Matching procedure for a discrete fluid flow model

Master Thesis

RICK DUURSMA

June 2007

Vrije Universiteit, The Netherlands, Amsterdam, De Boelelaan 1081a 1081 HV, Department of Mathematics

Blekinge Tekniska Högskolan, Sweden, Karlskrona, Valhallavägen SE-371 41 Karlskrona, Department of Telecommunication Systems
Matching procedure for a discrete fluid flow model

RICK DUURSMA

© Rick Duursma
This thesis was typeset using \LaTeX.
Preface

The program for the master of science in Stochastics and Financial Mathematics is concluded by a research project or an external project to be carried out within a business, industry or research facility. The main goal of such an internship is to develop your skills and abilities and expand your professional network.

Not knowing exactly what I wanted, I contacted Sandjai Bhulai and Rob van der Mei for a subject. The subject ’Equivalent queue model capturing end-to-end performance in mobile networks’ is interesting and opens up many possibilities for the development of new services in a variety of application domains. My main task was to create a model reflecting the properties of observed traffic, because such a model can be used for scaling studies and can serve as a basis for the interpretation of measurements results, e.g. for deriving decisions about performance alarms.

The work was done at the School of Engineering at Blekinge Institute of Technology (BTH) in the context of the Personal Information for Intelligent Transport Systems through Seamless communications and Autonomous decisions (PIITSA) project funded by the Swedish Agency for Innovation Systems VINNOVA (project number 2003-02873). The motto of the BTH is ’Quality and Concern’ and its educational focus is on Applied IT and Sustainable Development of Industry and Society. The training and education offered at the Institute are of the very finest international standards. The Institute co-operates closely with business, government and community at the local, national and international levels.

First of all, I want to thank Rob van der Mei and Kate Wac, who have respectively been my supervisors at the Vrije Universiteit Amsterdam (VU) and the University of Geneva. Because I have a mathematical background, they gave me a warm welcome in the field of telecommunication. Further, thanks also for the support from Lennart Isaksson, Patrik Carlsson and Stevan Chevul in helping me out with computer problems. Finally, I gratefully acknowledge the skilful assistance of my supervisor in Sweden, Markus Fiedler, who, despite his tight schedule always tried to find time to help me out with my problems!
Contents

Preface 4

I Discrete fluid flow model 13

1 Introduction 15

2 Discrete fluid flow model 17

2.1 Model description and model assumptions . . . . . . . . . . . . . . . . . . 17
  2.1.1 General features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
  2.1.2 Model description . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
  2.1.3 Model assumptions . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
  2.1.4 More notation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

2.2 Iterative computation of the simultaneous probability of $X$, $R_{\text{in}}$ and $C$ . . 23
  2.2.1 Preparation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
  2.2.2 Iteration step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

2.3 Derivation of the BRD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
  2.3.1 Derivation of the BRD at the input side . . . . . . . . . . . . . . . . 26
  2.3.2 Derivation of the BRD at the output side . . . . . . . . . . . . . . . 26

2.4 Derivation of the ACF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
  2.4.1 General definition of the ACF . . . . . . . . . . . . . . . . . . . . . 28
  2.4.2 Derivation of the ACF at the input side . . . . . . . . . . . . . . . 29
  2.4.3 Derivation of the ACF at the output side . . . . . . . . . . . . . . . 30

2.5 Loss statistics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

3 Influence of the parameters on the model 35

3.1 Influence of the parameters . . . . . . . . . . . . . . . . . . . . . . . . . . 35
  3.1.1 General statements . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
  3.1.2 Examples . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

3.2 BRDs with a special structure . . . . . . . . . . . . . . . . . . . . . . . . . 38
  3.2.1 A constant BRD at the input side . . . . . . . . . . . . . . . . . . . 38
  3.2.2 A symmetric BRD at the input side . . . . . . . . . . . . . . . . . . 39
  3.2.3 A constant BRD at the output side . . . . . . . . . . . . . . . . . . 39
3.2.4 Symmetric BRD at the output side ........................................... 40
3.3 ACFs with a special structure ...................................................... 40
  3.3.1 ACFs with a special structure at the input side ...................... 40
  3.3.2 ACFs with a special structure at the output side .................... 41
3.4 Combination of the BRD at the input side and at the output side .... 41
  3.4.1 Shared and shaping bottleneck ................................................. 41
  3.4.2 Examples .................................................................................. 42
3.5 Combination of the ACF at the input side and at the output side ... 44
  3.5.1 ACF in case of a shared and a shaping bottleneck ..................... 45
  3.5.2 Examples .................................................................................. 45
3.6 Buffer content ............................................................................. 46
  3.6.1 A constant buffer content .......................................................... 46
  3.6.2 A symmetric buffer content ...................................................... 46
  3.6.3 General feature of the buffer content ....................................... 47

4 Simulation ..................................................................................... 51
  4.1 Introduction ................................................................................. 51
  4.1.1 Preparation ............................................................................... 51
  4.1.2 Confidence intervals ................................................................. 52
  4.2 Simulation of the simultaneous probability of $X$, $R^\text{in}$ and $C$ .... 53
    4.2.1 Single simulation run ............................................................... 53
    4.2.2 Notation and preparation ........................................................ 55
    4.2.3 Results .................................................................................... 55
  4.3 Simulation of the BRD and the ACF at the input side and at the output side ..................................................... 56
    4.3.1 Notation and preparation ........................................................ 56
    4.3.2 Results .................................................................................... 58
  4.4 Simulation of the buffer content ................................................... 59
    4.4.1 Notation and preparation ........................................................ 59
    4.4.2 Results .................................................................................... 60

II Matching procedure .................................................................... 61

