages of various simulations to compute the desired performance measure.

Averaging of output

Normally, averaging comes down to the standard procedure of dividing the sum of values by the number of considered values, $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. However, the standard procedure can give problems, since the discontinuity of discrete events result in any data between events. With this lack of data, one can consider two possible averages methods, namely 'event averaging' and 'time-averaging'.

The first method, event averaging, takes the mean of events at every point in time of the simulation clock. This mean results in an average at every time unit. The only problem of this method occurs when two or more events happen at the same time. For example, event i ends and event $i+1$ starts at the same time. The question arises which event has to be used in the average. In this study, only the starting points of events are used. Consequently, a graph can be drawn as in Figure 3.3(a). Herein, the policy is to take the mean of $t_{i,1}, t_{i,2}, \ldots, t_{i,n}$ for n simulations at event i. The second method



(a) *Method 1: time-averaging* (b) *Method 2: event averaging*

Figure 3.3: *Average (–) of simulation 1 (-·) and simulation 2 (- -)*

averages times of every event number. These events occur at a certain time τ. The τ values should be averaged for every event i and then Figure 3.3(b) can be obtained.

Although the averages of these methods look very different in Figure 3.3, the expectation of both methods is to result in the same averages for large numbers of simulations. Appendix C shows that for larger simulation amounts the averages converge to each other. But, is there a better method or which method is preferred? Event averaging has the advantage of the possibility of making it 'real time', since one does not have to wait until an event is finished to take the mean. So, the history can be kept up-to-date immediately with event averaging. The greatest disadvantage of time-averaging occurs when the wip-level is taken from averaged output. The averaged output shows 'jumping' of the wip-level to the nearby levels, see Figure 3.4. Basically, the jumping is the result

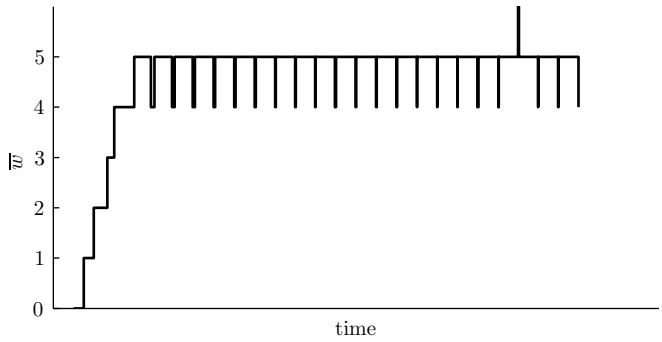


Figure 3.4: *Jumping of wip-level*

of a single event step which logically changes by '1', while time units change by 10^{-5} in χ. In brief, event averaging is preferred, because event averaging can be performed 'real-time' and has a finer grid.

3.2 Discrete Event Simulation

The DEM is simulated by letting it evolve over time. In this time, state variables change instantaneously at separate points in time. Thus, changes can only be made at a countable number of points in time. An event is represented by that instant change and time is represented by a simulation clock. The simulation output data will be a sequence of times at which an event occurs. One could imagine that the amount of data that has to be stored and manipulated in a simulation can become very large. Therefore, computers are used to perform a simulation of a DEM.

$M/M/1$ model

In this thesis, DEMs are simulated using $\chi\,0.8$, see Vervoort and Rooda [Ver03]. Since an analytical solution for the $M/M/1$ has been derived earlier, this queueing system will also be considered here. The DEM of an $M/M/1$ queue has been divided in several straightforward processes.

The first process is the generator, which create events that can represent customers, products, lots etc. This generator (G) feeds, with an exponentially distributed rate λ, the infinite buffer (B). A nonempty buffer supplies an idle machine (M) with an event. The supplied event will be processed with exponential rate μ on Machine (M). Finally, machine (M) leads the processed event to exit (E) process, where the event leaves the system. In Figure 3.5, a schematic overview is given of the structure of the $M/M/1$ or $GBME$ queue. The combination of a buffer and machine (BM) is also referred as a workstation (W). The occupation of workstation changes in a transient analysis simulations. A simulation with increasing workstation occupancies is referred

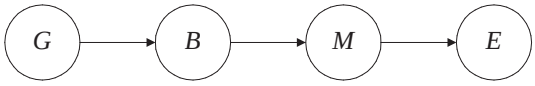


Figure 3.5: *DEM structure of the $M/M/1$*

to a ramp-up simulation, otherwise a ramp-down simulation occurs. Both simulations can be performed with χ to obtain transient occupancies and wip-levels. Steady state wip-levels can be computed with definition of (2.8) and DEM simulation of the $GBME$ queue. Finally, the $GBME$ queue is specified with its χ code in Appendix B.

Comparison of analytical and simulation models

With the specification of the $GBME$ queue, a DEM-simulation can be performed to computed performance measures. These measures have been compared with the earlier derived analytical model in Figure 3.6. This figure shows the wip-levels in time for

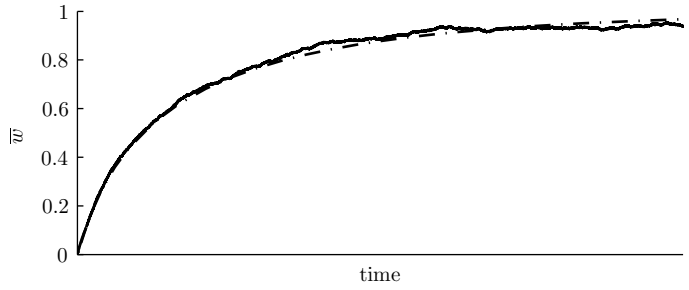


Figure 3.6: *WIP-levels of the DEM (–) and the analytical model (-·)*

both models and these models have almost identical wip-levels except for some DEM fluctuations. These irregularities are a result from variability and become less when the number of simulations increases. Increasing numbers of simulations lead to more computational efforts. The need for increasing the number of simulations is one of the reasons that a DEM can result in a computationally expensive situation. Even more fluctuations occur for the throughput, since a derivative is more sensitive for inconsistencies. To avoid these inconsistencies in a validation process, properties have been researched in this thesis. The next section deals with properties.

3.3 Scaling properties

A DEM simulation has to meet Property 2.1, where process rate independency is defined. Of course, process rate independency is a property that has to be satisfied by a DEM. Besides, a DEM can contain more corresponding properties than cannot be determined in the analytical model. Consequently, DEMs have involved in search for properties

that can be used to validate other modeling techniques. Furthermore, the property search can give more insight of manufacturing systems. Besides, properties can make simulations at e.g. different process rates redundant. The term properties is quite general. Here, the discussed properties are based on scaling and the following scaling properties have been proposed:

1. process rate scaling
2. utilization scaling
3. workstations scaling

Process time scaling

The first property of concern is a simple one. The analytical proof of the $M/M/1$ in (2.14) already showed that the process rate μ is proportional to time, such that $\mu \sim t$. Note that the process rate independency only holds for identical utilizations. Consequently, process times with constant utilizations and different process times can be scaled. Then, a DEM simulation meets the process rate property. Intuitively, one would think the process rate property holds for several identical single server queues in series. To validate this suggestion, a DEM has been set-up with 10 identical workstations.

This 10 workstation queue in series has been simulated for an utilization of $\rho = 0.5$ and process rates of $\mu_1 = 0.2$ and $\mu_2 = 0.1$. These process rates have been used to determine the wip-level in a scaled time with Property 2.1. The scaled wip-levels should result in an identical curve in dimensionless time for both situations, $\mu_i\, t$. In Figure 3.7, the described DEM experiments can be seen, where time has been made dimensionless

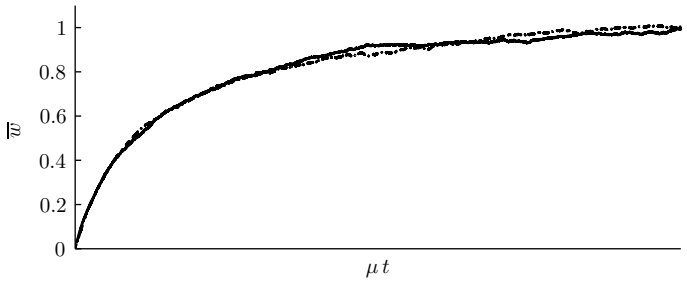


Figure 3.7: *Scaling of process rates* $\mu_1 = 0.2$ *(–) and* $\mu_2 = 0.1$ *(-·) for 10 identical* $M/M/1$*'s*

by multiplying it with the corresponding μ_i and plotting it against the wip-level. As expected, both simulation runs have identical curves, but sometimes the wip-levels differ due to variability.

Utilization scaling

The second suggested property of interest is the utilization scaling. Instead of process rate scaling, utilization scaling has not been determined in Chapter 2. Therefore, an approach to determine the utilization scaling factor has been chosen that uses several. One of those options used a trail and error factor to make graphs with a different utilization come to each other when plotted in e.g. wip-level versus time. A disadvantage of this option was the lack of fundamental strategy. Another option came forward in [Coh82]. Herein, asymptotic relations have been expressed for utilizations, $\rho < 1$, $\rho = 1$ and $\rho > 1$. In these asymptotic expressions for $\rho < 1$ and $\rho > 1$, a time dependent correction term decreases mainly exponentially with:

$$\tau_r = \frac{1}{\mu(1-\sqrt{\rho})^2} \qquad \forall\ \rho < 1.$$

This term τ_r has been denoted as the relaxation time of the $M/M/1$ queue. The relaxation time gives an idea about the speed of approach to the stationary situation. Consequently, the relaxation time is proposed as an estimator for scaling time versus e.g. wip-level at different utilizations. The advantage of the relaxation time option is that the relaxation time has a fundamental background.

Two simulation runs have been performed to give an indication about the effectiveness of the relaxation time for the utilization property. Both experiments consist of a single server $M/M/1$ queue in which the process rate has been set-up as $\mu = 1.0$. Of course, the utilization differs in the experiments. The considered utilizations have a value of $\rho = 0.5$ and $\rho = 0.75$, since the relaxation is an approximation a limited distinction has been chosen. With these utilizations, the relaxation time becomes $\tau_r = 11.7$ and $\tau_r = 55.7$ respectively.

The expected behavior of the approximation is a difference in the initial transient behavior and a good approximation of the asymptotic behavior, since the relaxation considers asymptotic behavior. The experiment has to show how much the transient behavior differs from both simulations. Furthermore, in the experiment the wip-level has to be scaled as well, because $\overline{w} = \frac{\rho}{1-\rho}$ in steady state. So, that both corrected wip-levels have been indexed to one. The experiment is shown in Figure 3.8, where the indexed

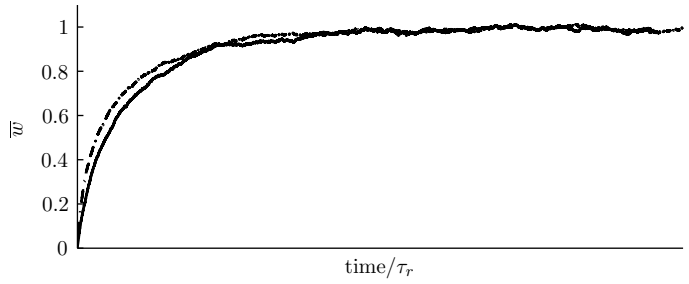


Figure 3.8: *Utilization scaling for* $\rho_1 = 0.5$ *(–) and* $\rho_2 = 0.75$ *(- ·)*

wip-level has been put out in time over the relaxation time. In the resulting graph, the indexed wip-levels correspond to each well except for the initial transient wip-levels. Shortly, the relaxation time is a reasonable estimator for the utilization property in which it can be used to validate other models. Herein, the other models can show a contrast in behavior which is not substantiality large.

Workstation scaling

Another question is whether or not it is possible to scale performance measures to the number of identical $M/M/1$ queues in series. Therefore, the third suggested property is the one of workstation independency. Scaling workstations seems to result in a trial and error process. However, a relation has been formed based on the fact that workstation n is responsible for $1/n$ of the total occupancy time influence of the whole system. This proposal leads to the following expression of the workstation time:

$$\tau_n = \frac{1}{\mu} \sum_{i=1}^{n} \frac{1}{i} \tag{3.1}$$

A test has been composed for the case of a single server $M/M/1$ queue and five $M/M/1$ queues in series. Both queues have the same utilization of $\rho = 0.5$ and the same process rate of $\mu = 1.0$. The experiment hes been performed for the wip-level in time, where time will be corrected with the workstation time τ_n. The wip-levels should result in $\overline{w} = 1.0$ and $\overline{w} = 5.0$ in steady state respectively. Consequently, the wip-level has been corrected by its steady state value and will be indexed at one. The wip-level correction should result in identical steady state values for both simulations. To anticipate on the transient behavior is more difficult, but both wip-levels should more or less in agree with each other.

The result of both cases is shown in Figure 3.9, where wip-level has been plotted in time.

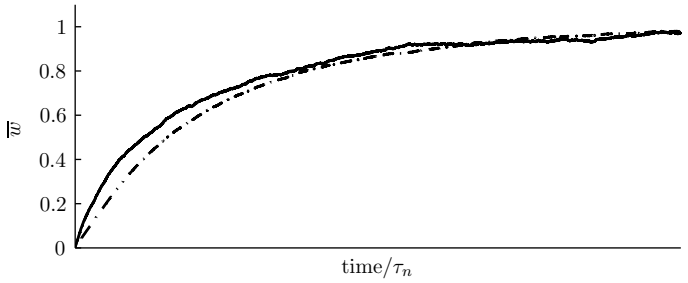


Figure 3.9: *Scaling of workstations with* $\rho = 0.5$ *where* $n_1 = 1$ *(–) and* $n_2 = 5$ *(-·)*

The accuracy of the approximation scaling appears to be in reasonable agreement for the transient wip-levels and (3.1) gives a nice approximation for showing the independency of the number of identical workstation in a queue in series. However, the accuracy is not ideal, since a lack of fundamental proof is present.

In all, a lot of possible properties come to mind for manufacturing systems. Unfortunately, an explicit determination and derivation is hard to obtain for scaling properties. The choice of scaling properties has proven that approximations can be found for different characteristics. Though, the validation effect and usability are not yet known, setting-up scaling properties is a step towards validating models using properties of manufacturing systems.

3.4 Résumé

This chapter deals with Discrete-Event Models (DEMs). The translation of DEM output data has been treated in which events are transformed to performance indicators. Furthermore, performance measures are discussed with respect to issues of averaging events of several simulations. Besides, some attention has been paid to the number of simulations to be run. Logically, when more simulation runs are performed, a higher accuracy of the average can be obtained.

The considered DEMs consisted of single and multiple server $M/M/1$ queues in series. Consequently, the $M/M/1$ queues have an infinite buffer and exponential arrival and process rates. The DEM variant of the $M/M/1$ queue has been referred as the $GBME$ queue. The $GBME$ queue will be used further in this thesis and can be seen as standard. Sometimes the standard $GBME$ queue requires adjustments and then these adjustments will be treated in the upcoming chapters. In Section 3.3, the $GBME$ queue has been used to search for properties of manufacturing systems. In the search for properties, three suggestions have been made. The first suggested property is the earlier proved process rate property and this property has been validated for several workstations in series. The second suggested property is the utilization property for which an approximation has been found in the relaxation time. Finally, a workstation property has been considered in which no exact solution has been found either. However, an approximation has been used which connects queues with a different number of workstations. The useability of these suggested properties have not been discussed. Obviously, the search of (scaling) properties can be continued with these ideas in mind.

In brief, this chapter presents the DEM as modeling technique for manufacturing systems. In Chapter 2, the presented modeling technique uses queueing theory, especially Markov chains and processed. The next chapter will be used to describe a third modeling type of this thesis, continuous models.

Chapter 4

Transfer Function

In the previous two chapters, two well-known modeling techniques have been discussed. This chapter shows another modeling method that can represent the dynamics of manufacturing system with a transfer function.
A transfer function can be seen as a linear differential relation between input and output signals. This linear relation can be obtained with two possible approaches. The first approach is capturing the dynamics via system identification, see e.g. Sage and Melsa [Sag71]. System identification is an experimental approach in which the transfer function can be computed using the output of a sine input signal. Obtaining the transfer function via system identification becomes very hard due to variability. Therefore, this chapter presents the second approach in which the transfer function has been determined analytically based on queueing models. In Section 4.1, queueing models of chapter 2 have been transformed with Taylor and Padé approximation into a transfer function. Finally, the obtained transfer functions will be validated for single server queues and queues in series in Section 4.2 and Section 4.3.

4.1 Analytical approach

By making a linear relation between input and output signals, dynamics of manufacturing systems can be represented. The relation between them is a transfer function $H(s)$ and is expressed below:

$$Y(s) = H(s)U(s). \tag{4.1}$$

The procedure is schematically shown in Figure 4.1 to obtain a dynamical model of the

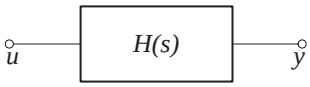


Figure 4.1: *Representation of the relation between input and output*

manufacturing system. The response of the transfer function can be computed with the *step* routine of Matlab. The step that has to be taken can be the input rate (λ) of the $M/M/1$. For example, a step from '0' to 'λ' on time '0' corresponds with initially empty buffers or, in other words, with a ramp-up simulation. Obviously, the response has to be the output rate, which is similar to the throughput (δ) in this case.