5 Genetic Algorithm ....................................................................... 63
  5.1 Basic concepts .......................................................................... 63
  5.2 Fitness function for matching the input side .............................. 65
    5.2.1 Fitness value for the BRD at the input side ........................... 65
    5.2.2 Fitness value for the ACF at the input side ........................... 72
    5.2.3 Total fitness value ................................................................. 72
  5.3 Fitness function for matching the output side ............................ 72
    5.3.1 Adjusting the BRD of the model ............................................ 72
  5.4 Boundaries ................................................................................ 73
    5.4.1 Boundaries at the input side .................................................... 73
### 5.4.2 Boundaries at the output side

- 74

### 5.5 Modifications to the standard GA

- 74
  - 5.5.1 Amount of crossover and mutation
    - 74
  - 5.5.2 Stopping criteria
    - 75
  - 5.5.3 Discretization
    - 76
  - 5.5.4 Start population
    - 76

### 6 Sequential Quadratic Programming

- 79
  - 6.1 Basic concepts
    - 79
  - 6.2 Input parameters
    - 80
      - 6.2.1 Fitness function
        - 80
      - 6.2.2 Boundary constraints
        - 80
      - 6.2.3 Merit function
        - 80
  - 6.3 Modifications to the standard SQP algorithm
    - 81
      - 6.3.1 Stopping criteria
        - 81
      - 6.3.2 Discretization
        - 81

### 7 Matching preparations

- 83
  - 7.1 Factorial designs
    - 83
  - 7.2 Meta-models
    - 84
  - 7.3 Determine the best of several methods
    - 85

### 8 Results

- 87
  - Input side
    - 87
      - 8.1 Choosing the parameters
        - 87
        - 8.1.1 Parameters obtained from the measurements
          - 87
    - 8.2 Determination of $\Delta$ and $p$ from a $2^2$ factorial design
      - 89
    - 8.3 Determination of $\Delta$ and $p$ from a response surface
      - 91
    - 8.4 Determination of the best method
      - 93
    - 8.5 Accuracy of the obtained method
      - 93
    - 8.6 Determination of the optimal input parameters
      - 94
    - 8.7 Matching in practice
      - 95
        - 8.7.1 Influence of the weights
          - 95
        - 8.7.2 Matching problems
          - 96
  - Output side
    - 97
      - 8.8 Choosing the parameters
        - 97
        - 8.8.1 Parameters obtained from the measurements
          - 97
        - 8.8.2 Remaining parameters
          - 98
8.8.3  Several methods .................................................. 98
8.9  Determination of $\Delta$ and $p$ ................................................. 98
  8.9.1  Determination of $\Delta$ and $p$ from a $2^2$ factorial design .......... 98
  8.9.2  Determination of $\Delta$ and $p$ from a response surface .......... 99
8.10  Determination of the best method .............................................. 100
8.11  Accuracy of the obtained method .............................................. 100
8.12  Determination of the optimal capacity parameters ...................... 101
8.13  Matching in practice ....................................................... 103
  8.13.1  Influence of the weights ............................................... 103
  8.13.2  Matching problems ..................................................... 104
8.14  Determination of the optimal capacity parameters under extra restrictions 105

| Combined input side and output side | 106 |
| 8.15  The obtained model | 106 |

9  Outlook ................................................................. 111

Appendix A ............................................................... 112

Appendix B ............................................................... 114
Part I

Discrete fluid flow model
Chapter 1

Introduction

The emergence of 3G networks gives rise to new mobile services in many different areas of our daily life. Examples of demanding mobile services are mobile-healthcare services allowing the continuous monitoring of a patient's vital signs. This type of applications poses strict constraints on the Quality of Service (QoS), for example in terms of service availability and end-to-end response times. Typical response-time related performance metrics are end-to-end delay, delay variation and throughput.

Both the PIITSA project [2] and the Mobihealth project [1] have measured such end-to-end performance metrics in real GPRS and UMTS networks and visualised critical network inputs on application performance. Further, reference [11] presents a passive measurement method for monitoring the network’s impact on quality-sensitive packet streams, by collecting throughput statistics on small time scales at the source host and the destination host. The results presented there reveal a clear influence of the network, seen from variations and autocorrelation of application-perceived throughput. Such an influence has to be considered when choosing the right kind of network for a specific task.

In order to allow for some kind of reverse engineering aiming at finding reasons for certain behaviors observed from the measurements, it would be interesting to identify a queuing model reflecting the properties of the traffic source and basic properties of the equivalent queue. Such a model would allow for both classical queuing analysis and the construction of realistic simulation models. Thus, the model would help to yield results that are hard to reach via measurements. Further, the model can serve as a basis for the interpretation of measurement results, e.g. for deriving decisions about performance alarms from observed behaviors.

Because we are particularly interested in the traffic flow properties, we focus on a discrete fluid flow traffic model to describe the network behaviour. First, we give a model description containing the mathematical details and analyse the model. After that, we describe a procedure to match the model to observations.
The outline of this thesis is as follows. This work is divided in two parts. The first part consists of chapters two, three and four. In chapter two the general features of the model are discussed. Here we derive formulae for an iterative calculation of bit rate summary statistics at the inlet and outlet of the fluid flow buffer, such as the bit rate distribution (BRD) and the autocorrelation function (ACF). Then, in chapter three, we use the derived formulae to analyse the model and we examine the influence of the parameters on the just mentioned bit rate statistics. Chapter four deals with a corresponding simulation which has been implemented. Then, in part two, which runs from chapter five to chapter nine, we describe a matching procedure for the model involving the BRD and the ACF. We start by presenting a stochastic matching method called the Genetic Algorithm (GA). This provides the substance of chapter five. In chapter six another matching approach, called Sequential Quadratic Programming (SQP), is discussed. Further preparations for the matching procedure are the subject of chapter seven. Chapter eight presents the results. Finally, in chapter nine, we draw conclusions and discuss future work.
Chapter 2

Discrete fluid flow model

In this chapter we create the so-called discrete fluid flow model. In the first section we give a model description and discuss the model assumptions. After that, in section two, we describe the behaviour of the buffer. Here-to, it is necessary to compute the simultaneous probability of the buffer content process, the input process and the capacity process. With the aid of this probability we derive the BRD and the ACF for the model at both the input side and the output side. This provides the substance of sections three and four. In the last section we discuss the loss statistics in which we are interested.

2.1 Model description and model assumptions

In this section we give a description of the discrete fluid flow model. We first discuss the general features of the model and explain in detail how the model works. After that, we describe the notation for the parameters and processes involved, and point out the necessary model assumptions.

2.1.1 General features

Network performance management is facing the challenge of handling different QoS issues. For this reason, shortage of network capacity, implying delay or loss has to be identified and to be classified. These tasks imply the need for tractable analytical performance models. Many analytical performance models describe packet processes and focus on the analysis of packet delay and packet loss. The transfer of these results into throughput values, which can be observed from network management tools, is in general difficult. As a consequence, many network administrators base their performance management actions on their experience rather than on performance models. To facilitate model-based bottleneck identification, it is necessary to align measurement-based approaches with tractable analytical performance models. In particular, an analytical model should directly address throughput and load measurements. They should reliably predict the qualitative behaviour of the network.
Taking these requirements into account, the stochastic fluid flow model emerges as an interesting candidate for an analytical performance model. The fluid model considers averages of bit rates on small time scales, because these averages can easily be measured. With the aid of these averages, we obtain analytical estimates on the impact of network entities on data streams and their statistics.

In order to explain the general features of the model, we first need to discuss the term “bottleneck”. A bottleneck is common phrase when referring to shortcomings in the communication facility, so there is not enough capacity to handle incoming packets. Of course this causes the entire process to slow down or to stop, which means the packets will be delayed or even worse, get losted.

In a fluid flow bottleneck disturbances of streams are calculated from comparing required rates with available rates. As long as there is enough capacity, incoming packets pass the bottleneck as if it was not there at all, and shortterm queueing of packets is hardly covered. The model focuses rather on the consequences of temporary overload situations when the demand for capacity exceeds the available capacity of the bottleneck.

Now, figure 2.1 provides a better understanding of how the model works. At the input side of the buffer the incoming data stream, which is varying over time, determines the input flow. Depending on the amount of capacity, the buffer content grows, falls or remains the same, and from this we can derive the output flow. Whenever the buffer size is small, bottlenecks will introduce medium and limited delay but quite large packet losses, and bottlenecks with large buffers are expected to cause large packet delays but only few packet losses.

In order to describe the behaviour of the buffer we need to introduce some notation for the model. This is the topic of the next subsection.

![Figure 2.1: Fluid flow funnel.](image-url)
2.1.2 Model description

Suppose we have an observation window of $\Delta w$ seconds and an averaging interval of $\Delta t$ seconds. In the following we assume that $\Delta w, \Delta t \in \mathbb{N}$ and denote this number by $n$. Interval $s$ is denoted by $(s\Delta t, (s+1)\Delta t]$, where $s \in \{0, 1, \ldots, n\}$ (see figure 2.2).

![Figure 2.2: Observation window.](image)

The time series for the input rate and the output rate are notated by $R^{\text{in}} = \{R^{\text{in}}_s\}_{s=1}^n$ and $R^{\text{out}} = \{R^{\text{out}}_s\}_{s=1}^n$, respectively, and the difference between $R^{\text{in}}$ and $R^{\text{out}}$ by $\Delta R = \{\Delta R_s\}_{s=1}^n$. The buffer content time series is denoted by $X = \{X_s\}_{s=1}^n$, where $X_s$ describes the buffer content at the beginning of interval $s$. The buffer is assumed to be of finite size $K$. Finally, for the buffer capacity time series we use the notation $\{C_s\}_{s=1}^n$. Figure 2.3 shows the fluid flow funnel with the just introduced notation.

Now that we introduced the notation for the set of bottleneck parameters and the processes involved, the buffer content in an arbitrary interval $s$ is determined by

$$X_s = \min(\max(X_{s-1} + (R^{\text{in}}_s - C_{s-1})\Delta t, 0), K). \quad (2.1)$$

Here $(R^{\text{in}}_{s-1} - C_{s-1})\Delta t$ determines the rate with which the buffer content changes, and the minimum and maximum respectively control the cases in which we run into an empty or a full buffer before the end of the averaging interval. In order to determine the amounts of traffic in (2.1), we need to determine the simultaneous probability of $X$, $R^{\text{in}}$ and $C$. Hereeto, it is necessary to make assumptions about the processes involved. The model assumptions are the subject of the next subsection.

![Figure 2.3: Fluid flow funnel with mathematical notation.](image)
2.1.3 Model assumptions

In a standard fluid flow analysis (such as the ones described in [4] and [12]), the queue size variation is approximated to a continuous variable, and the source is modeled by a Markov chain with exponentially distributed state durations. Now, for the discretized version of the fluid flow model we assume something similar, namely, we assume that both processes $R^{\text{in}}$ and $C$ are ergodic, first-order Markov chains on finite state spaces which are independent of each other. Hence, the source is again modeled by a Markov chain, but in this case the state durations are geometrically distributed. From this it follows that the Markov chain satisfies the so called Markov property, which means that the future evolution of the system depends only on the current state of the system and not on its past.

Further, we correspond with each state a certain rate. For the input chain the rates describe the speed with which packets are send by the application, and for the capacity chain the rates describe the speed with which packets can be handled. The differences between the rates in the input chain and the capacity chain determine the rate with which the buffer content changes. Now, if we allow all possible values for the rates, implementation of this model on a computer becomes infeasible. Hereto, we assume the rates to be multiples of certain base rates. This means that all the processes involved become discrete random variables. Figure 2.4 illustrates Markov chains with two states and three states and with transition probabilities and rates denoted by $p$ and $r$, respectively.