From this point on, the concept arises of determining a transfer function with the analytical solution derived in Chapter 2. To obtain the transfer function for input and output rate of the $M/M/1$, the wip-level is computed in the frequency domain;

$$\begin{aligned} P(z,t) &= \sum_{k=0}^{\infty} P_k(t) z^k \\ \frac{\partial P(z,t)}{\partial z} &= \sum_{k=0}^{\infty} k\, P_k(t) z^{k-1} \\ \frac{\partial P(1,t)}{\partial z} &= \sum_{k=0}^{\infty} k\, P_k(t) = \overline{w}(t) \\ \mathcal{L}\left(\frac{\partial P(1,t)}{\partial z}\right) &= \frac{\partial P(1,s)}{\partial z} = \frac{\lambda}{s^2} - \mu\left(\frac{1}{s^2} - \frac{P_0^*(s)}{s}\right). \end{aligned}$$

From these equations, the input rate is defined as λ/s and the output rate or throughput is defined as $\mu(1/s - P_0^*(s))$. So, with Equation (4.1) the following transfer function can be computed:

$$h^*(s) = \mu\left(\frac{s}{\lambda}\right)\left(\frac{1}{s} - P_0^*(s)\right) = \frac{1}{\rho}\left(1 - sP_0^*(s)\right). \tag{4.2}$$

However, this transfer function cannot be used, since transfer functions have to be linear ODE relations. In Laplace, a transfer function is the quotient of two polynomial relationsThe resulting problem lies in the square root of $\zeta_1(s)$ function, which is part of $P_0^*(s)$ in (2.13). Therefore, a Taylor series should be used to estimate a transfer function. The Taylor approximation has been performed around the steady state frequency $s = 0$. Logically, steady state is constant and therefore the steady state frequency becomes $s = 0$. Around the steady state frequency, the Taylor approximation becomes:

$$\mathcal{T}_n(s) = h^*(0) + h^{*\prime}(0)s + \frac{1}{2!}h^{*\prime\prime}(0)s^2 + \cdots + \frac{1}{n!}h^{*(n)}(0)s^n + \cdots . \tag{4.3}$$

Yet, the output of the dynamical model cannot be computed, since the output of the model requires a transfer function that has to be the ratio of two polynomials. So, the function needs a numerator and a denominator. The following step consists of the approximation of the numerator and denominator of a transfer function. Obtaining the numerator and denominator is an obstacle that can be conquered by using a Padé approximation. Initially, the approximation assumes that every polynomial function can be written as a function whose singularities are poles of the ratio of two polynomials. Of course, most functions have singularities which are not poles, therefore the previous

stated Taylor approximation has been performed. The general specification of the Padé approximation is expressed as,

$$f_{n+m+1}(z) = \frac{P_n(z)}{Q_m(z)} = \frac{p_n z^n + \cdots + p_2 z^2 + p_1 z + p_0}{q_m z^m + \cdots + q_2 z^2 + q_1 z + q_0}.$$

Herein, coefficients $p_0, p_1, \ldots, p_n$ and $q_0, q_1, \ldots, q_m$ can be computed by an $n+m+1$ order Taylor series.

Before determining a specific Padé approximation, two conditions are required that the approximation has to meet. First, when a step of λ is put on the transfer function the response at $t = 0$ has to be '0'. Second, for $t \to \infty$ the response has to be λ. Given these conditions and rewriting them results in the following expression:

$$\begin{array}{rcccl} \lim\limits_{t\to\infty} \overline{\delta}(t) & = & \lim\limits_{s\downarrow 0} \overline{\delta}(s) & = & \lambda \\ \lim\limits_{t=0} \overline{\delta}(t) & = & \lim\limits_{s\to\infty} \overline{\delta}(s) & = & 0. \end{array} \tag{4.4}$$

Anticipating on these conditions respectively, the approximation has to meet $\frac{p_0}{q_0} = 1$ and the numerator has to be of a lower order $m > n$.

Example 4.1 (Derivation of a 2^{nd} order transfer function)**.**
In this example, a transfer function will be composed using (4.2). This equation has been derived with Markov theory for the $M/M/1$ queue. A transfer function will be determined in this example with a denominator of the second order. Consequently, the Padé structure of the desired transfer function results in:

$$\mathcal{T}_4(s) = \frac{P_1(s)}{Q_2(s)} = \frac{p_1 s + p_0}{q_2 s^2 + q_1 s + q_0}.$$

The next step towards a transfer function is to compute the Taylor series $\mathcal{T}_4(s)$, which has been done using Maple. However, the ability of solving the Taylor series depends on two assumptions, which are quite trivial. First, the arrival rate (λ) has to be positive, otherwise products flow backwards or they do not flow at all. Second, the process rate (μ) has to be greater than the arrival rate, because the utilization has to be less than one. Now, the solution for the Taylor series becomes:

$$\mathcal{T}_4(s) = 1 + \frac{1}{-\mu+\lambda} s - \frac{\mu}{(-\mu+\lambda)^3} s^2 + \frac{(\lambda+\mu)\,\mu}{(-\mu+\lambda)^5} s^3 + O\left(s^4\right) \quad \forall\ \mu > \lambda > 0,$$

which is the approximation of Equation (4.2). With these series, the following Padé approximant has been obtained:

$$H_2(s) = \frac{\frac{\lambda+\mu}{(\mu-\lambda)^2} s + 1}{\frac{\mu}{(\mu-\lambda)^3} s^2 + \frac{2\mu}{(\mu-\lambda)^2} s + 1}.$$

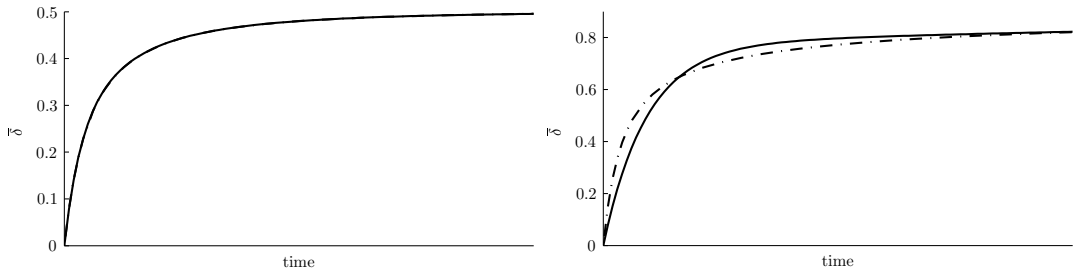
Obviously, the example has been taken just for a second order approximation to maintain the overview of the example. Of course, one can take higher orders for Padé and Taylor. A higher order will lead to an awful lot of terms which are not significant to show for this example.

The derived transfer function of Example 4.1 satisfies conditions of (4.4). Moreover, the transfer function meets the scaling property as defined in Section 3.3. The property can be found easily by substituting λ with $\rho\,\mu$ and then the powers of μ and s have to be equal.

4.2 Results for single $M/M/1$

After being able to identify the $M/M/1$ system, the performance of the transfer function has to be investigated. For the performance evaluation, it turned out that the performance of a 2^{nd} order approximation as in Example 4.1 was not satisfying enough. Therefore, a 6^{th} order Padé approximation is composed that meets the criteria. Example 4.2 has been set-up to compare the obtained transfer function with the analytical model.

Example 4.2 ($M/M/1$ queue)**.**
This example will be used to validate a transfer function that will be approximated with the use of earlier defined Markov model for the $M/M/1$ queue. The analytical solution of (2.17) has been used to validation the transfer function approximation. The transfer function has been obtained on a similar way as in Example 4.1. However, a sixth order transfer function has been created to increase the accuracy of the approximation. The experiment has been performed for two different utilizations of $\rho = 0.5$ and $\rho = 0.9$. The considered process rates are $\mu = 1.0$ for both cases, resulting in arrival rates $\lambda = 0.5$ and $\lambda = 0.9$ respectively.
The output of the transfer function has been generated with putting a step λ on the input. So, the step λ will be put on a sixth order Padé approximation with the use of the *step* routine of Matlab. In Figure 4.2, the result is shown for the throughput



(a) $\rho = 0.5$ (b) $\rho = 0.9$

Figure 4.2: *Throughput for analytical (-·) and Padé transfer function (–)*

in time. Furthermore, the throughput of the analytical solution of (2.17) has been shown in time as a validation model. Figure 4.2(a) shows the result for an utilization of $\rho = 0.5$. Herein, both experiments are identical, since the throughputs overlap for both situations. The other graph, in Figure 4.2(b), shows a difference between the approximated transfer function model and the analytical solution of the $M/M/1$ Markov

model. Obviously, the difference is a result of the approximation. Shortly, the Padé approximation has its best accuracy for the lower utilization.
For the situation of Figure 4.2(b), the wip-level and the flow time have also been computed to see the difference on another performance measures. Besides, the wip-level and flow time have computed to see the effect of required integration of throughput, see (2.18) and (2.19). Using this integration together with the input, these performance measures have been computed. Figure 4.3 shows the wip-level only, since the flow time

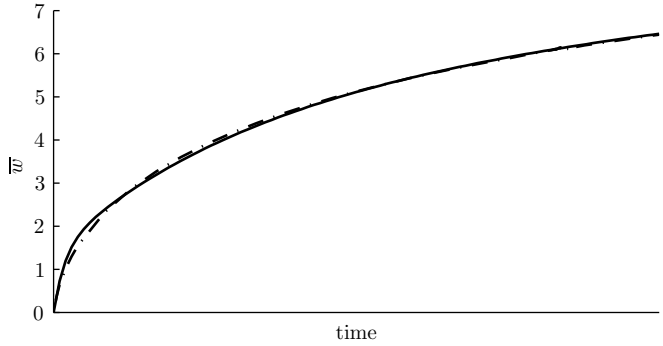


Figure 4.3: *wip-level of the analytical model (-·) and Padé transfer function (–)*

results in a similar graph. In Figure 4.3, the inaccuracy of the throughput in Figure 4.2(b) is only marginally influenced by the required integration of the throughput to obtain the wip-level. Furthermore, the focus of Figure 4.3 has been put on the beginning of the wip-level in time, otherwise the difference cannot be seen clearly.
In conclusion, although the accuracy of higher utilizations can be put on the line for the throughput, the accuracy of the wip-level and flow time seem to be more satisfying.

In brief, the transfer function has been validated in this section. Herein, the transfer function model has been created with the queueing theory solution for the $M/M/1$ in (2.17). The results of the Padé approximations have been satisfying, especially for utilizations below $\rho = 0.9$. The satisfaction of the approximated transfer function raises the question of the possibility of expanding the transfer function to several $M/M/1$ queues in series.

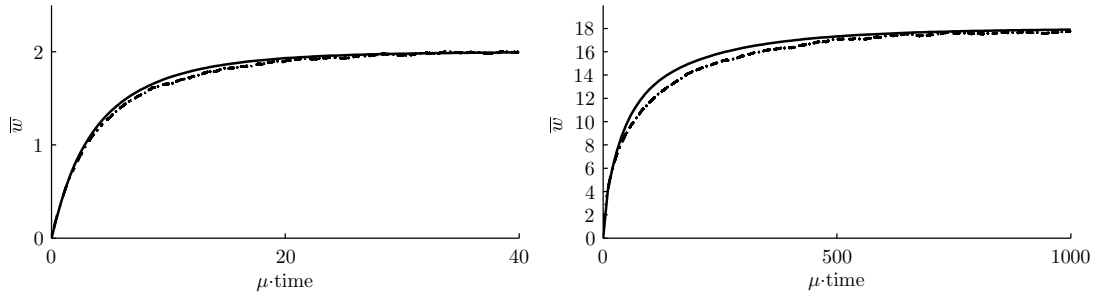
4.3 Results for $M/M/1$ in series

The Padé approximation of the $M/M/1$ has to be converted for queues in series, since the single server transfer function changes for a queue in series. Fortunately, the single server transfer function describes the behavior of every individual server of queues in series. Therefore, the transfer function of queues in series can be obtained by multiplying transfer functions for every considered single server in series. The required multiplication can not performed with the Padé approximation, because terms get lost due to the Padé approximation. Appendix D shows that terms get terms lost. To avoid lost

terms, the transfer function of queues in series has to be derived with a multiplied Taylor approximation. This Taylor outcome has to be approximated with Padé to obtain a transfer function of queues in series. The transfer function of a queue of n workstations can be determined with the n^{th} power of the Taylor approximation.

In this section, the results are compared of the developed transfer function with a DEM output, since an analytical solution for the $M/M/1$ has not been derived. Furthermore, this section examines the wip-level only, since the wip-level is the smoothest curve from the DEM simulation output. DEM simulations have been performed for the following three cases of serial queueing transfer functions: the five-in-line queue, the identical and the non-identical tandem queue.

Example 4.3 (Identical tandem queue)**.**
In this example, an experiment has been performed which compares the approximated transfer function model with a DEM as a validation model. The approximated transfer function has been obtained from the sixth order transfer function for the $M/M/1$ queue. The Taylor series approximation has been powered to determined a transfer function with help of a sixth order Padé approximation.
The tandem queue contains two identical $M/M/1$ queues in series. Logically, a queue of two servers is the closest possible to the $M/M/1$. The experiment has been executed for two kind of utilizations $\rho = 0.5$ and $\rho = 0.9$. Moreover, both workstations have a process rate of $\mu = 1.0$, which results in a arrival rate of $\lambda = 0.5$ and $\lambda = 0.9$.
The experiment has been performed only for the wip-level, because the wip-level is the easiest to determine from a DEM simulation. For steady state, wip-levels for the simulations should result in $\overline{w} = 2$ and $\overline{w} = 18$, for $\rho = 0.5$ and $\rho = 0.9$ respectively. Figure 4.4 indicates the results of both examples, where wip-level in time has been



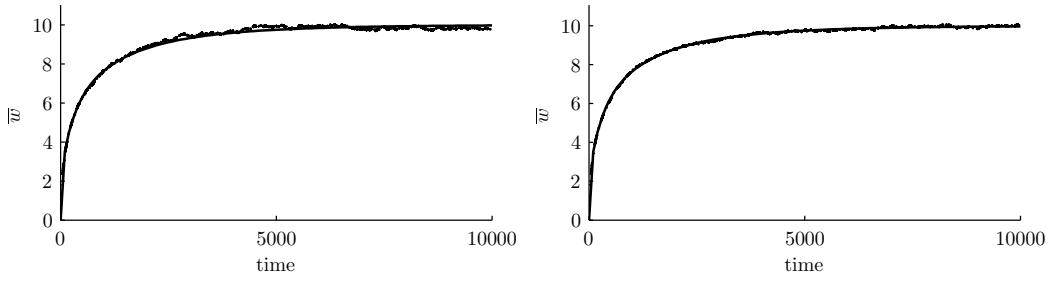
(a) $\rho = 0.5$ *with two workstations* (b) $\rho = 0.9$ *with two workstations*

Figure 4.4: *The wip-level of the DEM (··) and the Padé approximation*

plotted. As expected, the wip-levels reach the desired steady state values. For both utilizations, one can distinguish the DEM from the transfer function approximation. However, the accuracy of the transfer function model is better for an utilization of $\rho = 0.5$ than $\rho = 0.9$. From the wip-levels of Figure 4.4, it appears that the approximation holds quite well compared with the validation model of the DEM even for an utilization of $\rho = 0.9$.

In Example 4.3, the expansion of the single server transfer function model has been validated for identical servers with similar process rates. The accuracy of the approximation has been satisfying and accordingly another example has been set-up to validate the transfer function approximation on non-identical tandem queues.

Example 4.4 (Non-identical tandem queue)**.**
This example uses a DEM to validate the transfer function approximation of non-identical tandem queues. The non-identical tandem queue has two servers with different process rates, μ_1 and μ_2. For the non-identical case, two possibilities have been examined with different process rates $\mu_1 > \mu_2$ and $\mu_1 < \mu_2$. Consequently, a different utilization on each server is the result, since the same lots flow through the servers.
Therefore, the example has been performed two times, for utilizations $\rho_1 = 0.9$ with $\rho_2 = 0.5$ and for utilizations $\rho_1 = 0.5$ with $\rho_2 = 0.9$. Both experiments have been executed for an arrival rate of $\lambda = 0.1$. For the first experiment of $\rho_1 = 0.9$ with $\rho_2 = 0.5$, the combination of arrival rate and utilization results in process rates $\mu_1 = 1/5$ and $\mu_2 = 1/9$. For the second case of $\rho_1 = 0.5$ with $\rho_2 = 0.9$, the arrival rate sets the process rates on $\mu_1 = 1/9$ and $\mu_2 = 1/5$.
The transfer functions have been determined with two Taylor approximations for $M/M/1$ queue. One approximation derives a sixth order transfer function for $\rho = 0.9$ via Padé approximation. The other sixth order approximation transfer function has been made for $\rho = 0.5$. These two transfer functions have been multiplied with each other to obtian the transfer function used in this experiment.
Both cases have been performed for the wip-level in time, since the wip-level is the easiest performance indicator to obtain from the DEM. The expected wip-levels are the same for the different utilizations, since the server are switched. For steady state, wip-levels have to be $\overline{w} = 10$. The result of both cases is shown in Figure 4.5, in which



(a) $\rho_1 = 0.9$, $\mu_1 = 1/9$ *and* $\rho_2 = 0.5$, $\mu_2 = 1/5$ *for each workstation respectively*

(b) $\rho_1 = 0.5$, $\mu_1 = 1/5$ *and* $\rho_2 = 0.9$, $\mu_2 = 1/9$ *for each workstation respectively*

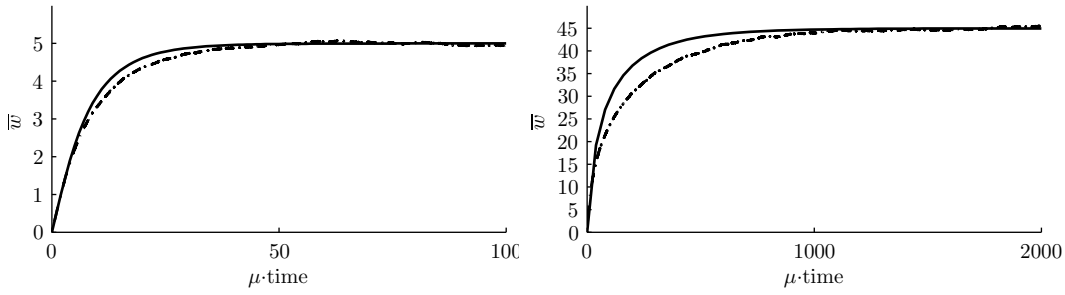
Figure 4.5: *The wip-level of the DEM (-·) and Padé approximation for the non-identical tandem queue*

the wip-level is stated against time. The accuracy of both graphs fulfill the steady state wip-level demand of $\overline{w} = 10$. Furthermore, the graphs seem to be as accurate as the one of the single server for $\rho = 0.9$ in Figure 4.3. Finally, both results are more or less similar, except for some variability differences.