In the next subsection we introduce the notation corresponding to the just mentioned model assumptions.

![Figure 2.4: Markov chains with two states (left) and three states (right) with transition probabilities and rates denoted by $p$ and $r$, respectively.](image)
2.1.4 More notation

We first discuss the notation for the two Markov chains $R^{\text{in}}$ and $C$, and then we discuss the notation for the buffer content $X$ and the process $R^{\text{out}}$.

The state spaces for the processes $R^{\text{in}}$ and $C$ are given by

\[
\Omega_{R^{\text{in}}} = \left\{ r^{\text{in}}_1, \ldots, r^{\text{in}}_{M_{R^{\text{in}}}} \right\} \quad \text{and} \quad \Omega_C = \left\{ c_1, \ldots, c_{M_C} \right\},
\]

respectively. As discussed in the previous subsection, the rates $r^{\text{in}}_i$ and $c_j$ are multiples of certain base rates, which we denote by $\Delta r$ and $\Delta c$, respectively. Further, the $l$-step transition matrices with corresponding transition probabilities are given by

\[
P^{(l)}_{R^{\text{in}}} = \left\{ p^{(l)}_{i_1i_2} \right\}_{i_1,i_2 \in \left\{1,\ldots,M_{R^{\text{in}}} \right\}} \quad \text{and} \quad P^{(l)}_C = \left\{ p^{(l)}_{j_1j_2} \right\}_{j_1,j_2 \in \left\{1,\ldots,M_C \right\}},
\]

respectively, where

\[
p^{(l)}_{i_1i_2} = \mathbb{P} \left( R^{\text{in}}_{s+l} = r^{\text{in}}_{i_2} | R^{\text{in}}_s = r^{\text{in}}_{i_1} \right) \quad \text{and} \quad p^{(l)}_{j_1j_2} = \mathbb{P} \left( C_{s+l} = c_{j_2} | C_s = c_{j_1} \right).
\]

In case of single-step transition matrices, we discard the superscripted $(l)$ from the notation. Now, because we assumed the two Markov chains to be ergodic, we know that the corresponding steady-state probability vectors

\[
\pi_{R^{\text{in}}} = \left( \pi^{R^{\text{in}}}_1, \ldots, \pi^{R^{\text{in}}}_{M_{R^{\text{in}}}} \right)^T \quad \text{and} \quad \pi_C = \left( \pi^C_1, \ldots, \pi^C_{M_C} \right)^T
\]

exist and are unique satisfying $\pi^{R^{\text{in}}}_{R^{\text{in}}} P_{R^{\text{in}}} = \pi_{R^{\text{in}}}$ and $\pi^{T}_C P_{C} = \pi_{C}$. Note that, because the states of a finite, aperiodic, irreducible Markov chain are ergodic, it suffices to assume that both chains are aperiodic and irreducible. The existence of the steady-state probability vectors allows us to focus on stationary properties of the model. We refer to [17] for more information about steady-state distributions for Markov chains.

Next, because the differences between the rates in the input chain and the capacity chain determine the behaviour of the buffer, working with base rates implies that the buffer content can take only a finite number of possible values. This means that $X$ is a sequence existing of discrete random variables, hence we denote its state space by $\Omega_X := \{x_1, \ldots, x_{M_X} \}$. In addition, if we denote the grid for the buffer content by $\Delta x$ and recall that the buffer size is given by $K$, then the state space for the process $X$ is given by $\Omega_X = \{0, \Delta x, \ldots, K \}$. In order to determine the grid for the buffer content we use the following procedure. First, we define $\Delta x$ by $\Delta x := |r^{\text{in}}_1 - c_1| \cdot \Delta t$. Then, we run through all possible combinations $(i,j)$, where $i \in \{1, \ldots, M_{R^{\text{in}}} \}$ and $j \in \{1, \ldots, M_C \}$, and replace $\Delta x$ by $\gcd(|r^{\text{in}}_i - c_j| \cdot \Delta t, \Delta x)$ for each combination. Next, we verify if $K$ is a multiple of $\Delta x$, i.e. if $\frac{K}{\Delta x} \in \mathbb{N}$. If this is the case, we are done. Otherwise, we have to change to the smaller grid given by $\gcd \left( \Delta x, K - \Delta x \cdot \left\lfloor \frac{K}{\Delta x} \right\rfloor \right)$. The following example demonstrates this.
procedure.

**Example** Let $\Omega_{R^\text{in}} = \{0, 6\}$, $\Omega_C = \{3\}$, $P_{R^\text{in}}>0$, $K = 5$, $\Delta t = 1$ and $X_0 = 0$. First, we define $\Delta x$ by $\Delta x := |r_1^\text{in} - c_1| \cdot \Delta t = 3$. Then, the only possible combination of $(i, j)$ is given by $(2, 1)$, so that $\Delta x := \gcd(|r_2^\text{in} - c_1| \cdot \Delta t, \Delta x) = \gcd(6, 3) = 3$. Finally, because $\frac{K}{\Delta x} \not\in \mathbb{N}$, we have to adjust the grid to $\gcd(\Delta x, K - \Delta x \cdot \lfloor \frac{K}{\Delta x} \rfloor) = \gcd(3, 2) = 1$. Note that, we can either adjust the grid to $\gcd(\Delta x, K - \Delta x \cdot \lfloor \frac{K}{\Delta x} \rfloor)$ or to $\gcd(\Delta x, K - \Delta x \cdot \lfloor \frac{K}{\Delta x} + 1 \rfloor)$ (see figure 2.5), but due to well-known properties of the greatest common divisor, we have that

$$\gcd \left( \Delta x, K - \Delta x \cdot \left\lfloor \frac{K}{\Delta x} \right\rfloor \right) = \gcd \left( \Delta x, K - \Delta x \cdot \left\lfloor \frac{K}{\Delta x} + 1 \right\rfloor - \Delta x \right)$$

so both methods give the same result.

The last process we need to discuss is the output rate process $R^\text{out}$. The state space for the process $R^\text{out}$ is given by $\Omega_{R^\text{out}} = \{r_1^\text{out}, \ldots, r_{M^\text{out}}^\text{out}\}$. Now, because the process $R^\text{out}$ is in general not a Markov chain (which is shown in section 2.4.3), we do not have to introduce notation for a transition matrix or transition probabilities, as we did for the process $R^\text{in}$. The process $R^\text{out}$ depends on the level of the buffer content and the rate with which the buffer content changes and turns out to be much more complex.
In section 2.3.2 we look at this process in more detail when we discuss the BRD at the output side.

### 2.2 Iterative computation of the simultaneous probability of $X$, $R_{in}$ and $C$

The main task is to describe the behaviour of the buffer. Now, because the buffer's behaviour is determined by equation (2.1), this amounts to computing the simultaneous probability of the processes $X$, $R_{in}$ and $C$. In this section we discuss an procedure to compute this probability in an iterative way. Hereeto, we first need to make some preparations, namely, we need to initialize the model and describe the stationarity assumptions.

#### 2.2.1 Preparation

As mentioned in the introduction of this section, we are interested in the simultaneous probability of $X$, $R_{in}$ and $C$. In the sequel we use the following notation for this probability

$$l_s(x_k, r_{in}^i, c_j) := P(X_s = x_k, R_{in}^s = r_{in}^i, C_s = c_j). \ [x_k \in \Omega_X, r_{in}^i \in \Omega_{R_{in}}, c_j \in \Omega_C]$$

(2.2)

Now, because we assumed the processes $R_{in}$ and $C$ to be Markov chains, we know that the future evolution of these processes depends only on their current state and not on their past. Further, if we take a closer look at equation (2.1), we see that the buffer content in an arbitrary interval is determined by the buffer content in the previous interval plus the rate with which the buffer content changes. Altogether, it seems reasonable to aim for an iterative computation of the probability in the preceding display. Before we can discuss how the probability in an arbitrary interval can be found from the previous interval, there are two main preparations that need to be made. The next two subsections deal with this.

#### Stationarity

In this work we focus on steady-state probabilities. For the two Markov chains $R_{in}$ and $C$ this means that we start from the stationary distributions $\pi_{R_{in}}$ and $\pi_C$. Further, we warm-up the model for a certain amount of intervals $s_w$ until we reach the situation where

$$\gamma := \sum_{x_k \in \Omega_X} \sum_{r_{in}^i \in \Omega_{R_{in}}} \sum_{c_j \in \Omega_C} |l_s(x_k, r_{in}^i, c_j) - l_{s-1}(x_k, r_{in}^i, c_j)| < \epsilon. \ [s > s_w, \epsilon > 0]$$

(2.3)

For $s > s_w$ we denote the simultaneous probability in (2.2) by $l(x_k, r_{in}^i, c_j)$. Now, because of the complexity of the model, it is not feasible to derive a general statement, telling us how long the warm-up period should be in order to assure that (2.3) is satisfied. There are to many parameters that have an influence on the behaviour of the model. Moreover, whenever we are trying to verify if (2.3) is feasible at all we already run into problems. This is because of the following reason. Recall that the rate with which the buffer content
changes is determined by the differences \((r_i^{\text{in}} - c_j)\Delta t\). Now, because we assumed the processes \(R^{\text{in}}\) and \(C\) to be two independent Markov chains the first step is to combine these two chains into an multidimensional Markov chain. If we define \(p_{i_1,i_2;j_1,j_2}\) by

\[
p_{i_1,i_2;j_1,j_2} := p_{i_1,i_2}^{R^{\text{in}}} \otimes p_{j_1,j_2}^{C},
\]

then figure 2.6 illustrates the multidimensional Markov chain for the \(2 \times 2\) case. Under the conditions as described in [5], the obtained multidimensional Markov chain is again ergodic, and hence has a unique limit distribution. Unfortunately, the Markov chain property gets destroyed by the special cases when we run into an empty or an full buffer, hence we cannot derive a statement about a limit distribution.

But, our experience with running the iterative computation on a computer is that the limiting behaviour as described in (2.3) does occur. The speed with which this limiting behaviour occurs depends on the choice of the parameters involved. In section 3.1, we investigate the influence of the parameters on the model in more detail and several examples are presented to support the derived statements.

![Multidimensional Markov chain](image)

**Figure 2.6:** Multidimensional Markov chain in the case of two input states and two capacity states.

**Initialization**

The second preparation step deals with the initialization of the model. So the question is how to initialize the probabilities in the first interval. From the initialization we can then work repeatedly on successive parts of the problem in an iterative way. We discuss two
possible initializations for the model.

For the first initialization we start with an empty buffer almost surely, i.e. \( \mathbb{P}(X_0 = 0) = 1 \). Due to the independence of the processes \( R_i^{\text{in}} \) and \( C \), the probability in (2.2) in the first interval is then given by
\[
l_0(x_k, r_i^{\text{in}}, c_j) = \begin{cases} 0 & \text{if } x_k \neq 0 \\ \pi_i^{R_i^{\text{in}}} \pi_j^{C} & \text{otherwise}
\end{cases}
\]

The second initialization assigns to each possible starting value for the buffer content the same probability, independent of \( Q \) and \( C \), i.e.,
\[
l_0(x_k, r_i^{\text{in}}, c_j) = \mathbb{P}(X_0 = x_k) \mathbb{P}(R_0^{\text{in}} = r_i^{\text{in}}) \mathbb{P}(C_0 = c_j)
= \frac{1}{K} \Delta x + 1 \pi_i^{R_i^{\text{in}}} \pi_j^{C}.
\]

### 2.2.2 Iteration step

Now that we initialized the model and described the stationarity assumptions, we are ready to discuss the iteration step. Hereto, let \( s \) \((s \geq 1)\) be an arbitrary interval. In order to compute (2.2) for interval \( s \), it suffices to know (2.2) for all possible values for the previous interval \( s - 1 \). In the following theorem, we describe an iterative computation of the probability in (2.2).