After satisfactory results for two $M/M/1$ queues in series, the effect on the Padé ap-

proximation can be tested for a queue in series that is even stretched further. Therefore, an example has been introduced which considers five queues in series.

Example 4.5 (Five-in-line queue)**.**
In this example, five identical $M/M/1$ queues have been placed in series to demonstrate the maximum possible extension of the single server transfer function. The result of the transfer function model will be validated with a DEM. The simulations for the five-in-line queue has been performed for a process rate $\mu = 1.0$. Furthermore, two utilizations have been regarded $\rho = 0.5$ and $\rho = 0.9$, which results in arrival rates of $\lambda = 0.5$ and $\lambda = 0.9$.
The transfer function has been obtained from the single server Taylor series from the $M/M/1$ queue. The Taylor approximation of the $M/M/1$ queue has been taken to the fifth power, respecting the original order of the Taylor approximation. The powered Taylor approximation has been converted to a transfer function using Padé approximation. Finally, a sixth order transfer function has been obtained for this example.
Again, only the wip-level has been considered in this example, since the easiness to determine the wip-level from a DEM. The steady state outcome of the sixth order transfer function should be a wip-level of $\overline{w} = 5$ and $\overline{w} = 45$ for utilizations $\rho = 0.5$ and $\rho = 0.9$ respectively.
Figure 4.6 shows the validations for both experiments, where wip-levels are plotted in



(a) $\rho = 0.5$ *with 5 workstations* (b) $\rho = 0.9$ *with 5 workstations*

Figure 4.6: *The wip-level of the DEM (··) and the Padé approximation for the five-in-line queue*

time. The steady state wip-levels correspond for both experiments, but the transient behavior does not agree for the considered utilizations. The inaccuracy of the utilization $\rho = 0.5$ seems acceptable when the results of Figure 4.6(a) and Figure 4.4(a) are compared. The comparison of DEM and the transfer function shows a disappointing accuracy for an utilization $\rho = 0.9$. The utilization $\rho = 0.9$ seems the maximum transfer function extension in combination with a five-in-line queue and a sixth order Padé approximation.
In short, the validation results show that the transfer function extension can reach further for the utilization $\rho = 0.5$. However, the transfer function extension has reached a maximum for the utilization $\rho = 0.9$.

In all, during the study of multiple identical workstations in series, it appeared that

the influence of the order of the approximation reduces when transfer function has been powered. Furthermore, the sixth order transfer function model has reached a maximum when the $M/M/1$ model has been extended to a five-in-line queue with the utilization $\rho = 0.9$.

4.4 Résumé

In this chapter, an earlier derived solution (2.13) in the derivation of the time-dependent behavior has been used to describe an $M/M/1$ queue with a transfer function. This transfer function cannot be obtained immediately, since the transfer function has to be the quotient of two polynomials. The polynomial quotient has been computed by applying a Taylor and Padé approximation to develop a transfer function model of the $M/M/1$ queue.

The $M/M/1$ transfer function model has several advantages compared to the used time-dependent solution of Chapter 2. The first advantage involves the increased applicability, since the single server transfer function can describe multiple $M/M/1$ queues in series as well. A second advantage occurs when one wants to design a controller, since controller design is quite straightforward with transfer functions.

Results of the developed transfer functions have been discussed in Section 4.2 and Section 4.3. These sections examine the consequences of the approximation in the developed transfer function with validation models. Section 4.2 regards the single server queue validation results in which accuracy loss have been observed for queues with high utilizations. Furthermore, accuracy loss has been observed for the expansion to queues in series, see Section 4.3.

In brief, this chapter deals with the development of transfer functions for infinite queues. Therefore, finite queues will be considered in the next chapter, where similar modeling techniques of the last three chapters come forward again.

Chapter 5

Models for finite queues

Sofar, modeling techniques from queueing theory, DEMs and transfer functions have been used to describe infinite queues. This chapter switches to modeling finite buffers to show the contrast between infinite and finite buffers. Finite buffers will be considered, since the assumption of infinite buffers cannot be realistic in some manufacturing environments.
Furthermore, Section 5.1 shows that assuming finite buffers has the advantage of a finite description with Markov theory . Besides Markov theory, DEMs and transfer function models have been treated in Section 5.1. These models are further worked out in Section 5.2, where adjustments to the Markov model and DEM will be made. These adjustments will be used to match these models. The adjusted models have been validated in this section 5.2 as well. Finally, Section 5.3 considers two queues in series to show the transition from single server queues to queues in series. Again, the DEM and Markov model will be modified to result in similar models and these models will be validated as well.

5.1 $M/M/1/N$ queue

The Markov model of the $M/M/1/N$ queue can be obtained easily using the model of the $M/M/1$ queue. Instead of having an infinite number of states, the $M/M/1/N$ queue has a number of $N+1$ states. So, the process of the $M/M/1/N$ queue also corresponds

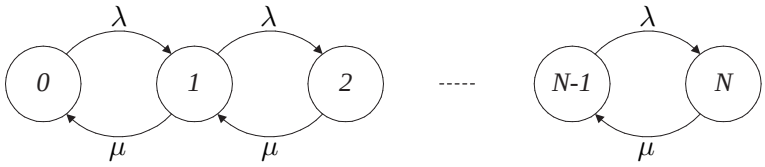


Figure 5.1: *The transition state diagram of an $M/M/1/N$ queue*

to a Markov birth-death process. In the $M/M/1/N$ queue, the birth and death rates are constant and the population has an N-size boundary. The transition state diagram of the N-state birth-death process is shown in Figure 5.1. In manufacturing, an N-state birth-death process corresponds with a server process, which can hold a single job and a buffer which contains $N-1$ waiting places. As done in Section 2.2, a time-dependent notation will be derived. In this notation, three ordinary differential equations are required, a single equation for the most left and for the most right state of Figure 5.1. Furthermore, $N-1$ identical equations can describe the states in between. Thus, an extra equation has to be added to the $M/M/1$ forward equations of (2.5) for state N. Moreover, (2.5a) is adapted and limited to state $N-1$. With these two modifications, the time-dependent expression of the $M/M/1/N$ queue results in:

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t) \tag{5.1a}$$

$$\frac{dP_k(t)}{dt} = -(\lambda+\mu)P_k + \lambda P_{k-1}(t) + \mu P_{k+1}(t) \quad \forall\ k = 1, 2, \ldots, N-1 \tag{5.1b}$$

$$\frac{dP_N(t)}{dt} = -\mu P_N(t) + \lambda P_{N-1}(t). \tag{5.1c}$$

Again, these ODEs form the fundamentals for deriving the limiting and time-dependent behavior of the $M/M/1/N$ process. The limiting and time-dependent behavior are shortly discussed.

Limiting behavior

Limiting behavior can be derived from the set of differential equations (5.1) by setting $\frac{dP(t)}{dt} = 0$. As for the $M/M/1$ case, certain conditions can be addressed which can be relevant before computing limiting behavior. Like the $M/M/1$ queue, the $M/M/1/N$ queue $\{X(t), t \geq 0\}$ is an irreducible continuous time Markov chain, see Definition 2.2. Note that $X(t)$ denotes the number of customers in the system at time t. The second condition applies to recurrence, for an $M/M/1/N$ queue all states are positive recurrent even for $\rho > 1$. When $\rho > 1$, (5.1) remains valid, because every state can be reached at all times.

The irreducibility and positive recurrence conditions lead to the ability of no constrictions in computing the limiting distribution. In the steady state, the probability distribution function satisfies the equation:

$$p_k = \frac{1-\rho}{1-\rho^{N+1}}\rho^k, \qquad \forall\ k = 0, 1, \ldots, N, \tag{5.2}$$

as can be found in e.g. Buzacott and Shanthikumar [Buz93]. The steady state performance measures are computed in a similar way as the $M/M/1$ case. Thus, the wip-level is the sum of all probabilities times the state-k,

$$\overline{w} = \sum_{k=0}^{\infty} k\, p_k, \tag{5.3}$$

and the throughput is the process rate times the non-idle probability,

$$\overline{\delta} = \mu\big(1 - p_0\big). \tag{5.4}$$

These equations also hold for time-dependent probabilities. Finally, the flow time can be calculated by using Little's Law.

Time-dependent behavior

The time-dependent probability distribution function is much harder to obtain. However, the time-dependent probability distribution function is still complex but not as complex in comparison to the time-dependent solution of the $M/M/1$ queue. Therefore, the derivation of the time-dependent solution is not treated here and can be found in [Tak62]. Takacs solution is presented as:

$$\begin{aligned}
P_k(t) &= p_k - C_0 \sum_{j=1}^{N} \frac{C_j}{\gamma_j} \, \mathrm{e}^{-\gamma_j \, \mu t} \\
p_k &= \frac{1-\rho}{1-\rho^{N+1}} \rho^k \\
C_0 &= -\frac{2\,\rho^{(k-i)/2}}{N+1} \\
C_j &= \left[\sin\left(\frac{i\,j\,\pi}{N+1}\right) - \sqrt{\rho}\sin\left(\frac{(i+1)j\,\pi}{N+1}\right)\right]\left[\sin\left(\frac{k\,j\,\pi}{N+1}\right) - \sqrt{\rho}\sin\left(\frac{(k+1)j\,\pi}{N+1}\right)\right] \\
\gamma_j &= \rho + 1 - 2\sqrt{\rho}\cos\left(\frac{j\pi}{(N+1)}\right).
\end{aligned}$$

Other interesting solutions of the time-dependent probability distribution function are treated in Sharma and Tarabia [Sha00], where easier to compute formulas are presented. Kulkarni [Kul95] also presents some methods to compute the time-dependent behavior. One of them puts (5.1) in matrix notation, $\dot{P}_n(t) = P_n(t)Q$ and takes its Laplace transform. Then, together with an initial condition of $P(0) = I$, the following approach results in:

$$P^*(s) = [sI - Q]^{-1}. \tag{5.5}$$

In this equation, $P^*(s)$ results in an $(N+1) \times (N+1)$ matrix with all time-dependent probabilities $(N+1)$ from every initial state $(N+1)$. For an initially empty system, the row $P^*_{1j}(s)$ holds all probabilities for single states, $j = 0{+}1, \ldots, N{+}1$. At this point, the individual equations of $P^*_{1j}(s)$ can be seen as probability transfer functions. Of course, transfer functions which can be used easily for applying control theory with standard ODE solvers. Now, a transfer function of performance measures for the wip-level and the throughput can be derived with the obtained probability transfer functions. In Example 5.1, transfer functions of the throughput will be worked out further.

Unfortunately, some issues remain. The problem for the derivation of performance measures is to define the flow time. For infinite buffer queues, the flow can be expressed

using the poisson input of λt. However, finite queues do not have a constant input rate λ anymore due to blocking. The flow time can be obtained numerically in spite of the lack of a linear input relation. Nevertheless, an explicit expression cannot be formed for the flow time.

Another problem of (5.5) is the calculation of the inverse. Computing an inverse of a matrix is a time consuming process. Fortunately, matrix Q contains a lot of zeros and can be formed sparse, which results in less computation time. For example, the matrix to be inverted for an $M/M/1/100$ is occupied with $\frac{2\cdot 2+3\cdot 98}{10.000} = 3\%$ of the places. Although, the lack of an expression for the flow time and the presence of an inverse, solving (5.5) is quite straightforward when the Q matrix is available. The following example has been used to illustrate a relatively simple method to compute a transfer function for an $M/M/1/2$ queue and its performance measures.

Example 5.1 (Derivation of a transfer function for an $M/M/1/2$ queue)**.**
Here, a derivation has been performed for finite queue to obtain a transfer function. Furthermore, the result of the derivation has been illustrated with a graph of the throughput. The considered finite queue contains a single buffer and one machine place, thus an $M/M/1/2$ queue is of concern.
The previously described forward equations (5.1) will be used to compute a transition state matrix, Q. Accordingly, the following relation holds for the forward equations: $\partial P(t)/\partial t = P(t)Q$. Herein, the transition state matrix for the $M/M/1/2$ queue is stated as:

$$Q = \begin{bmatrix} -\lambda & \lambda & 0 \\ \mu & -(\lambda+\mu) & \lambda \\ 0 & \mu & -\mu \end{bmatrix}$$

and with probabilities,

$$P(t) = [P_0(t)\ P_1(t)\ P_2(t)]\,.$$

Accordingly, $P^*(s)$ can be computed with (5.5):

$$P^*(s) = \begin{bmatrix} s+\lambda & -\lambda & 0 \\ -\mu & s+\lambda+\mu & -\lambda \\ 0 & -\mu & s+\mu \end{bmatrix}^{-1} = \frac{1}{s\left((\lambda+\mu+s)^2-\lambda\mu\right)}\times$$

$$\begin{bmatrix} s^2+2\,s\mu+\lambda\,s+\mu^2 & \lambda\,(s+\mu) & \lambda^2 \\ \mu\,(s+\mu) & (s+\lambda)\,(s+\mu) & (s+\lambda)\,\lambda \\ \mu^2 & (s+\lambda)\,\mu & s^2+2\,\lambda\,s+s\mu+\lambda^2 \end{bmatrix}.$$

In $P^*(s)$, transfer functions are stated for all possible probabilities with all initial possibilities on $t=0$. In row $P^*_{1j}(s)$, the probabilities are given for an initial empty system. So, the wip-level in the system correspond with zero, $w=0$. Transfer functions of the throughput and wip-level can be derived using (5.3) and (5.4). The transfer function for the throughput only needs the zero probability transfer function $P^*_0(s)$, accordingly the equation for $P^*_0(s)$ results in:

$$P^*_{(1,1)}(s) = P^*_0(s) = \frac{s^2+2\,s\mu+\lambda\,s+\mu^2}{s\,(s^2+2\,s\mu+2\,\lambda\,s+\mu^2+\lambda\,\mu+\lambda^2)}. \tag{5.6}$$

With $P_0^*(s)$, the transfer function for the throughput has been computed and after some simplification it becomes:

$$H_\delta(s) = \frac{1}{\rho}\big(1 - sP_0^*(s)\big) = \frac{(s+\lambda+\mu)\,\mu}{s^2 + (2\,\mu + 2\,\lambda)\,s + \mu\,\lambda + \lambda^2 + \mu^2}. \tag{5.7}$$

The throughputs transfer function holds the exact time-dependent solution of the $M/M/1/N$ queue, while in Chapter 4 an approximation has been made for the $M/M/1$ queue. In the throughput's transfer function only two poles appear, while matrix Q is a 3×3 matrix. One pole disappears due to the sum of probabilities, which is one of course. Furthermore, limits for $t \downarrow 0$ and $t \to \infty$ can be computed with (5.7). The general result for $t \to \infty$ corresponds with the limiting or steady state throughput:

$$\delta = \mu\left(1 - \frac{1-\rho}{1-\rho^{N+1}}\right)$$

and for $t \downarrow 0$ the throughput comes down to, $\delta = 0$.
Now, (5.7) can be used to generate the transfer function which can be visualized in a plot. For this occasion, an arrival rate $\lambda = 0.5$ and process rate $\mu = 1.0$ have been chosen, resulting in a utilization $\rho = 0.5$. So, the expected throughput evolves to $\delta = 1 \cdot \left(1 - \frac{1-0.5}{1-0.5^3}\right) \approx 0.43$ instead of a steady state $\delta = \lambda$ for the infinite queue. In Figure 5.2, the time-dependent result of the throughput has been given for a step λ

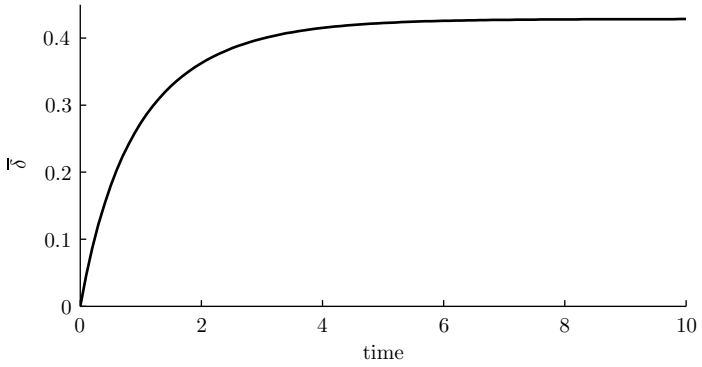


Figure 5.2: *Throughput response for* $\mu = 1.0$ *and* $\rho = 0.5$

using Matlab. The asymptotic value of the graph corresponds with the expected steady state throughput $\delta \approx 0.43$.
Shortly, a straightforward technique has been used to determine a probability transfer function of a finite queue. Besides, the transfer function has been adjusted to describe performance measures in time on a step response.

In a similar way, the wip-level can be computed. For the wip-level one has to sum up all probabilities times its corresponding wip-level. Note that, the wip-level can also be computed by using integrals of the influx and outflux (throughput) of the system. Then, not all probabilities are demanded for computing the wip-level. The computation of the

flow time can be performed with computing the integral of the throughput and the wip-level minus the integral of the throughput. Then, with the use of an interpolation algorithm the flow time can be numerically derived.