**Theorem 2.2.1** Let \( x_{k_1} \in \Omega_X \), \( r_i^{\text{in}} \in \Omega_{R_i^{\text{in}}} \) and \( c_{j_1} \in \Omega_C \), then
\[
l_s(x_{k_1}, r_i^{\text{in}}, c_{j_1}) = \sum_{x_{k_2} \in \Omega_X} \sum_{r_i^{\text{in}} \in \Omega_{R_i^{\text{in}}}} \sum_{c_{j_2} \in \Omega_C} l_{s-1}(x_{k_2}, r_i^{\text{in}}, c_{j_2}) p_{i_{11}^{\text{in}}} p_{j_{12}^{C}} \cdot I,
\]

where \( I \) is the indicator given by \( \mathbb{1}_E \), with
\[
E = \left\{ (r_{i_2}^{\text{in}} - c_{j_2}) \Delta t \right\},
\]

\[
\begin{cases} \text{“} \leq \text{”} & \text{if } x_{k_1} = 0 \\ \text{“} = \text{”} & \text{if } 0 < x_{k_1} < K \\ \text{“} \geq \text{”} & \text{if } x_{k_1} = K \\ \end{cases}
\]
\[
x_{k_1} - x_{k_2}
\]

**Proof.** First, denote the events \( A, B, C \) and \( D \) by
\[
A = \left\{ X_s = x_{k_1} \cup X_{s-1} = x_{k_2} ; R_i^{\text{in}} = r_i^{\text{in}} \cup R_{i_2}^{\text{in}} = r_{i_2}^{\text{in}} ; C_s = c_{j_1} \cup C_{s-1} = c_{j_2} \right\},
\]
\[
B = \left\{ X_s = x_{k_1} ; X_{s-1} = x_{k_2} ; R_i^{\text{in}} = r_i^{\text{in}} ; R_{i_2}^{\text{in}} = r_{i_2}^{\text{in}} ; C_s = c_{j_1} ; C_{s-1} = c_{j_2} \right\},
\]
\[
C = \left\{ X_{s-1} = x_{k_2} ; R_i^{\text{in}} = r_i^{\text{in}} ; R_{i_2}^{\text{in}} = r_{i_2}^{\text{in}} ; C_s = c_{j_1} ; C_{s-1} = c_{j_2} \right\},
\]
\[
\text{and } D = \left\{ X_{s-1} = x_{k_2} ; R_i^{\text{in}} = r_i^{\text{in}} ; C_s = c_{j_1} ; C_{s-1} = c_{j_2} \right\},
\]
respectively. Using this notation, (2.2) can be written as:

\[ l_s(x_{k_1}, r_{i_1}^\text{in}, c_{j_1}) = \mathbb{P}(A) = \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} \mathbb{P}(B) = \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} \mathbb{P}(C) \cdot I = \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} \mathbb{P}(D) \mathbb{P}(R_s^\text{in} = r_{i_1}^\text{in}|D) \cdot I = \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} \mathbb{P}(D) \cdot p_{i_1 i_2}^R \cdot I. \]

In the last step, we use that \( R^\text{in} \) is independent of \( C \), and that \( R^\text{in} \) is a Markov chain, i.e. that \( R_s^\text{in}|R_{s-1}^\text{in} \) is independent of \( R_m^\text{in} \) \((m \leq s - 2)\). Now, for the first probability in this triple sum, we can use exactly the same trick for the Markov chain \( C \) as we did for the Markov chain \( R^\text{in} \). This amounts to

\[ \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} \mathbb{P}(D) \cdot p_{i_1 i_2}^R \cdot I = \sum_{x_{k_2} \in \Omega_X} \sum_{r_{i_2}^\text{in} \in \Omega_{R^\text{in}}} \sum_{c_{j_2} \in \Omega_C} l_{s-1}(x_{k_2}, r_{i_2}^\text{in}, c_{j_2}) \cdot p_{i_1 i_2}^R \cdot p_{c_1 c_2}^C \cdot I. \]

\[ \square \]

2.3 Derivation of the BRD

In this section we discuss how to compute the BRD for the model. We first focus on the BRD at the input side, and after that, we use the obtained results to derive the BRD at the output side.

2.3.1 Derivation of the BRD at the input side

Because we focus on steady-state probabilities, and hence start the Markov chain \( R^\text{in} \) from its stationary distribution \( \pi_{R^\text{in}} \), the BRD at the input side is given by \( \pi_{R^\text{in}}^T \).

2.3.2 Derivation of the BRD at the output side

Next, we derive the BRD at the output side \((\pi_{R^\text{out}})\), which is defined in a similar way as the BRD at the input side. Hereto, recall from section 2.1.4 that the process \( R^\text{out} \) depends on the level of the buffer content and the rate with which the buffer content changes, and that the state space for the process \( R^\text{out} \) is given by \( \Omega_{R^\text{out}} = \{r_{1}^\text{out}, \ldots, r_{M^\text{out}}^\text{out}\} \). Now, in order to determine the BRD at the output side, we need to know how the elements in
\( \Omega_{\text{out}} \) are determined by the processes \( X, R^{\text{in}} \) and \( C \). Hereto, we take a closer look at how the process \( R^{\text{out}} \) is determined by the other processes.