5.2 Single server queue

In this section, the DEM performance will be compared with the Markov theory stated in the previous section. At first, the $GBME$ model in Section 3.2 has been inspected. The $GBME$ χ code can be found in Appendix B. In the $GBME$ model the buffer is infinite. Consequently, the $GBME$ queue has to be provided with a finite buffer which results in the $GB_{N-1}ME$ queue. Therefore, the buffer process of the $GBME$ model has been adapted to a finite one with a change the repetitive selective waiting statement, see (χ-5.1).

$$
\begin{array}{l}
*[\ \mathrm{len}(xs) < N-1; a?x \qquad \longrightarrow xs := xs +\!\!+ [x] \\
\ [\!]\ \mathrm{len}(xs) > 0; \qquad b!\,\mathrm{hd}(xs) \longrightarrow xs := \mathrm{tl}(xs) \\
\ ]
\end{array}
\qquad (\chi\text{-}5.1)
$$

The finite buffer contains $N-1$ places. Together with a single place in the machine M, the workstation contains N places. Therefore, the DEM will be referred to a $GB_{N-1}ME$ or GW_NE queue. Herein, W_N is a workstation consisting of N places.

The outcome of the GW_NE model can be compared with the behavior of the $M/M/1/N$ queue. Then, both models are not in agreement with each other. The discrepancy between the $GB_{N-1}ME$ and the $M/M/1/N$ model occurs due to different handling situations with full buffers. In the $M/M/1/N$ model, arriving jobs disappear when the buffer is full. In other words, the queueing system is not fed with arrivals anymore. Whereas in the DEM, the generator still creates new jobs with rate λ, even when the buffer is full. The generator stops producing lots when it cannot send lot to the buffer anymore. So, the generator does not stop immediately when the buffer is full, but the generator blocks when it tries to send a lot to a full buffer. Since, the $M/M/1/N$ and DEM do not correspond to each other, two possibilities remain to let them agree. First, the DEM can be adjusted and second the Markov model can be changed.

Adjustments DEM

In the first case, the generator of the DEM must be switched off when the buffer is full. Therefore, the generator of the χ model has to be equipped with a communication signal from the buffer. A non-full buffer should authorize the generator to produce lots by a communication signal. This extra communication changes both processes and the

codes of these processes can be seen in (χ-5.2) and (χ-5.3).

```
proc G(a: !lot, z: ?void, ρ, μ: real) =
[[ i : nat, u : →real
| i := 1; u := negexp(1/ρ/μ)
;*[ true⟶ z?; Δσu; a!⟨i, τ⟩; i := i + 1
  ]
]]
```
(χ-5.2)

```
proc B(a: !lot, a: ?lot, z: ?void, N: nat) =
[[ xs : lot*, x : lot, sent : bool
| xs := []; sent := false
;*[ len(xs) < N − 1; z!          ⟶ sent := true
  [[ sent;              a?x      ⟶ xs := xs ++ [x]; sent := false
  [[ len(xs) > 0;       b! hd(xs)⟶ xs := tl(xs)
  ]
]]
```
(χ-5.3)

In (χ-5.2), the generator receives an authorization of a non-full buffer to create lots. These created lots are sent after some exponential sampled time to the non-full buffer. The buffer can send lots to the next process for a non-full and full situation. With a full situation, the buffer cannot send the authorization signal to the generator leaving the generator switched off until an established communication. The appending of a communication signal results in another structure of the standard $GB_{N-1}ME$ into the modified $GB_{N-1}ME$, see Figure 5.5. Example 5.2 makes a validation for the modified

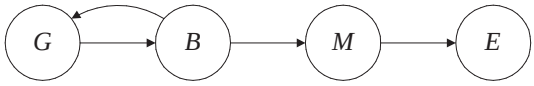


Figure 5.3: *Structure of the modified DEM*

$GB_{N-1}ME$ queue and the complete χ code of this modified queue can be found in Appendix B.2.

Adjustments $M/M/1/N$ model

Instead of changing the DEM, one can also obtain a Markov model which describes the standard $GB_{N-1}ME$ model. When the Markov chain of Figure 5.1 is considered, the construction of the Markov model is adjusted by appending a new state $N+1$, see Figure 5.4. Then, in state N the generator still releases new lots with rate λ and when a lot has been created the system blocks in state $N+1$. The only remaining situation to leave state $N+1$ is via rate μ or a service completion.

The adjustment of the $M/M/1/N$ into the $M/M/1/(N+1)$ model has consequences for the performance measures. The throughput will agree with the one of an $M/M/1/(N+$

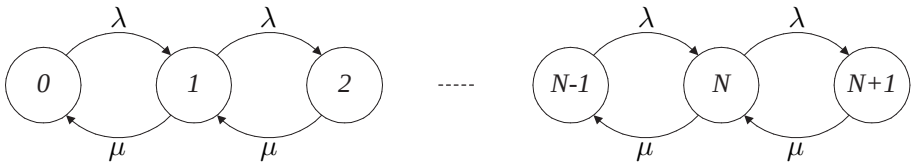


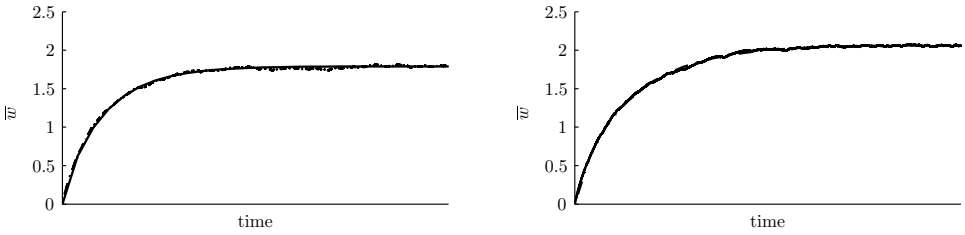
Figure 5.4: *The transition state diagram of an $M/M/1/(N+1)$ queue*

1) Markov model. However, a more serious change is required for the wip-level, since the system holds only N jobs in state $N+1$. Thus, the expression of the wip-level becomes:

$$\overline{w}(t) = \sum_{k=0}^{N} k p_k(t) + N P_{N+1}(t). \tag{5.8}$$

In the next example, both adjustment methods are validated for the wip-levels.

Example 5.2 (Single server validation)**.**
In this example, a validation has been made for a queue with four workstation places, thus $N = 4$. The validation has been performed for the wip-level with an utilization of $\rho = 0.9$ and a process rate of $\mu = 1.0$.
Two validations have been executed: one for a standard $M/M/1/N$ model and one for a modified $M/M/1/(N+1)$ model. The validation models are DEMs: one is a modified GW_NE queue and the other is a standard GW_NE model. Since wip-levels are easy to obtain of the DEM, only the wip-level will be considered in this example. Furthermore, the derived Markov models should be exact and, accordingly, an extensive validation is not required when the wip-levels correspond.
With (5.2), (5.3) and (5.8), the steady state values of the experiment can be computed. The steady states values of the validation should respect wip-levels, $\overline{w} \approx 1.79$ and $\overline{w} \approx 2.07$ for the standard $M/M/1/N$ and modified $M/M/1/(N+1)$ as can be computed with (5.3) and (5.8) respectively. In Figure 5.5, the results of the wip-level



(a) *Validation of modified GBME model* (b) *Validation of the $M/M/1/(N+1)$ model*

Figure 5.5: *Validation of the adjusted models*

validation have been plotted for the adjustment of the DEM and for the adjustment of the $M/M/1/N$ Markov model. Herein, wip-level has been stated against time. Moreover, the validation of the models has been successful, because the wip-level are equal except for some variability in the DEM. Furthermore, the steady state values match

their expected value.
Besides, a substantial contrast can be distinct between both figures 5.5(a) and 5.5(b) especially in steady state, because of different blocking policies. The $M/M/1/N$ queue has a wip-level of 86% of the standard $GB_{N-1}ME$. Consequently, blocking policies can differ a lot in wip-levels and other performance measures.
In short, a successful validation has been made for the earlier composed models in which the standard $M/M/1/N$ and GW_NE models have been adjusted.

5.3 Tandem queue

Usually, manufacturing systems consist of more than a single buffer and machine. Consequently, extending the single server system is of great interest. Unfortunately, transfer functions of the $M/M/1/N$ queue cannot be powered as in the case of an $M/M/1$ queue. The ability of powering transfer functions falls away when finite buffers are introduced. Mainly, because of the influence on the transfer function of the first workstation by processes later in the queue. So, process times of workstation not only depend on behavior in front of the line, but also on behavior behind the workstation in question of the system. Therefore, a 2D transition state diagram has been made, representing two $M/M/1/N$ queues in series. In Figure 5.6, the visualization of the tandem $M/M/1/N$

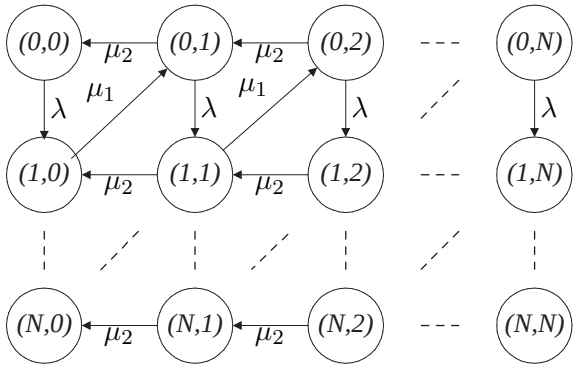


Figure 5.6: *The transition state diagram of an $M/M/1/N$ queue*

queue is shown. Note that, a transition state diagram does not have to be 2D, an 1D composition is also possible for queues in series. The situation of two $M/M/1/N$ queues in series become more clear in a 2D illustration. As for the single $M/M/1/N$ queue, the forward equations can be easily derived using Figure 5.6:

$$\frac{dP_{00}(t)}{dt} = -\lambda P_{00}(t) + \mu_2 P_{01}(t)$$
$$\frac{dP_{m0}(t)}{dt} = -(\lambda + \mu_1)P_{m0}(t) + \lambda P_{m-1,0}(t) + \mu_2 P_{m1}(t) \quad \forall\ m = 1, 2, \ldots, N-1$$
$$\frac{dP_{0n}(t)}{dt} = -(\lambda + \mu_2)P_{0n}(t) + \mu_1 P_{1,n-1}(t) + \mu_2 P_{0,n+1}(t) \quad \forall\ n = 1, 2, \ldots, N-1$$

$$\frac{dP_{mn}(t)}{dt} = -(\lambda+\mu_1+\mu_2)P_{mn}(t) + \lambda P_{m-1,n}(t) + \mu_1 P_{m+1,n-1}(t) + \mu_2 P_{m,n+1}(t)$$
$$\forall\ m,n = 1,2,\ldots,N-1$$
$$\frac{dP_{N0}(t)}{dt} = -\mu_1 P_{N0}(t) + \lambda P_{N-1,0}(t) + \mu_2 P_{N1}(t)$$
$$\frac{dP_{Nn}(t)}{dt} = -(\mu_1+\mu_2)P_{Nn}(t) + \lambda P_{N-1,n}(t) + \mu_2 P_{N,n+1}(t) \quad \forall\ n = 1,2,\ldots,N-1$$
$$\frac{dP_{0N}(t)}{dt} = -(\lambda+\mu_2)P_{0N}(t) + \mu_1 P_{1,N-1}(t)$$
$$\frac{dP_{mN}(t)}{dt} = -(\lambda+\mu_2)P_{mN}(t) + \lambda P_{m-1,N}(t) + \mu_1 P_{m+1,N-1}(t) \quad \forall\ m = 1,2,\ldots,N-1$$
$$\frac{dP_{NN}(t)}{dt} = -\mu_2 P_{NN}(t) + \lambda P_{N,N-1}(t).$$

Remarkable of the change to 2D is that the number of equations are squared for the two server queue in comparison with the single server. Logically, when more queues in series are considered the number of equations expand rapidly.

Similar to the finite single server queue in Section 5.2, the earlier treated tandem DEM ($GBMBME$) of Chapter 3 does not correspond with the tandem $M/M/1/N$ queue by only adding a finite amount of buffer places. Moreover, adding a workstation to the DEM of Figure 5.3 is not sufficient enough to describe this Markov model. So, to let these models match, one of them has to be adjusted.

Adjustments two workstation DEM

In the first case, the DEM has been modified to satisfy the tandem $M/M/1/N$ behavior. The tandem $M/M/1/N$ behavior cannot be described with the modified GW_NE queue by appending an extra workstation. An extra placed workstation results in a GW_NW_NE queue. This queue describes the tandem $M/M/1/N$ when a non-full buffer allows first machine to produce lots. Therefore, the lot production should be authorized by the buffer with an extra communication signal of the buffer. This communication signal results in another DEM structure that has been visualized in Figure 5.7. The introduced

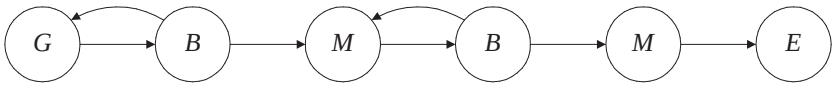


Figure 5.7: *Structure of the DEM of two $M/M/1/N$ queues in series*

communication between the first machine and the second buffer requires a change in χ codes. The χ has been discussed already and is shown in (χ-5.3). The χ adjustment of the first machine needs a straightforward replacement of the repetition, which is shown in (χ-5.4).

$$;*[\,\mathsf{true} \longrightarrow a?x; z?; \Delta\sigma u; b!x \;]\qquad (\chi\text{-}5.4)$$

In this χ code, the machine requires an authorization signal from the buffer before lot production starts. The complete χ code can be found in Appendix B.2. The complete modified GW_NW_NE queue has been validated with a Markov model of two $M/M/1/N$ queues in series in Example 5.4.

Adjustments Markov model of two $M/M/1/N$ queues in series

Of course, the other option is to adjust the Markov $M/M/1/N$ model. In this case, the different blocking conditions in the DEM have to be described by the Markov model. These blocking situations can occur at the generator and the first machine. The generator blocks when a new lot has been made with a saturated first buffer. The first machine blocks when the processing of a lot has finished and the second buffer is full. In Figure 5.8, a transition state diagram can be seen for the situation of $N = 2$ or a GW_2W_2E queue with a single buffer place and one machine position.

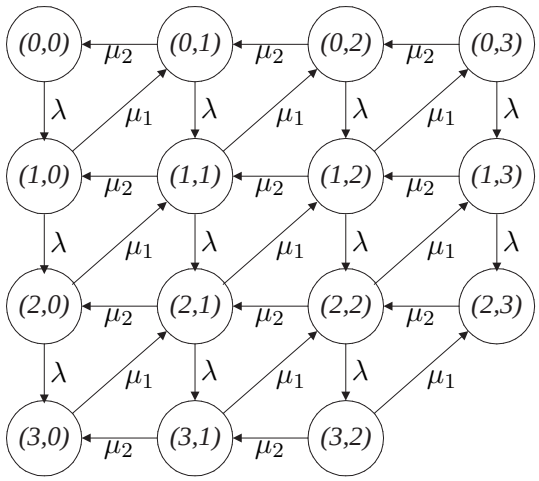


Figure 5.8: *State transition diagram of the GW_2W_2E model*

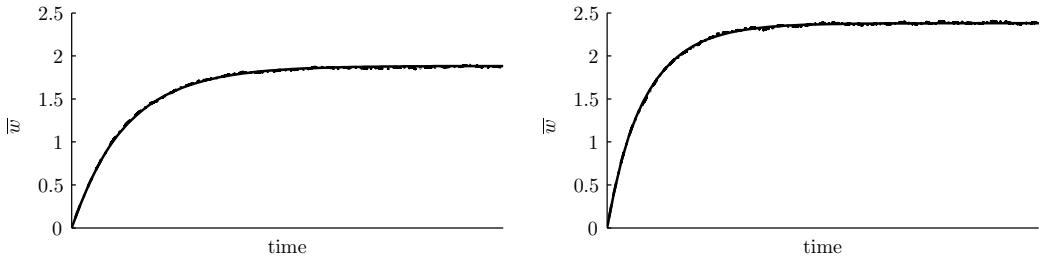
Below, an example has been performed to clarify Figure 5.8. Herein, an illustration has been made with a walk through the diagram. Moreover, a comparison has been using to find the Markov state with the corresponding DEM situation.

Example 5.3 (Diagram walk)**.**
This example should clarify Figure 5.8, because an example walk will be made through the diagram.
The start of the walk has been chosen to be state $(2, 2)$ of Figure 5.8. So, the system finds itself in state $(2, 2)$. In state $(2, 2)$, every buffer and machine are filled with one lot, so the maximum wip-level of four has been reached.
In the $(2, 2)$ position, the DEM's generator still creates lots with rate λ and the machines produces lots with rates μ_1 and μ_2.

If the situation happens that a new lot has been generated via rate λ, the system blocks at the generator in state $(3, 2)$. Or, if the situation happens that the first machine finishes processing, the system blocks at machine one in state $(1, 3)$.
From both states $(1, 3)$ and $(3, 2)$, the possibility of reaching both blocking situations is present by a single 'jump'. Then, the double blocking situation occurs with arrival rate λ or with processing rate μ_1 respectively.

Obviously, one can make the transitions for a lot of situations. For all these possibilities equations can be expressed. Like the previously discussed Markov process, a set of ODEs can be formed. In Appendix E, the derivation of the ODEs has been treated extensively. To complete the adjustment of the tandem $M/M/1/N$ queueing model, a validation has been made for the wip-level in Example 5.4.

Example 5.4 (Validation of tandem queues)**.**
This example treats a validation of the model for two workstations in series, where each workstation contains two places. The validation has been performed for the wip-level with an utilization of $\rho = 0.9$ and a process rates of $\mu = \mu_1 = \mu_2 = 1.0$.
The modified GW_2W_2E will be tested with the Markov model of two $M/M/1/2$ queues in series. Moreover, the modified Markov model describing the standard GW_2W_2E has been validated with the χ simulation data of the DEM. Therefore, only the wip-levels have been considered, because these are easy to obtain from the DEM. From the Markov theory, steady state values of the wip-level can be derived. The tandem $M/M/1/2$ queue has a wip-level $\overline{w} = 1.88$ and the wip-level of Markov model of the GW_2W_2E queue becomes, $\overline{w} = 2.38$.
In Figure 5.9, the validations are shown of both experiments for the wip-level in time.



(a) *Validation of the tandem $M/M/1/2$ model* (b) *Validation of modified GW_2W_2E model*

Figure 5.9: *Validation results*

Both situations, the DEM and Markov model correspond well to each other. Furthermore, it appears that the difference between both steady state wip-levels remain evidently visible due to other blocking policies. The $M/M/1/2$ queue has a wip-level of only 79% of the GW_2W_2E queue. Consequently, blocking policies can make a large difference, while the buffer places are the same. Logically, the wip-level difference has consequences for other performance measures. However, the difference in wip-levels decrease with larger buffer capacities or with lower utilizations.

5.4 Résumé

The advantage of using finite queueing transfer functions is the exact solution in comparison with transfer functions of infinite queueing models. Infinite queueing transfer functions have a reduced accuracy for high utilizations, while the accuracy of finite queueing transfer functions are not restricted by the utilization. Actually, the accuracy forms the main reason to prefer a transfer function of a finite queue, since some disadvantages are present as well.

The transfer functions' computation time can become very large for large finite queues, since it requires the inverse of the transition state matrix, see (5.5). A second disadvantage occurs when one wants to use a single server transfer function to describe queues in series. The extension to queues in series is not possible with finite buffers, since combining transfer functions cannot be used for extension of queues. This lack of extension ability results in new computation efforts for a Markov model. A third disadvantage is the missing of a relation of the flow time. The flow time cannot be obtained with a transfer function relation from the Markov theory, but a numerical derivation of the flow time is possible.

Probably, the influence of these disadvantages can be reduced with some extra research. Extra research can been demanded for studying the possibility of using network queueing theory relations to avoid extra computation efforts when single server had to be extended. Furthermore, these extra computation efforts may be avoided by studying differences in the transfer function between a single and a tandem system. This study seems relevant, since the transfer function of the first workstation changes due to an added second workstation.

Another interesting part of this chapter occurs during the validation the transfer functions. Observations of the derived transfer showed that the standard $M/M/1/N$ model and $GBME$ model do not correspond. The discrepancy between these queues is a result of different blocking policies. Consequently, the blocking policies in both models have been changed for each model such that the standard $M/M/1/N$ agrees with a adjusted $GBME$ model and vice versa. These model differences are essential to know before modeling a real-life manufacturing system.

In all, the derivation of the transfer function itself is quite straightforward, while problems can occur due to an inverse of large system matrices. The critical size of large system matrices has to be investigated to reveal the boundaries of the used Markov theory. Finally, the Markov models of this chapter will used in Chapter 6 to develop of a controller for a finite queue. Besides, the developed controller will implemented on a DEM.

Chapter 6

Control

In this thesis, a lot of references have been made to 'control' and its requirements. Sofar, control has not been treated, since the focus has been on modeling aspects of the presented control framework in Chapter 1. Finally, control aspects of the control framework are discussed in this chapter.
The control aspects part starts with the design of a control-law in Section 6.1. This control-law uses a state-feedback method, which estimates all system states with the earlier developed models. The estimated system states supply the system with a new input signal. These estimations should be based on the output signal of the χ model. Consequently, the χ model and the controller have to be integrated, which is treated in the implementation of Section 6.2. Finally, the implementation considers two queues in series for which single run simulation results and average performance measures have been shown.

6.1 Control design

The state space design method has been chosen to develop a control law for a manufacturing system. State space manipulates the earlier derived transfer functions into the following state-variable form [Fra94]:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx. \end{aligned}$$

A state space notation is easy to obtain from the transfer function with Matlab's *tf2ss* function. The advantages of state space design are especially for systems with more than one control input or more than one sensed output. In this thesis, only the applicability of a control implementation will be shown and therefore the state space design has been used for the simpler single input single output systems.

In control engineering, stability is one of the main issues of dynamic systems and for applying control. For the considered manufacturing systems, stability depends on the mean return time to a certain state or wip-level. If the mean return time to all is states is finite then the manufacturing system meets stability. In Definition 2.3, stability has been described in the positive recurrent condition. Moreover, the system is stable, since a transfer function can be derived. If the poles of (5.7) in Example 5.2 are in the left half plane, then a stable response occurs on an impulse.

In this chapter, control will be applied only to the finite buffer queues, resulting in a stable system to be controlled no matter the utilization.

Design of full state state feedback control

Instead of stability, controllability is required for the design of a control law. Consequently, one has to check whether or not the system is controllable. For the controllability, a controllability matrix has been defined as:

$$\mathfrak{C} = [B \; AB \; \cdots \; A^{n-1}B],$$

where A is $N \times N$ matrix [Fra94]. When the controllability matrix $\mathfrak{C}$ is nonsingular the corresponding A and B matrices are said to be controllable for single input single output systems. Controllability is a function of the state of the system and cannot be decided from a transfer function. However, a system is always controllable with an available transfer function. For the controllability, a minimal state-space realization has to be made of the transfer function. A minimal realization is automatically composed with Matlab's *tf2ss* routine.

Now, a control law can be developed. The purpose of a control law,

$$u = -Kx,$$

is to derive a set of pole locations for feedback that will agree with the desired behavior. The control gain K can be computed in several ways. The control gains most important task is to set the pole locations in the left half plane for stability reasons. The control matrix K can be computed via various algorithms [Fra94], such as Ackermann's formula with given pole locations . Yet, the selection of pole locations remain and this selection of poles have to be examined and determined. Accordingly, another most effective technique, which can be used of linear control law design, is the optimal linear quadratic regulator (LQR) [Fra94]. Since the primary goal is to obtain a controller which improves the performance of the system, the LQR is not discussed any further. In this study, the LQR design method has been used to solve the optimal control law and K has been obtained with the *lqr* function in Matlab.

Observer design

The design of the control law applies to all state variables of the system, since a full state feedback has been chosen for the system. The determination of all state-variables is not possible in most systems. Consequently, the next step is to design an observer or estimator which estimates all states of the system with the aid of a single measured value such as the wip-level. The full state estimate is referred to $\hat{x}$ and with the following equations the model of the observer has been expressed as:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + L(y - \hat{y}) \\ \hat{y} &= C\hat{x}. \end{aligned}$$

The observer is a full-order estimator of the desired DEM model to control. Here, instead of controllability, one has to check observability. Likewise, the observability Matrix $\mathfrak{O}$ has to be nonsingular or have a non-zero determinant. The observability matrix is defined as:

$$\mathfrak{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

Now, the only required element for the observer is the observer gain L. The observer gain L can be computed using the control gain. The selection of poles can be derived from the control law. In general, the estimator poles have to be faster than controller poles, resulting in faster reaction of the observer. As a rule of thumb, the estimator poles can be chosen to be a factor two to six greater than the controller poles [Fra94]. With the use of the observer, the control feedback becomes:

$$u = -K\hat{x}.$$

To make the control system complete, the steady state has to be introduced. In steady state, reference signal u_{ss} supplies the system of the desired input rate. Besides, the full state steady state value x_{ss} provides the system with steady state correction for the observer. When the steady state has been included, the final control law becomes:

$$u = u_{ss} - K(\hat{x} - x_{ss}).$$

With this final step, the whole control system can be schematically drawn. In Figure 6.1, the control diagram can be seen. For this occasion, the observer in the figure contains matrices, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ which can be computed with the Markov model. The matrices of the plant model are, A, B, C. These matrices cannot be obtained and can be considered as the χ model. Therefore, the rest of this report considers, A, B, C as state space matrices of the observer.

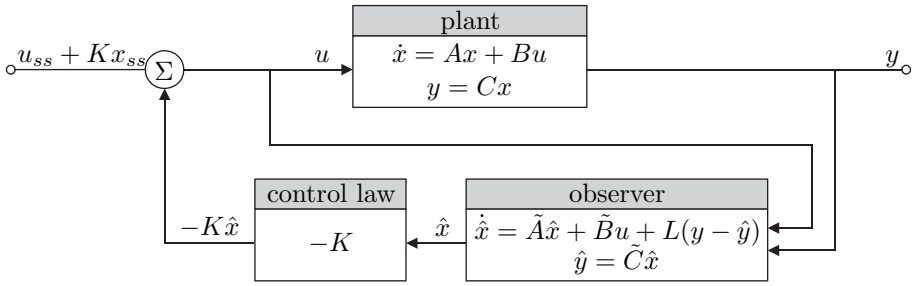


Figure 6.1: *Schematic diagram of the control strategy*

6.2 Implementation

The GW_2W_2E queue has been used for the implementation of a control law. The values used for the utilization and process rates of the GW_2W_2E queue are the same as in Example 5.4. The wip-level has been considered as output signal y of the controlled system. In Figure 5.9, the uncontrolled wip-level is shown with respect to time. With the control law, the target is to increase the wip-level earlier, so that the same steady state is reached sooner.

Simulink testing

First, a test has been made in Simulink. From the derived transfer function for a GW_2W_2E queue with $\rho = 0.9$ and $\mu_1 = \mu_2 = 1.0$, a state space realization has been made. In the state space, A is a (14×14) matrix, B is a (14×1) matrix and C is an (1×14) matrix as stated in Appendix F.1.

With the obtained state space realization, the control gain can be computed with the linear quadratic regulator technique. The observability gain L can be computed with the help of the rule of thumb, where the observer is chosen two times faster than the control gain. With these conditions, the model has been put in Simulink. Then, the control implementation can be made very clearly and understandably using Simulink. In Figure 6.2, the global Simulink testing model is shown and the total Simulink model can be found in Appendix F.1. Before the controller can be implemented a test has been performed. In the test, Simulink uses the Markov model as the plant and as the observer. Consequently, the choice of the observer gain L is not important in this situation, since $y - \hat{y}$ is zero. Logically, an observer is not demanded in this case, since y and $\hat{y}$ are identical scalars. The results of the Simulink test model can be seen in Figure 6.3. For the implementation the output y in the Simulink model corresponds with the wip-level and the input u with arrival rate λ. The graph of the Simulink test shows that steady state is reached sooner and that the demands have been fulfilled. Now, the 'plant' in the Simulink model has to be substituted by the χ model. This is simply performed by replacing the Markov model with an input and output signal to

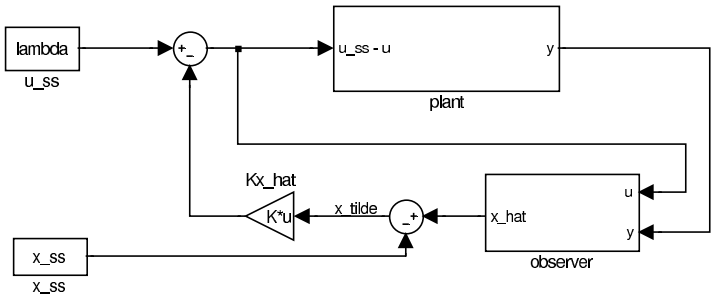


Figure 6.2: *Simulink model*

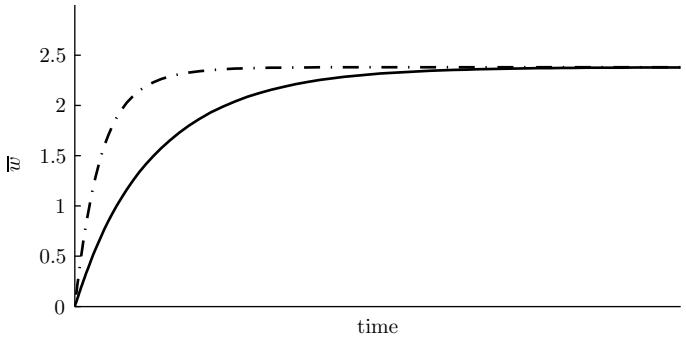


Figure 6.3: *Uncontrolled (–) versus controlled (·−) model*

the Matlab workspace. Python is perfectly suitable as an interface between Matlab's workspace and χ. In the next section, the χ integration will be discussed further and Appendix F.1 presents the whole Simulink model.

χ integration

Two important changes have to be made before the controlled χ model can be used. First, the χ file has to be adapted to the ability of control. Second, an interface has to be made which couples the Simulink model and the χ file.

1. χ adjustments.
The χ file has to be adjusted to the possibility of applying control. The generator will be modified for controlling the DEM in the χ model, since the generator releases lots with rate λ and Simulink supplies the generator of the input rate. Accordingly, the generator has been adjusted to receive a new λ as arrival rate. The generator consists of three tasks:

1. track the wip-level of the whole system;
2. communicate with the controller for the control signal;
3. send lots.

In order to receive a new λ, the generator has to keep up with the current wip-level (1). Simulink has to know the current wip-level to supply the generator with the control signal. Consequently, the generator has to communicate with Simulink (2). Finally, the generator has to be able to send lots (3). These three generator changes of the χ model are treated below.

First to be able to track the wip-level, a communication with the exit process has to be created in the generator process. Therefore, the exit process sends a void when it receives a lot. So, the generator can keep with the input lots and thus, can obtain the current wip-level. The communication between the generator and the exit process has to be able to happen at all time. Consequently, this communication comes with a boolean which is *true* in the first statement of the repetitive selective waiting of (χ-6.1).

Second, to provide the generator with the desired input rate, a sample frequency dt has been introduced. The update of the controller is performed at sample times t_σ, where the controller receives the current wip-level. In return, the generator receives the new input rate from the controller. The controlled input rate has to be exponentially distributed, so the new arrival time becomes a sample of the exponential distribution of one divided by the controllers output λ. The χ code of keeping the controller up-to-date can be found in the second option of the repetitive selective waiting of (χ-6.1).

Third, the issue of the release time to send lots remains. Therefore, the send time t_{send} has been introduced. The send time computes the absolute time when a lot has to be released. Consequently, the previous release time has to be included in t_{send}. Of course, the send time can be elapsed when the received λ has been increased by the controller. Accordingly, the send time becomes the maximum of the just computed send time and the absolute time:

$$t_{send} := \max(\tau, t_{prev} + \frac{t_{exp}}{\lambda}).$$

When the time reaches t_{send}, the generator tries to send a lot. Depending on the buffer quantity, the lots will immediately be sent or the generator switches to *trysend*. The *trysend* boolean is required to be able to sample at the sample frequency and to be able to receive the synchronization from the exit process continuously. In (χ-6.1), the new generator process has been stated. In Appendix F.2, the χ code of the whole model is shown.

$$
\begin{array}{l}
\textsf{proc}\ G(a\colon !\text{lot}, z\colon ?\text{void}, \rho, \mu\colon \text{real}) = \\
[\![\, dum, trysend : \text{bool}, i, wip : \text{nat}, u : \rightarrow\text{real}, \lambda, dt, t_\sigma, t_{exp}, t_{prev}, t_{send} : \text{real} \\
\mid i := 0; wip := 0; u := negexp(1.0); dt := 0.01; t_\sigma := \tau; t_{exp} := \sigma u; t_{prev} := \tau \\
; dum := \texttt{openmatlab}() \\
; \lambda := \texttt{runmodel}(\texttt{n2r}(wip)); t_{send} := \max(\tau, t_{prev} + \tfrac{t_{exp}}{\lambda}) \\
; *[\ \ \textsf{true}; \quad b? \qquad\quad \longrightarrow wip := wip - 1 \\
\quad [\!]\ \ \textsf{true}; \quad \Delta t_\sigma - \tau \quad \longrightarrow \lambda := \texttt{runmodel}(\texttt{n2r}(wip)) \\
\qquad\qquad\qquad\qquad\quad\ \ ; t_{send} := \max(\tau, t_{prev} + \tfrac{t_{exp}}{\lambda}) \\
\qquad\qquad\qquad\qquad\quad\ \ ; t_\sigma := dt + \tau \\
\quad [\!]\ \neg trysend; \Delta t_{send} - \tau \longrightarrow trysend := \textsf{true} \\
\quad [\!]\ \ trysend; a!\langle i, \tau\rangle \quad\ \longrightarrow trysend := \textsf{false} \\
\qquad\qquad\qquad\qquad\quad\ \ ; i := i + 1; wip := wip + 1 \\
\qquad\qquad\qquad\qquad\quad\ \ ; t_{prev} := \tau; t_{exp} := \sigma u \\
\qquad\qquad\qquad\qquad\quad\ \ ; t_{exp} := \sigma u + \tau \\
\qquad\qquad\qquad\qquad\quad\ \ ; t_{send} := \max(\tau, t_{prev} + \tfrac{t_{exp}}{\lambda}) \\
\quad] \\
; dum := \texttt{closematlab}() \\
]\!]
\end{array}
\qquad (\chi\text{-}6.1)
$$

2. Interface.

Since χ cannot communicate directly with Matlab, an interface is required. Python has been chosen as an interface, since a Python-to-Matlab communication is present with the Pymath module. Furthermore, the Pymath module can be imported in the χ file. Then, Matlab and Simulink can accessed in the χ file. With Pymath, the χ function *runmodel* has been defined, see Appendix F.2 for the code of the python script. Function *runmodel* runs the Simulink model with the current wip-level as income and a new arrival rate λ as outcome.

Results of implementation

Two kinds of results will be treated from the control implementation. First of all, one can be interested in a single simulation run. Then one would like to know how the controller works and what kind of control signal has been given in a single simulation. Second, the interest goes to multiple simulations runs. Then, one can see if the controller provides the desired average limiting behavior. Before discussing the average results of the controller, a single simulation will be treated.

The working of the controller will be looked upon from the χ outputs' perspective. Consequently, the χ output signal of interest is the wip-level (w_χ). The controlled χ model is a GW_2W_2E queue with identical process rates of $\mu = 1.0$ and a desired utilization of $\rho = 0.9$. So, the average input should be an arrival rate of $\lambda = 0.9$. Finally, a sample time of 1% of the process rate has been chosen, thus $dt = 0.01$.