For each interval \( s \), we have to consider three different cases. In the first case the buffer started empty and remains empty during the whole interval, so the output rate equals the input rate, i.e. \( R^{\text{out}}_s \equiv R^{\text{in}}_s \). The second case deals with the situation where the buffer started empty or non-empty and we end up with a non-empty buffer. Here the buffer is non-empty during the whole interval, hence the output rate equals the capacity, i.e. \( R^{\text{out}}_s \equiv C_s \). In the final case the buffer started non-empty and becomes empty before we reach the end of the interval. Here the output rate is a mixture of the input rate and the capacity. During the time the buffer is non-empty, \( R^{\text{out}}_s \) equals \( C_s \). So, if we denote by \( t^{s,e} \) the time that the buffer becomes empty in interval \( s \), then the output rate is given by

\[
R^{\text{out}}_s = C_s \cdot \frac{t^{s,e}}{\Delta t} + R^{\text{in}}_s \cdot \frac{(\Delta t - t^{s,e})}{\Delta t}.
\]

The two special cases in which we run into an empty or a full buffer are illustrated in figure 2.7. For the first case \( R^{\text{in}}_s - C_s < 0 \) and we reach an empty buffer in interval \( s \) at time \( s \Delta t - \frac{X_s}{R^{\text{in}}_s - C_s} \), and for the second case \( R^{\text{in}}_s - C_s > 0 \) and we reach an full buffer in interval \( s \) at time \( s \Delta t + \frac{K - X_s}{R^{\text{in}}_s - C_s} \).

Now, denote the events \( A, B \) and \( C \) by

\[
A := \left\{ X_s = x_{k1} = 0; R^{\text{in}}_s = r_{i1}^{\text{in}}; C_s = c_{j1}; r_{i1}^{\text{in}} = u; (r_{i1}^{\text{in}} - c_{j1}) \Delta t > x_{k1} \right\},
\]

\[
B := \left\{ X_s = x_{k2} > 0; R^{\text{in}}_s = r_{i2}^{\text{in}}; C_s = c_{j1}; \frac{x_{k2}}{\Delta t} + r_{i2}^{\text{in}} = u; (r_{i2}^{\text{in}} - c_{j2}) \Delta t < x_{k2} \right\},
\]

and

\[
C := \left\{ C_s = c_{j3} = u \right\},
\]

respectively. Then, it follows from the three cases just discussed that

\[
\pi^R_{\text{out}} > 0 \iff (1 \lor 2 \lor 3),
\]

where

\[
1 = \exists x_{k1} \in \Omega_X \exists r_{i1}^{\text{in}} \in \Omega_{R^{\text{in}}} \exists c_{j1} \in \Omega_C : A,
\]

\[
2 = \exists x_{k2} \in \Omega_X \exists r_{i2}^{\text{in}} \in \Omega_{R^{\text{in}}} \exists c_{j2} \in \Omega_C : B,
\]

and

\[
3 = \neg 1 \land \neg 2 \land \exists c_{j3} \in \Omega_C : C.
\]
2.4 Derivation of the ACF

Now that we discussed how to compute the BRD for the model, we next focus on a second bit rate statistic in which we are interested, namely, the ACF. Hereto, we first describe the general definition of the ACF. If you are familiar with the ACF, feel free to skip this part. After that, we show how to compute the ACF at the input side and the output side, respectively.

2.4.1 General definition of the ACF

We first discuss a term which is closely related to the ACF, namely the autocovariance function. Hereto, let \( Y = (Y_i)_{i \in \{1, \ldots, n\}} \) be an arbitrary sequence of discrete random variables with expectations \( \mu_i \) and variances \( \sigma_i^2 \). The covariance between \( Y_j \) and \( Y_k \), which is a
measure of their (linear) dependence, is denoted by $C_Y(j, k)$, and is defined by

$$C_Y(j, k) = \mathbb{E}[(Y_j - \mu_j)(Y_k - \mu_k)] = \mathbb{E}(Y_j Y_k) - \mu_j \mu_k.$$  

We now give two definitions that shed some light on the significance of the covariance $C_Y(j, k)$. If $C_Y(j, k) > 0$, then $Y_j$ and $Y_k$ are said to be positively correlated. In this case, $Y_j > \mu_j$ and $Y_k > \mu_k$ tend to occur together, and $Y_j < \mu_j$ and $Y_k < \mu_k$ also tend to occur together. Thus, for positively correlated random variables, if one is large, the other is likely to be large also. If $C_Y(j, k) < 0$, then $Y_j$ and $Y_k$ are said to be negatively correlated. In this case, $Y_j > \mu_j$ and $Y_k < \mu_k$ tend to occur together, and $Y_j < \mu_j$ and $Y_k > \mu_k$ also tend to occur together. Thus, for negatively correlated random variables, if one is large, the other is likely to be small. The difficulty with using $C_Y(j, k)$ as a measure of dependence between $Y_j$ and $Y_k$ is that it is not dimensionless, which makes it interpretation troublesome. As a result, we use the correlation $A_Y(j, k)$, defined by

$$A_Y(j, k) = \frac{C_Y(j, k)}{\sqrt{\sigma^2_j \sigma^2_k}}$$

as our primary measure of the (linear) dependence between $Y_j$ and $Y_k$. Since the denominator in the previous display is positive, it is clear that $A_Y(j, k)$ has the same sign as $C_Y(i, j)$. Furthermore, $-1 \leq A_Y(j, k) \leq 1$ for all $j$ and $k$, which follows from Schwarz's inequality. If $A_Y(j, k)$ is close to $+1$, then $Y_j$ and $Y_k$ are highly positively correlated. On the other hand, if $A_Y(j, k)$ is close to $-1$, then $Y_j$ and $Y_k$ are highly negatively correlated.