One can expect that the controller adjusts input rate λ depending on the actual wip-level in the system. A wip-level of $\overline{w} \approx 2.38$ corresponds with an utilization $\rho = 0.9$. Of course, a wip-level has an integer value and, consequently, the expected arrival rate cannot be $\lambda = 0.9$. The controlled input rate should be higher for a wip-level lower than $\overline{w} \approx 2.38$. On the other side, a higher controlled input should result from a wip-level lower than steady state. So, the actual wip-level and the input rate have to be considered in the single run evaluation. Furthermore, since the steady state wip-level cannot occur in a single simulation run, the observer has to keep up with the current wip-level. Therefore, the observer estimate is of interest as well for the evaluation. Accordingly, Figure 6.4 shows he controlled input rate λ (--), the estimate wip-level of

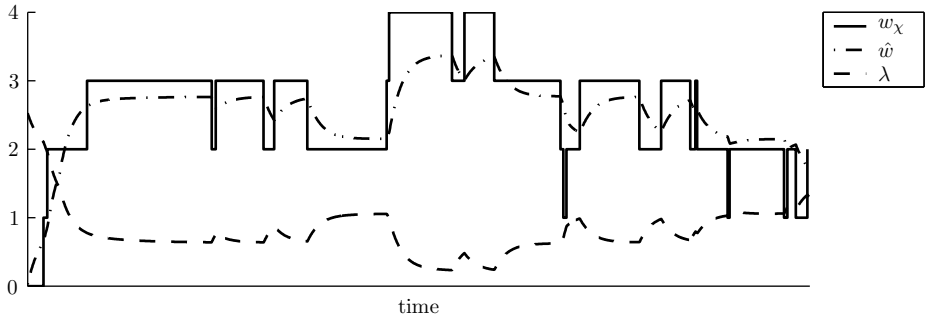


Figure 6.4: *Result of the implementation*

the observer $\hat{w}(\cdot-)$ and of the χ output w_χ in time. The estimate of the observer has to meet two situations: a decrease and an increase from the reference (steady state) wip-level. For wip-levels three and four, the input signal has to be lower than the reference input, $\lambda = 0.9$. Otherwise, for wip-levels zero to two, the input rate has to be higher than $\lambda = 0.9$, since this is the corresponding input rate of a steady state wip-level of $\overline{w} \approx 2.38$. In the graph, the input rate meets these requirements. The graph also shows that the estimate of the observer stays behind on the wip-level for the step from wip-level two to three. The observer cannot reach the wip-level, since the observer always leaves a little difference between the estimate and the actual value. The difference between the observers' wip-level and DEM output can result in poor overall behavior. Consequently, an average of multiple simulations has to be taken.

In Figure 6.5, the average result of multiple simulations are shown. In this figure, the controlled and uncontrolled wip-levels have been plotted for the GW_2W_2E queue in time. The simulation results for the controlled system do not meet the desired steady state value of the uncontrolled system. However, the transient phase does meet the requirements, since the transient controlled wip-level increases more than the uncontrolled wip-level. The discrepancy between both steady states can be related to several circumstances. First, the difference can be a result of the discrete wip-levels, e.g. the wip-level jump from two to three is not reached by the estimate in Figure 6.4. Second, the sample time can be the deciding factor of different results. With a higher sample time, the observer is able to keep up more frequently with the controller. Third, an

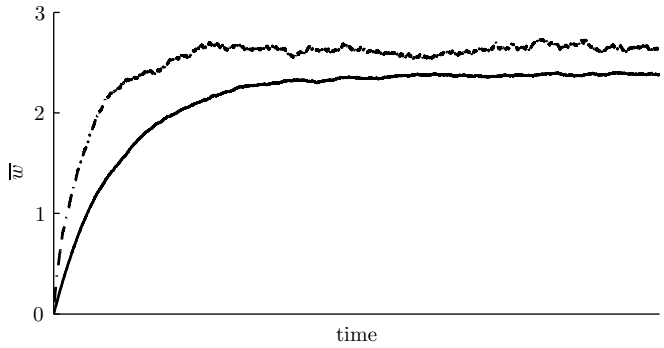
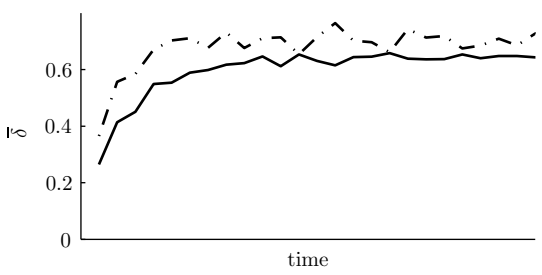


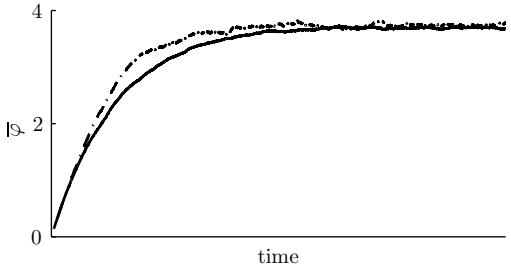
Figure 6.5: *Result of the implementation for the uncontrolled (–) and controlled (–·) wip-level*

incorrect model can be a result of the steady state difference. Due to continuously changing input rates λ the used observer model is not correct, since the model has been composed with constant arrival rates. These rates have been determined from the process rate and the desired utilization. Finally, the steady state error can be related to the effect of the choice of the control gain K and the observer gain L.

In addition to the wip-level, responses from other performance measures of the control model are also of interest. Therefore, Figure 6.6 shows the results of the controlled and uncontrolled GW_2W_2E queue. In Figure 6.6, the controlled model has a higher



(a) *Throughput of uncontrolled (–) and controlled (·–) model*



(b) *Flow time of uncontrolled (–) and controlled (·–) model*

Figure 6.6: *Result of the implementation for the throughput and the flow time*

throughput and an approximately similar flow time for steady state.

6.3 Résumé

The development of a controller has been treated in this chapter. The controller has been designed with the state feedback method of standard control theory. The state feedback controller has been made in Simulink with the earlier derived Markov model. The Markov model estimates all system states with a system output signal. The considered output signal should be obtained from the χ model.

As the χ model's output, the wip-level has been chosen, since determining this performance measure is straightforward. For the input signal, a straightforward technique has been chosen as well. The selected input signal will provide the controlled system with an arrival rate. The arrival rate input and the wip-level output have to passed with a communication between the controller in Simulink and the DEM in χ. The required communication has been established with Python as an interface to be able to test the controlled DEM.

The controlled DEM has been tested with a single run simulation and with averages of multiple simulations. The results of the single run simulation fulfilled the expectations, since an increase of the arrival rate occurs when the actual wip-level is below the steady state wip-level. Furthermore, a decrease of the arrival rate happens when the actual wip-level is lower than the the steady state wip-level.
A successful test has not been obtained for the averages of multiple simulations. These averages consider performance measures of the flow time, throughput and wip-level. The wip-level should result in a the desired reference steady state value. Unfortunately, the wip-levels do not match the reference steady state value. Perhaps, the steady state difference occurs due to the discrete wip-levels, since a continuous model does not accounter discrete values. A second possibility of the difference can be the choice of the gain K or the observability gain L. Finally, the significance of Markov model can be reduced, because the arrival rate is not a constant exponentially distributed rate anymore.

In all, a partially successful control implementation has been performed on a DEM. The DEM wip-level has been used as the only output signal for the controller, while other performance measures have not been examined as output signal. Studying these other performance measures can be relevant for the control of manufacturing systems. Furthermore, the study of achieving the reference steady state value can be considered.

Chapter 7

Conclusions and Recommendations

In this chapter, findings of this study are discussed. In the first section, conclusions are presented and finally , recommendations are made for future research.

Conclusions

The conclusions are divided into four categories. The first category deals with properties of manufacturing systems. Second and third, conclusions for infinite and finite queueing models are discussed. In the last category, conclusions of the control implementation will be treated.

Properties of manufacturing systems

During the search for properties of manufacturing systems, the focus has been put on scaling properties of behavior in time. By introducing scaling properties, a new time scale has been introduced to result in independency for fluctuations of that particular modeling parameter. Three types of scaling properties have been presented which should create independency of parameter changes in the process rate, utilization and number of workstations. Each scaling factor has been studied to determine the possible related scaling factor. This scaling factor enables someone to compare performance measures in time under different conditions of the process rate, utilization and number of workstations. The process rate, utilization and number of workstations should be isolated in the compared behavior. Then, the comparison between models with different conditions can be used as a validation method. Below, the results are presented for each suggested scaling factor.

Process rate The process rate property makes time dimensionless by multiplying it with the process rate. With the process rate property, different $M/M/1$ systems can show identical behavior in dimensionless time for e.g. wip-levels. The adopted identical behavior holds under the conditions that the utilization and number of workstations are the same in the $M/M/1$ queues. For a single server $M/M/1$ queue, Markov theory has been used to proof the process rate property. Besides, the process rate property has been validated for queues in series.

Utilization In contrary to the process rate property, an exact solution of the utilization property has not been found with Markov theory. Consequently, one has to settle with an approximation. This approximation has been formed by the relaxation times. The relaxation time indicates the speed of approach to the stationary situation.

Number of workstations An exact solution for the workstation property has not been found, but the influence of number of workstations in a system has been determined with an approximation. The approximation reduces the influence per workstation on the whole system when the number of workstations increases.

Infinite queueing models

Three modeling techniques have been used to describe infinite queues, the single server $M/M/1$ queues and $M/M/1$ queues in series.

First, the single server $M/M/1$ queue has been modeled using Markov chains and processes. Deriving a time-dependent model with Markov theory appeared to be very complex for even the simplest queueing systems. Besides, the Markov model has a limiting applicability, since only a model of a single server queue has been derived.

Second, DEMs have been used to describe a single server queue as well as multiple $M/M/1$ queues in series. Disadvantages of DEMs include computational expensiveness of simulations and complexity of controller design for large queueing systems.

Third, transfer functions have been modeled to avoid computation expensiveness and control design issues of the DEM. These transfer functions have been derived with ODEs from Markov theory. The infinite number of ODEs has to be rewritten with Z-transformation. Then, a transfer function can be obtained with the use of Rouché theorem, Taylor series and Padé approximation. So, the derived transfer function is an approximation of the $M/M/1$ queue. This approximated transfer function satisfies the earlier proven process rate property. An advantage of the derived transfer function is the ability to expand a single server queue to queues in series. Consequently, a more useable model has been developed than the single server time-dependent $M/M/1$ model. Furthermore, the developed model is perfectly suitable for the design of controllers with straightforward control engineering techniques.
A disadvantage of the developed transfer function is caused by the approximation, since

this approximation reduces the accuracy for high utilizations and extension to queues in series.

Finite queueing models

Similar to infinite queues, three modeling type for finite queues have been discussed in this study: Markov models, DEMs, and transfer functions.

Although, the first modeling type, Markov modeling, is not as complex to compute for finite as for infinite queues, several issues remain. Markov models for $M/M/1/N$ queues in series have the disadvantage of a limiting applicability, since every number of servers requires a new computation. Furthermore, this required computation is still complex. Therefore, expanding the model to queues in series consumes a lot of time.

The DEM is the second considered modeling type. This modeling type can be adjusted easily from infinite to finite buffers. However, the same problems remain for the finite case: computational expensiveness and complex controller development. Furthermore, a new problem occurs, the blocking policy. The standard blocking policies of an $M/M/1/N$ model and GB_NME model do not correspond. Therefore, one of these models has to be adjusted to meet the other blocking policy.

The third modeling type considers transfer functions that have been derived with ODEs from Markov theory. The derivation of these transfer functions is much easier for finite queues compared to infinite queues. However, the finite queue transfer function has a lower usability, since it is not possible to extend the single server transfer function to transfer functions for queues in series. This disadvantage can be overcome when an algorithm derives the transfer function of the Markov ODE probability relations. A more important disadvantage that cannot be solved immediately is the impossibility of deriving a transfer function for the flow time.

Control implementation

In Chapter 6, an implementation of a controller has been presented. The integration of a controller on a χ model has been successfully accomplished.

The implementation has been performed for two queues in series with a single buffer and one machine place, a GW_2W_2E queue. This queueing system has been modeled with a Markov based transfer function. The obtained transfer function has been used for a state space notation to develop the observer for a state feedback controller. The feedback controller has been designed in Simulink. The Simulink controller has been connected to the χ model with Python to be able to perform a simulation.

A single simulation of the controlled χ model showed an increase of the input rate for a steady state value lower than the reference value. Furthermore, a steady state value higher than the reference value leads to a decrease of the input rate. Unfortunately, the

reference signal is exceeded in steady state when multiple simulation runs are performed. So, the control implementation has succeeded partially, the expectations of a single simulation have been reached, but the desired steady state has not been obtained in multiple runs.

Recommendations

Several questions have come forward in this study that remain unanswered. These open issues will be discussed in this section.

Properties of manufacturing system

The search for relevant properties of manufacturing systems appeared to be very difficult. Therefore, only properties with respect to scaling have been treated. One can expect that more properties exist like the physical property which prohibits backward flow. Consequently, another perspective can be used to search for other properties that are not related to scaling with the use of Markov models. Besides, more research can be performed on the effect on validation of the process rate property with e.g. PDE models. Finally, within the scaling properties one can study the existence of nonlinear relations between queueing systems with different parameters like the utilization.

Approximated transfer function of an $M/M/1$ queue

The approximated transfer function, defined in this thesis, appeared to be very applicable and useable for control. The major disadvantage of the determined transfer function is the accuracy decrease for high utilizations due to linearization. So, this linearization has to be avoided to increase the accuracy. For example, the possibility of developing a nonlinear control law can be investigated. This nonlinear control law has to be derived using the nonlinear probability relation of the $M/M/1$ queue. With the nonlinear relation, the exact expression of the $M/M/1$ queue will be used. Therefore, the controller should hold at every utilization for $\rho < 1$ and the controller should control the manufacturing system perfect.

Exponential distribution

The models in this thesis come along with an exponential distribution. Research on a general distribution can be of interest as well.

Markov theory can still be used with non exponential distributions, for a coefficient of variation $c \neq 1$. With $c < 1$, the Erlang distribution can be used which is based on a sequence of identical exponential distributions. The distribution sequence allows several

exponential phases in the arrival and process rates. The exponential phase corresponds with a Markov state, where a service completion consists of a sequence of states.

A sequence of states can be defined for $c > 1$ as well. For $c > 1$, a study can be made which uses a sequential exponential distribution with different means, a Coxian distribution. Now, an identical traject of creating transfer functions can be followed with the use for control.

Finite queue transfer function

The applicability of the finite queueing transfer function models seems endless, but there will be a maximum number of states in the model. Especially, the inverse of the probability function asks for computation time and is a limiting factor.

Another shortcoming is the lack of an expression for the flow time. The flow time is a highly important parameter to decrease in manufacturing. Consequently, research on the flow time could be performed, since controlling the flow time is desired in a lot of situations. Although the flow time cannot be expressed, one can compute the flow time numerically. Furthermore, the wip-level and flow time can be seen as a dual problem, since the basic difference is hooked orientation. With the orientation and a numerical data, creating a control law for flow time seems possible with the use of Markov theory.

Control

In this thesis, the focus has been put on modeling techniques and scaling properties that are relevant for considered framework. The control part of the framework has been introduced to show the effect of a controller on the entire system. The result of the controlled system has not been totally satisfied, since the focus was not put on control. Therefore, the issues of the control implementation need some extra attention.

Extra effort can be made for the discrete wip-level, which can give problems for the reference output signal. A reference signal for other performance measures result in issues that need to be addressed for the throughput and flow time. Reference signals of the throughput and flow time contain the issue of having only one sample that can be obtained by the DEM over an unknown time span. Besides, a throughput and flow time can have inaccuracies, since the flow time and throughput are derived from a single lot. This single lot contains information about that particular lot and not of the system as a whole. These remaining issues need to be addressed in other research studies.

Bibliography

[Ada02] I. Adan and J. Resing. *Queueing theory*. Lecture notes, Eindhoven University of Technology, Department of Mathematics and Computer Science, 2002.

[Ber04] R.A. van den Berg. Partial differential equations in modelling and control of manufacturing systems. Master's thesis, Technische Universiteit Eindhoven, Systems Engineering Group, 2004.

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

[Coh82] J.W. Cohen. *The single server queue*. North-Holland, Amsterdam, 1982.

[Dag95] C.F. Daganzo. Requiem for second-order fluid approximations of traffic flow. *Transportation Research. Part B, Methodological*, 29(4):277–286, 1995.

[Erd54] A. Erdelyi, W. Magnus, F. Oberhettinger, and F.G. Tricomi. *Tables of Integral Transforms*. McGraw-Hill, New York, 1954.

[Fra94] Gene F. Franklin, Abbas Emami-Naeini, and J. David Powell. *Feedback Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., third edition, 1994.

[Hop01] W.J. Hopp and M.L. Spearman. *Factory physics: Foundations of Manufacturing Management*. Irwin/McGraw-Hill International Editions, Singapore, second edition, 2001.

[Jac54] R.R.P. Jackson. Queueing systems with phase-type service. *Operational research quarterly*, 5, 1954.

[Jac63] J.R. Jackson. Jobshop-like queueing systems. *Management Science*, 10, 1963.

[Ken53] D.G. Kendall. Stochastic processes occuring in the theory of queues and their analysis by the method of the imbedded markov chain. *Ann. Math. Stat*, pages 338–354, 1953.

[Khi32] A. Khinchine. Mathematical theory of stationairy queues. *Mat. Sbornik*, 39:73–84, 1932.

[Kle75] L. Kleinrock. *Queueing systems*, volume I: Theory. Wiley-Interscience, London, 1975.

[Kul95] V.G. Kulkarni. *Modeling and analysis of stochastic systems.* Chapman-Hall, London, 1995.

[Kul99] V.G. Kulkarni. *Modeling, analysis, design and control of stochastic systems.* Springer-Verlag, New York, 1999.

[Law00] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill Higher Education, New York, third edition, 2000.

[Lit61] J.D.C. Little. A proof of queueing formula $l = \lambda w$. *Operations Research*, (9):338–387, 1961.

[Pla04] S. Platschorre. Modelling of manufacturing lines using higher order PDEs. Master's thesis, Technische Universiteit Eindhoven, Systems Engineering Group, Eindhoven, 2004.

[Pol30] F. Pollazcek. Über eine Aufgabe der Wahrscheinlichkeitstheorie. *I-II Math. Zeitschrift*, 32:64–100, 729–750, 1930.

[Roo03] J.E. Rooda and J. Vervoort. *Analysis of manufacturing systems.* Lecture notes, Eindhoven University of Technology, Department of Mechanical Engineering, 2003.

[Sag71] A.P. Sage and J.L. Melsa. *System identification.* Academic Press, London, 1971.

[Sha00] O.P. Sharma and A.M.K. Tarabia. A simple transient analysis of an $M/M/1/N$ queue. *Sankhyā Ser. A*, 62(2):273–281, 2000.

[Tak62] L. Takacs. *Introduction to the theory of queues.* Oxford University Press, Oxford, 1962.

[Ver03] J. Vervoort and J.E. Rooda. *Learning χ 0.8.* Preliminary version, Eindhoven University of Technology, Department of Mechanical Engineering, 2003.

[Wol82] R.W. Wolff. Poisson arrivals see time averages. *Oper. Res.*, 30, 1982.

Appendix A

Time-dependent $M/M/1$ behavior

The time-dependent solution of an $M/M/1$ queue has been derived in this appendix. Here, almost all essential and less essential steps are worked out. The forward Chapman-Kolmogorov's equations form the starting point for the derivation:

$$\frac{dP_k(t)}{dt} = -(\lambda+\mu)P_k + \lambda P_{k-1}(t) + \mu P_{k+1}(t) \quad \forall\ k = 1, 2, \ldots \tag{A.1}$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t). \tag{A.2}$$

The first step is the transition to a continuous domain. This is performed by the Z-transform:

$$P(z,t) \triangleq \sum_{k=0}^{\infty} P_k(t) z^k \tag{A.3}$$

Now, the k^{th} differential equation is multiplied by z^k, leading to:

$$\sum_{k=1}^{\infty} \frac{dP_k(t)}{dt} z^k = -(\lambda+\mu)\sum_{k=1}^{\infty} P_k(t) z^k + \lambda \sum_{k=1}^{\infty} P_{k-1}(t) z^k + \mu \sum_{k=1}^{\infty} P_{k+1}(t) z^k.$$

After some rearranging, this yields to:

$$\frac{\partial}{\partial t}\big[P(z,t) - P_0(t)\big] = \\ -(\lambda+\mu)\big[P(z,t) - P_0(t)\big] + \lambda z P(z,t) + \frac{\mu}{z}\big[P(z,t) - P_0(t) - zP_1(t)\big].$$

Certain terms can be eliminated using the equation for $k = 0$ and it results in:

$$\frac{\partial}{\partial t} P(z,t) = -\lambda P(z,t) - \mu[P(z,t) - P_0(t)] + \lambda z P(z,t) + \frac{\mu}{z}[P(z,t) - P_0(t)].$$

After rearranging this equation the following equation is obtained:

$$z\frac{\partial}{\partial t}P(z,t) = (1-z)\big[(\mu-\lambda z)P(z,t) - \mu P_0(t)\big] \tag{A.4}$$

To lose the time derivative Laplace-transformation has been applied, see:

$$z\big[sP^*(z,s) - P(z,0^+)\big] = (1-z)\big[(\mu-\lambda z)P^*(z,s) - \mu P_0^*(s)\big].$$

Rearranging gives

$$P^*(z,s) = \frac{zP(z,0^+) - \mu(1-z)P_0^*(s)}{zs-(1-z)(\mu-\lambda z)} \tag{A.5}$$

With initial condition $P(z,0^+) = z^i$

$$P^*(z,s) = \frac{z^{i+1} - \mu(1-z)P_0^*(s)}{zs-(1-z)(\mu-\lambda z)}$$

To determine $P_0^*(s)$, the roots of the denominator have been used,

$$g(z) = \lambda z^2 - (\lambda+\mu+s)z + \mu$$

In the unit circle of $|z| < 1$ the denominator has only one root, since $g(0) = \mu > 0$ and $g(1) = -s < 0$. Therefore the numerator must have that same root, since $\sum_{k=0}^{\infty} z^k = \frac{1}{1-z}$ is finite at all time.

$$\zeta_1(s) = \frac{(\lambda+\mu+s) - \sqrt{(\lambda+\mu+s)^2 - 4\lambda\mu}}{2\lambda}$$

Using this in the root of the numerator

$$P_0^*(s) = \frac{\zeta_1(s)^{i+1}}{\mu(1-\zeta_1(s))}$$

Let

$$\zeta_2(s) = \frac{\mu}{\lambda\zeta_1(s)}$$

be the other root, now return to

$$\begin{aligned}
P^*(z,s) &= \frac{\mu(1-z)P_0^*(s) - z^{i+1}}{\lambda(z-\zeta_1(s))(z-\zeta_2(s))} \\
&= \frac{\mu(1-z)P_0^*(s) - z^{i+1}}{\lambda(z-\zeta_1(s))(z-\zeta_2(s))} \frac{(z-\zeta_1(s)) - (z-\zeta_2(s))}{\zeta_2(s)-\zeta_1(s)} \\
&= \frac{\mu(1-z)P_0^*(s) - z^{i+1}}{\lambda(\zeta_2(s)-\zeta_1(s))} \left(\frac{1}{z-\zeta_2(s)} - \frac{1}{z-\zeta_1(s)}\right) \\
&= \frac{\mu(1-z)P_0^*(s) - z^{i+1}}{\lambda(\zeta_2(s)-\zeta_1(s))} \left[\frac{1}{\zeta_1(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_1(s)}\right)^n - \frac{1}{\zeta_2(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_2(s)}\right)^n\right].
\end{aligned}$$

Rearranging gives:

$$P^*(z,s) = \frac{\mu P_0^*(s)}{\lambda(\zeta_2(s)-\zeta_1(s))}\left[\left(\frac{1}{\zeta_1(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_1(s)}\right)^n - \frac{1}{\zeta_2(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_2(s)}\right)^n\right)\right.$$
$$\left. - \left(\frac{z}{\zeta_1(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_1(s)}\right)^n - \frac{z}{\zeta_2(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_2(s)}\right)^n\right)\right]$$
$$- \frac{1}{\lambda(\zeta_2(s)-\zeta_1(s))}\left[\left(\frac{z^{i+1}}{\zeta_1(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_1(s)}\right)^n - \frac{z^{i+1}}{\zeta_2(s)}\sum_{n=0}^{\infty}\left(\frac{z}{\zeta_2(s)}\right)^n\right)\right].$$

With the above equation the derivation of an inverse Z-transform is possible for $k = 0, 1, 2, \ldots$

$$P_k^*(s) = \mu P_0^*(s)(a_k - a_{k-1}) - a_{k-i-1} = \frac{z_1(s)^{i+1}}{1-z_1(s)}(a_k - a_{k-1}) - a_{k-i-1}$$

where

$$a_n = \begin{cases} \dfrac{1}{\lambda(\zeta_2(s)-\zeta_1(s))}\left(\dfrac{1}{\zeta_1(s)^{n+1}} - \dfrac{1}{\zeta_2(s)^{n+1}}\right) = \dfrac{1}{\mu}\displaystyle\sum_{j=0}^{n}\left(\frac{\lambda}{\mu}\right)^{n-j}\zeta_2(s)^{n-2j} & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases} \tag{A.6}$$

After some simplification:

$$P_k(s) = \frac{1}{\lambda}\left[\sum_{j=i+k+1}^{\infty}\left(\frac{\mu}{\lambda}\right)^{j-k-1}\zeta_2(s)^{-k} + \sum_{j=(k-i)\vee 0}^{k-1}\left(\frac{\lambda}{\mu}\right)^{k-i-j}\zeta_2(s)^{-(2j-k+i+1)}\right] \tag{A.7}$$

according to [Erd54]†

$$\int_0^{\infty} \mathrm{e}^{-\rho t} b^j I_j(bt)dt = \frac{\left(\rho - \sqrt{\rho^2-b^2}\right)^j}{\sqrt{\rho^2-b^2}} \quad Re\ \rho > b,\ j = 0, 1, \ldots$$

and,

$$\int_0^{\infty} \mathrm{e}^{-pt}\ n\ a^{-n}\frac{I_n(at)}{t}dt = \left(p + \sqrt{p^2-a^2}\right)^{-n} \tag{A.8}$$

This results in the inverse Laplace transform of $\zeta_2(s)^{-j}$

$$\mathrm{e}^{-(\lambda+\mu)t} k \rho^{k/2} \frac{I_k(2\mu t\sqrt{\rho})}{t}$$

†Volume I, p.237, property [49]

Where I_k is the modified Bessel function of the first kind and with identity, inverting results in,

$$p_k(t) = \frac{\mathrm{e}^{-(\lambda+\mu)t}}{\lambda t}\Bigg[\rho^{k+1}\sum_{j=k+i+1}^{\infty}\rho^{k/2}I_j(2t\sqrt{\lambda\mu}) + \rho^{(k-i+1)/2}\sum_{j=(k-i)\vee 0}^{k-1}(2j-k+i+1)I_{2j-k+i+1}(2t\sqrt{\lambda\mu})\Bigg] \quad \text{(A.9)}$$

Using the identity

$$\frac{2k}{x}I_k(x) = I_{k-1}(x) - I_{k+1}(x),$$

then some simplification leads to the final solution is:

$$p_k(t) = \mathrm{e}^{-(\lambda+\mu)t}\Bigg[\rho^{(k-i)/2}I_k(2t\sqrt{\lambda\mu}) + \rho^{(k-i-1)/2}I_{k+i1}(2t\sqrt{\lambda\mu}) + (1-\rho)\rho^k\sum_{j=k+i+2}^{\infty}\rho^{-j/2}I_j(2t\sqrt{\lambda\mu})\Bigg] \quad \text{(A.10)}$$

Appendix B

χ code

B.1 χ code $M/M/1$ or $G(BM)^nE$

```
from random import *
from std import *
// chi 0.8

type lot = nat#real

const ni  : nat  = 1 //number of workstations
,     np1 : nat  = 2 //number of workstations plus 1

// generator
proc G (a:!lot, u, te: real, bufi: nat)=
|[ buf, n:nat, d:->real
 | n:=1; d:= negexp (te/u); buf:=ni*bufi
 ; *[ buf > 0 -> !0, tab (), 0.0, nl (); buf:= buf - 1]
 ; *[ true -> delta sample d; a!<n, time>; n:=n+1; !0, tab (), time, nl ()]
]|

// buffer
proc B (a:?lot, b:!lot, bufi: nat)=
|[ xs:lot*, x:lot
 | xs:= []
 ; *[ len (xs) < bufi -> xs:= xs ++ [<0,0.0>] ]
 ; *[ true; a?x -> xs:= xs ++ [x]
    | len (xs)>0; b!hd (xs) -> xs:= tl (xs)
    ]
 ]|

// machine
proc M (a:?lot, b:!lot, te: real)=
|[ x: lot, d:->real
 |  d:= negexp (te)
 ; *[ true -> a?x; delta sample d; b!x
    ]
]|

// exit
proc E (a:?lot)=
|[ x:lot
```

```
 | *[ true; a?x -> !1, tab (), time, nl ()
    ]
]|

// cluster declaration
clus S (u,te: real, bufi: nat)=
|[ a :  (-lot)^np1, b :  (-lot)^ni
 | G (a.0,u,te,bufi)
|| i:nat <- 0 .. ni: B (a.i,b.i,bufi)
|| i:nat <- 0 .. ni: M (b.i,a. (i+1),te)
|| E (a.ni)
]|

xper (u,te: real, bufi: nat) = |[ S (u,te,bufi) ]|
```

B.2 χ code $M/M/1/N$ queue

```
from random import *
from std import *
// chi 0.8

type lot = nat#real

const ni  : nat  = 2 // number of workstations
,     np1 : nat  = 3 // number of workstations plus 1
,     nm1 : nat  = 1 // number of workstations minus 1

proc G(a:!lot, rho, mu:real, z:?void)=
|[ i:nat, d:->real
 | i:=1; d:= negexp(1/rho/mu)
 ; *[ true
      -> z?; delta sample d; a!<i, time>
       ; i:= i + 1; !0, tab(), time, nl()
    ]
]|

proc B(a:?lot, b:!lot, N:nat, z:!void)=
|[ xs:lot*, x:lot, sent: bool
 | xs:= []; sent:= false
 ; *[ len(xs) < N-1; z!     -> sent:= true
    | sent; a?x             -> xs:= xs ++ [x]; sent:= false
    | len(xs) > 0; b!hd(xs) -> xs:=tl(xs)
    ]
 ]|

proc M(a:?lot, b:!lot, te: real, z:?void)=
|[ x: lot, d: -> real, buffull, idle: bool, t:real
 | d:= negexp(te); buffull:= false; idle:= true
 ; t:= sample d + time
 ; *[ true
      -> a?x
      ; z?; delta sample d; b!x
    ]
]|

proc Me(a:?lot, b:!lot, te: real)=
|[ x: lot, d:->real
 |  d:= negexp(te)
 ; *[ true -> a?x; delta sample d; b!x
```

```
    ]
]|

proc E(a: ?lot)=
|[ x: lot
 | *[true; a?x -> !1, tab(), time, nl()]
]|

// Cluster declaration
clus S(rho,mu: real,N: nat)=
|[ a: (-lot)^np1, b: (-lot)^ni, z: (-void)^ni
| G(a.0,rho,mu,z.0)
|| i: nat <- 0 .. ni: B(a.i,b.i,N,z.i)
|| i: nat <- 0 .. nm1: M(b.i,a.(i+1),mu,z.(i+1)) // forall nm1 > 0
|| Me(b.nm1,a.ni,mu)
|| E(a.ni)
]|

xper(rho,mu: real, N: nat) = |[ S(rho,mu,N) ]|
```

Appendix C

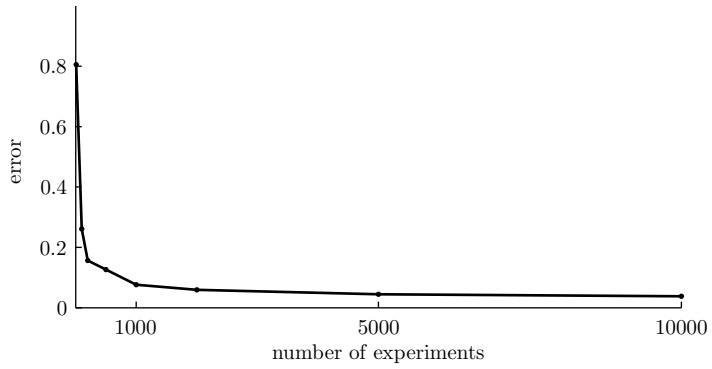
Averaging discrete events

Two methods of averaging can be distinguished:

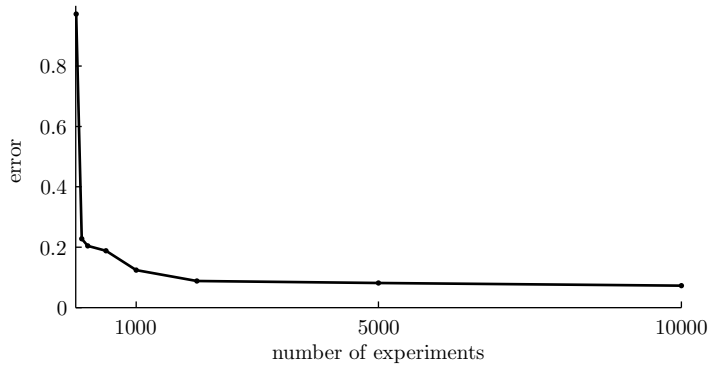
Event averaging event averaging, takes the mean of events at every point in time of the simulation clock. This results in an average at every time unit. The only problem that occurs is at times where two or more events occur. For example, event i ends and event $i+1$ starts. The question arises which event has to be used in the average. In this case, only input events are used. Consequently, a graph can be drawn as in Figure 3.3(a). Herein, the policy is used to take the mean of $t_{i,1}, t_{i,2}, \ldots, t_{i,n}$ for n simulations at event i.

Time-averaging The second possibility is to take the event numbers at a certain time τ. So, for every τ all events are averaged, see Figure 3.3(a).

Here, the methods have been compared with each other. Note that, only the points are compared at the number events only, so at $event = 0, 1, 2, \ldots$. For these values, the times are substracted from each other result in an error. It has been performed for both signals, input and output in Figure C.1. From Figure C.1, it is obvious that output behavior is more stochastic in this case. This is logical since input is Poisson process and the output evolves to a Poisson process in time of the simulation clock.



(a) *Input error*



(b) *Output error*

Figure C.1: *Errors between averaging methods in relation with number of simulation*

Appendix D

Powering approximations

This appendix treats reliability of the Taylor and Padé approximations when powered. In the following equations, two suggestion are made. Herein the Taylor approximation can and the Padé approximation cannot be powered:

$$\Big(\mathcal{T}\big(f(x)\big)\Big)^m = \mathcal{T}\big(f^m(x)\big) \tag{D.1}$$

$$\Big(\mathcal{P}\big(f(x)\big)\Big)^m \neq \mathcal{P}\big(f^m(x)\big), \tag{D.2}$$

This holds for a same order approximation. In the functions $f(x)$ used in this thesis, the Taylor approximation can be powered without the loss of accuracy, see (D.1). The Padé approximation however cannot be powered, since the solutions do not have to be the equal of (D.2). First, (D.1) will be treated and (D.1) can be written as:

$$\mathcal{T}_n\big(f(x)\big) = f(0) + f'(0)x + \frac{1}{2!}f''(0)x^2 + \cdots + \frac{1}{(n-1)!}f^{(n-1)}(0)x^{(n-1)} + O(x^n). \tag{D.3}$$

With $m = 2$, (D.3) becomes:

$$\big(f(0)+f'(0)x+\cdots+\frac{1}{(n-1)!}f^{(n-1)}(0)x^{(n-1)}+O(x^n)\big)^2 = f^2(0)+2f(0)f'(0)x+\cdots+O(x^n)$$

$$\mathcal{T}_n\big(f^2(x)\big) = f^2(0) + 2f(0)f'(0)x + \cdots + O(x^n)$$

When, a second order Taylor approximation is considered for $h(s) = (s+a)$ to the $m^t h$ power around $s = 0$:

$$\begin{aligned} \mathcal{T}_2(h(s)) &= a + 2as + O(s^2) \\ \mathcal{T}_2(h^m(s)) &= a^{2m} + 2ma^{2m-1}s + O(s^2). \end{aligned}$$

Now, if $\mathcal{T}_2(h(s))$ is powered, $\mathcal{T}_2(h(s))^m$ has to result in the same solution:

$$\Big(\mathcal{T}_2(h(s))\Big)^m = \Big(a + 2as + O(s^2)\Big)^m = a^{2m} + 2ma^{2m-1}s + O(s^2).$$

For Taylor series, a powering a approximation is not different than taking the original function to the power. For the powering of the Padé approximation the structure of Example 4.1 is used:

$$\begin{aligned}\mathcal{P}_5(\mathcal{T}_4(s)) &= \frac{p_1 s + p_0}{q_2 s^2 + q_1 s + q_0} \\ \mathcal{P}_5(\mathcal{T}_4^m(s)) &= \frac{p_1 s + p_0}{q_2 s^2 + q_1 s + q_0}.\end{aligned}$$

With $m = 2$, these equations give the following solution:

$$\mathcal{P}_5(\mathcal{T}_4(s))^2 = \frac{2\,\frac{(\lambda+\mu)}{\mu^2-2\,\lambda\,\mu+\lambda^2}\,s+1}{\left(-2\,\frac{\mu}{-\mu^3+3\,\lambda\,\mu^2-3\,\mu\,\lambda^2+\lambda^3}+4\,\frac{\mu^2}{(\mu^2-2\,\lambda\,\mu+\lambda^2)^2}\right)s^2+4\,\frac{\mu}{\mu^2-2\,\lambda\,\mu+\lambda^2}\,s+1)} \tag{D.4}$$

$$\mathcal{P}_5(\mathcal{T}_4^2(s)) = \frac{4\frac{\lambda^2}{3\,\lambda^3-7\,\mu\,\lambda^2+5\,\lambda\,\mu^2-\mu^3}\,s+1}{\frac{\mu^2+\lambda^2-6\,\lambda\,\mu}{\mu^4-6\,\mu^3\lambda+12\,\mu^2\lambda^2-10\,\lambda^3\mu+3\,\lambda^4}s^2-2\,\frac{(\lambda^2-4\,\lambda\,\mu+\mu^2)}{3\,\lambda^3-7\,\mu\,\lambda^2+5\,\lambda\,\mu^2-\mu^3}\,s+1}. \tag{D.5}$$

Obviously, (D.4) and (D.5) are not the same. Consequently, (D.2) holds since both methods do not have to be equal.

In short, Taylor approximations can be powered without accuracy loss in comparison with the original function. Whereas, in a Padé approximation power loss can be present in comparison with the original function.

Appendix E

Markov model of a GW_2W_2E queue

The DEM of Appendix B can be formed with two workstations. Then, the GW_nW_nE originates and together with a single buffer and machine place, the GW_2W_2E is born. When the chi code of an GW_2W_2E process is translated to Markov, it does not correspond with the tandem $M/M/1/N$ model. Therefore, the standard $M/M/1/N$ Markov model has to be adjusted to be able to describe the GW_2W_2E queue. The Markov description requires the specification of all states in the system of the GW_2W_2E queue. The following states are present in the system for the idle first workstation:

(0,0) Both workstations are empty;

(0,1) First Workstation is empty and second workstation is occupied by a single lot;

(0,2) First Workstation is empty and second workstation is occupied by two lots (full).

A single blocking situation at the first workstation is present when:

(0,3) First Workstation tries to send a lot and second workstation is full. Thus, blocking occurs between the first and second workstation.

Of course, the machine of the first workstation can be occupied with a single lot:

(1,0) First Workstation is occupied by a single lot and second workstation is empty;

(1,1) Both workstations are occupied by a single lot;

(1,2) First Workstation is occupied by a single lot and second workstation is full.

Again, a blocking possibility occurs when machine has finished processing and the second workstation is full:

(1,3) First buffer and machine are occupied by a single lot. Second workstation is full.

Now, the situations with a full first workstation can be looked upon:

(2,0) First workstation is full and second workstation is empty;

(2,1) Both workstations is full and second workstation is occupied by a single lot;

(2,2) Both workstations are full.

The state for the blocked situation is described as:

(2,3) First workstation is full and second workstation is full. The generator and the first workstation try to send a lot.

In this case, a double blocking situation occurs. Blocking happens between the generator and the first workstation and between the first and second workstation. The three last states of the GW_2W_2E system contain blocking between the generator and first workstation:

(3,0) Generator blocks, first workstation is full and second workstation empty;

(3,1) Generator blocks, first workstation is full and second workstation is occupied by a single lot;

(3,2) Generator blocks and both workstations are full.

With the state specifications, the corresponding wip-levels can be formed, as can be seen in Table E.1 All states can be reached via each other. So, all states are connected with

w	states
0	(0,0)
1	(0,1), (1,0)
2	(0,2), (1,1), (2,0), (3,0)
3	(0,3), (2,1), (2,2), (3,1)
4	(1,3), (2,2), (2,3), (3,2)

Table E.1: *Wip-levels and corresponding states*

each other and with the use of the Markov theory they can be linked using arrival rate λ and process μ_1 and μ_2, of machines one and two respectively. In Figure E.1, an illustration has been made, the transition-state diagram. With Figure E.1, a time-dependent

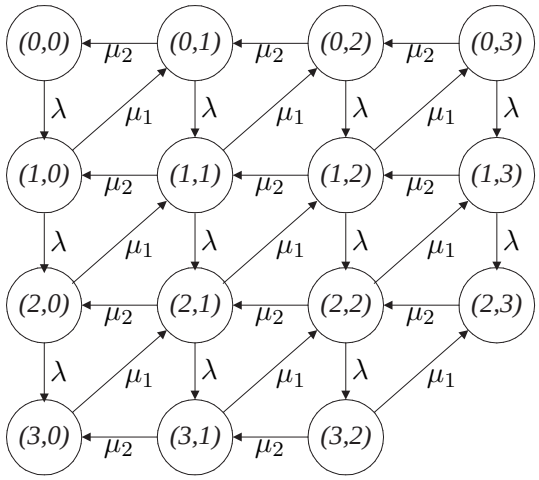


Figure E.1: *State transition diagram of the GW_2W_2E model*

probability notation can be derived easily. The following forward ODE probability relations can expressed:

$$
\begin{aligned}
\frac{dP_{00}(t)}{dt} &= -\lambda P_{00}(t) + \mu_2 P_{01}(t) \\
\frac{dP_{01}(t)}{dt} &= -(\lambda + \mu_2) P_{01}(t) + \mu_1 P_{10}(t) + \mu_2 P_{02}(t) \\
\frac{dP_{02}(t)}{dt} &= -(\lambda + \mu_2) P_{02}(t) + \mu_1 P_{11}(t) + \mu_2 P_{03}(t) \\
\frac{dP_{03}(t)}{dt} &= -(\lambda + \mu_2) P_{03}(t) + \mu_1 P_{12}(t) \\
\frac{dP_{10}(t)}{dt} &= -(\lambda + \mu_1) P_{10}(t) + \lambda P_{00}(t) + \mu_2 P_{11}(t) \\
\frac{dP_{11}(t)}{dt} &= -(\lambda + \mu_1 + \mu_2) P_{11}(t) + \lambda P_{01}(t) + \mu_1 P_{20}(t) + \mu_2 P_{12}(t) \\
\frac{dP_{12}(t)}{dt} &= -(\lambda + \mu_1 + \mu_2) P_{12}(t) + \lambda P_{02}(t) + \mu_1 P_{30}(t) + \mu_2 P_{13}(t) \\
\frac{dP_{13}(t)}{dt} &= -(\lambda + \mu_2) P_{13}(t) + \lambda P_{03}(t) + \mu_1 P_{22}(t) \\
\frac{dP_{20}(t)}{dt} &= -(\lambda + \mu_1) P_{20}(t) + \lambda P_{10}(t) + \mu_2 P_{21}(t) \\
\frac{dP_{21}(t)}{dt} &= -(\lambda + \mu_1 + \mu_2) P_{21}(t) + \lambda P_{11}(t) + \mu_1 P_{30}(t) + \mu_2 P_{22}(t) \\
\frac{dP_{22}(t)}{dt} &= -(\lambda + \mu_1 + \mu_2) P_{22}(t) + \lambda P_{12}(t) + \mu_1 P_{31}(t) + \mu_2 P_{23}(t) \\
\frac{dP_{23}(t)}{dt} &= -\mu_2 P_{23}(t) + \lambda P_{13}(t) + \mu_1 P_{32}(t)
\end{aligned}
$$

$$\begin{aligned}
\frac{dP_{30}(t)}{dt} &= -\mu_1 P_{30}(t) + \lambda P_{20}(t) + \mu_2 P_{31}(t) \\
\frac{dP_{31}(t)}{dt} &= -(\mu_1 + \mu_2) P_{31}(t) + \lambda P_{21}(t) + \mu_2 P_{32}(t) \\
\frac{dP_{32}(t)}{dt} &= -(\mu_1 + \mu_2) P_{32}(t) + \lambda P_{22}(t).
\end{aligned}$$

Appendix F

Control

In this appendix, some aspects of the implementation of a controller are further clarified. In the next section, the Simulink model is visualized. The last section deals with the codes of χ and Python.

F.1 Simulink

The Simulink model of Figure F.1 has been used for the implementation of the controller. In the figure, a feedback law will be created with steady state values u_ss and x_ss and

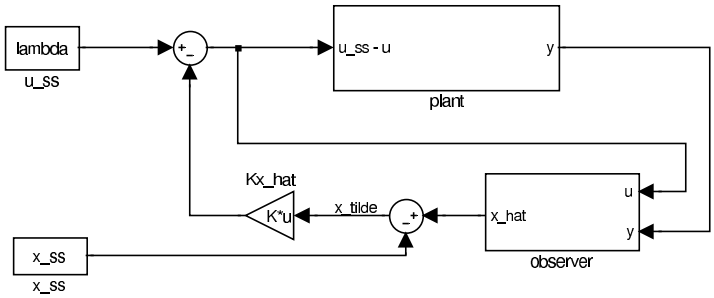


Figure F.1: *Simulink model*

the observer. With an utilization of $rho = 0.9$ and $\mu_1 = \mu_2 = 1.0$, the steady state values will be:

$$\begin{aligned} u\_ss &= 0.9 \\ x\_ss &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2003 \end{bmatrix}. \end{aligned}$$

A control gain K amplifies x_ss and x_hat (observer outcome). The control gain K has been computed with the dynamical model of the observer. The overall input of the plant will be:

$$u = u_{ss} - K(x - x_{ss})$$

The Simulink model consists of two components, the observer and the plant. The observer in the model is visualized in Figure F.2. The observer contains the state space

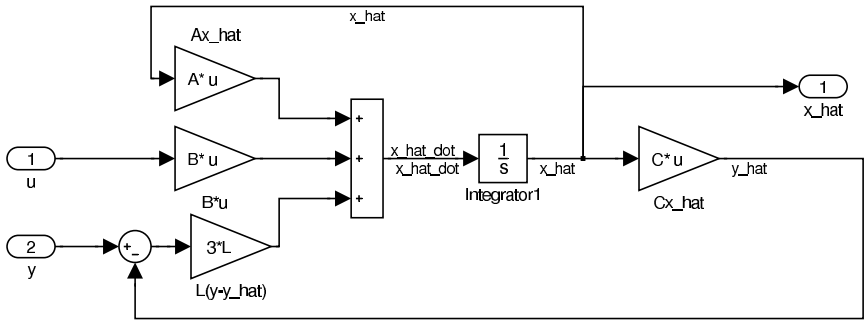


Figure F.2: *observer*

model derived from the Markov theory. The state space includes the following matrices:

$$A = \begin{bmatrix} -29.9 & -6.5 & -0.9 & -0.2 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 & -0.0 \\ 64 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 64 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/8 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$C = \begin{bmatrix} 1.0 & .47 & .10 & .026 & .0094 & .0047 & .0035 & .0019 & .0015 & .0009 & 0.0007 & .0008 & .0010 & .0012 \end{bmatrix}.$$

The Control gain K is computed with Matlab's *lqr* function. Herein, state space matrices A and B are used of the observer. The other two requirements of the *lqr* have been chosen as:

$$\begin{aligned} Q_{lqr} &= C^T \cdot C \\ R_{lqr} &= 1, \end{aligned}$$

where C is a state space matrix from the observer. With the *lqr* algorithm, the control gain becomes

$$K = \begin{bmatrix} 0.7163 & .3308 & .0705 & .0183 & .0065 & .0032 & .0024 & .0013 & .0010 & .0006 & .0005 & .0005 & .0007 & .0008 \end{bmatrix}.$$

The observer gain L has been determined with the rule of thumb Franklin, Emami-Naeini and Powell [Fra94] in which L can be chosen to be two to six times faster than the control gain K. In this case, the observer gain has been chosen to be two times faster than the control gain. The integrator of the observer has an initial value which

is set to the final $\hat{x}$ for each simulation run. The initial value is set to zero, which correspond with an initial empty system.

Finally, the plant communicates with Matlab's workspace which is addressed by the χ file via pymath. The plant model can be seen in Figure F.3. The *ode45* function has

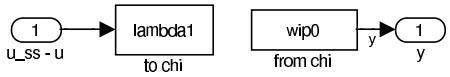


Figure F.3: *plant*

been used to solve the Simulink model.

F.2 Implementation codes

χ code

```
// chi 0.8
// GWnE.chi -> for describing G (Wn)^ (ni)E queues
from random import *
from std import *
from control import *

type lot = nat#real

const ni  : nat  = 2 //number of workstations
,     np1 : nat  = 3 //number of workstations+1

proc G (a:!lot, b:?void, rho, mu :real)=
|[ dum, trysend: bool, i, wip: nat, u:->real
 , tsample, lambda, dt, texp, tprev, tsend: real
 | i:= 0; wip:= 0; u:= negexp (1.0)
 ; lambda:= rho*mu; dt:= 0.01
 ; texp:= sample u; tprev:= 0.0
 ; dum:= openmatlab (); lambda:= runmodel (n2r (wip))
 ; tsend:=max (time, tprev + texp/lambda)
 ; *[ true; b? -> wip:= wip - 1
    | true; delta tsample - time
      -> lambda:= runmodel (n2r (wip))
       ; tsend:=max (time, tprev + texp/lambda)
       ; tsample:= dt +  time
    | not trysend; delta tsend - time
      -> trysend:= true; wip:= wip + 1
    | trysend; a!<i,time>
      -> trysend:= false; i:= i + 1
       ; tprev:= time; texp:= sample u
       ; tsend:= texp/rho/mu + time
       ; !0, tab (), time, nl ()
    ]
 ; dum:= closematlab ()
]|

proc B (a:?lot, b:!lot, N:nat)=
|[ xs:lot*, x:lot
 | xs:= []
```

```
 ; *[ len (xs) < N-1; a?x -> xs:= xs ++ [x]
    | len (xs) > 0; b!hd (xs) -> xs:= tl (xs)
    ]
 ]|

proc M (a:?lot, b:!lot, te: real)=
|[ x: lot, d:->real
 |  d:= negexp (te)
 ; *[ true -> a?x; delta sample d; b!x
    ]
]|

proc E (a:?lot, b:!void)=
|[ x:lot
 | *[ true -> a?x; b!; !1, tab (), time, nl ()
   ]
]|

// Cluster declaration
clus S (rho,mu: real,N: nat)=
|[ a :  (-lot)^np1, b:  (-lot)^ni, z:-void
 | G (a.0,z,rho,mu)
|| i: nat <- 0 .. ni: B (a.i,b.i,N)
|| i: nat <- 0 .. ni: M (b.i,a. (i+1),mu)
|| E (a.ni,z)
]|
```

Python code

```
# control.py
#!usr/bin/python2.2
# Version 2 with Matlab interfacing

from Numeric import *
import pymat

global H

def runmodel (w):
    pymat.put (H,'w',[w])
    pymat.eval (H,'wip0=[0 w]')
    pymat.eval (H,"if exist ('xFinal')==0, xFinal=0, disp (num2str (xFinal)), end")
    pymat.eval (H,'xFinal')
    pymat.eval (H,"sim ('impl_simlink50',[0 .01])")
    pymat.eval (H,'y=lambda1 (end)')
    y=pymat.get (H,'y')
    return y[0]

def openmatlab ():
    global H
    H=pymat.open ("matlab -nosplash")
    return 1

def closematlab ():
    pymat.close (H)
    return 1
```

External library file

```
\\ control.ext
language "python"
file "control"

ext runmodel (w: real)->real
ext openmatlab ()->bool
ext closematlab ()->bool
```