Further, the sequence $Y$ is said to be covariance-stationary if

$$\mu_i = \mu \quad \text{for } i = 1, 2, \ldots, n \text{ and } -\infty < \mu < \infty$$

$$\sigma^2_i = \sigma^2 \quad \text{for } i = 1, 2, \ldots, n \text{ and } \sigma^2 < \infty$$

and $C_Y(j, j+k)$ is independent of $j$ for $k = 1, 2, \ldots, n$. Thus, for a covariance-stationary process, the mean and variance are stationary over time (the common mean and variance are denoted by $\mu$ and $\sigma^2$, respectively), and the covariance between two observations $Y_j$ and $Y_{j+k}$ depends only on the separation $k$ (we call this the lag) and not on the actual time values $j$ and $j+k$. For a covariance-stationary process, we denote the covariance and correlation between $Y_j$ and $Y_{j+k}$ by $C_Y(k)$ and $A_Y(k)$, respectively (and we call them autocovariance and autocorrelation), where

$$A_Y(k) = \frac{C_Y(k)}{\sigma^2}.$$

### 2.4.2 Derivation of the ACF at the input side

Because we start the input Markov chain $R^{in}$ from its stationary distribution, this process is clearly covariance-stationary. This means that the ACF for the input rates at the $k$-th
lag \( A_{R_{in}}(k) \) is given by

\[
A_{R_{in}}(k) = \frac{E[R_{s+k}^{in}R_{s}^{in}] - \mu_{R_{in}}^2}{\sigma_{R_{in}}^2}, \quad [k \geq 0, \ s > s^w]
\]

where \( \mu_{R_{in}} \) and \( \sigma_{R_{in}}^2 \) are the expectation and the variation of the process \( R_{in} \), respectively. Note that, because the ACF is symmetric, we only consider nonnegative lags. Further, because the process \( R_{in} \) is an ergodic first-order Markov chain, the expectation in the preceding display can be rewritten as follows:

\[
E[R_{s+k}^{in}R_{s}^{in}] = E[E[R_{s+k}^{in}R_{0}^{in}|R_{s}^{in}]]
\]

\[
= E \left[ \sum_{r_{i1}^{in} \in \Omega_{R_{in}}} \sum_{r_{i2}^{in} \in \Omega_{R_{in}}} p_{i1i2}^{R_{in}(k)} E[R_{s-k}^{in} = r_{i1}^{in}|R_{0}^{in}] \right]
\]

\[
= \sum_{r_{i1}^{in} \in \Omega_{R_{in}}} \sum_{r_{i2}^{in} \in \Omega_{R_{in}}} p_{i1i2}^{R_{in}(k)} E[R_{s-k}^{in} = r_{i1}^{in}|R_{0}^{in}] \tag{2.4}
\]

### 2.4.3 Derivation of the ACF at the output side

Recall from the previous subsection that, in order to derive the expression for the steady-state ACF at the input side, we used the fact that \( R_{in} \) is an ergodic first-order Markov chain which is covariance-stationary. Unfortunately, we cannot use a similar approach for the process \( R_{out} \), because in general \( R_{out} \) is not a Markov chain and not covariance-stationary, as will be shown in the following example.

**Example** Suppose that \( \Omega_{R_{in}} = \{1, 4\} \), \( \Omega_C = \{3\} \), \( K = 3 \) and \( \Delta t = 1 \). Further, let all the transition probabilities be positive. We now show that

\[
0 \leq P(R_{out}^{3} = 2|R_{out}^{2} = 3) \neq P(R_{out}^{3} = 2|R_{out}^{2} = 3; R_{out}^{1} = 3) = 0,
\]

which proves that \( R_{out} \) is in general not a Markov chain. The first probability in the preceding display is indeed positive, because

\[
P(R_{out}^{3} = 2|R_{out}^{2} = 3) \geq P(R_{in}^{1} = 1; R_{in}^{2} = 4; R_{in}^{3} = 1; C_{1} = 3; C_{2} = 3; C_{3} = 3) > 0.
\]

For the second probability in (2.5), note that, only in cases for which \( R_{in}^{1} = R_{in}^{2} = 4 \) and \( C_{1} = C_{2} = 3 \), the output rate equals three in the first two intervals, and this leads to a buffer content of two. Now, whether we have an input rate of one or an input rate of four in the third interval, the output rate never equals two, hence the corresponding probability is indeed zero. Clearly, \( R_{out} \) is not covariance-stationary, which immediately can be seen from the fact that \( E[R_{in}^{1}] \neq E[R_{in}^{2}] \).
Because the process $R^{\text{out}}$ is not covariance-stationary, we cannot compute the ACF at a certain lag, but we can only compute the ACF between two specified points. Fortunately there is a way to overcome this problem, which is described in [16] on page 247. Here it is explained that, in order to draw inferences about an underlying stochastic process from a set of output data, one must sometimes make assumptions about the stochastic process that may not be strictly true in practice. Without such assumptions, however, statistical analysis of the output data may not be possible. An example of this is to assume that the process is covariance-stationary. In the following example we justify this assumption.

**Example** In this example we justify the covariance-stationary assumption for the process $R^{\text{out}}$. Hereto, we denote by $k^{\text{max}}$ the number of lags to be included. Now, we show that $\mu_{R^{\text{out}}}$ and $\sigma^2_{R^{\text{out}}}$ are stationary over time, and that

$$\Lambda_{s^w} := \sum_{k=1}^{k^{\text{max}}} \mathbb{E} \left[ (R_{s^w+k}^{\text{out}} - \mu_{R^{\text{out}}}) (R_{s^w+k+1}^{\text{out}} - \mu_{R^{\text{out}}}) \right] - \mathbb{E} \left[ (R_{s^w+1}^{\text{out}} - \mu_{R^{\text{out}}}) (R_{s^w+k+1}^{\text{out}} - \mu_{R^{\text{out}}}) \right]$$

tends to zero if we increase $s^w$. Of course, this means that we need to specify all the parameters involved. Here, we choose the parameters as follows: $\Omega_{\text{Rin}} = \{1,3\}$, $\Omega_{\text{C}} = \{2,4\}$, $P_{\text{Rin}}$ and $P_C$ are given by

$$P_{\text{Rin}} = \begin{pmatrix} 1/3 & 2/3 \\ 1/2 & 1/2 \end{pmatrix} \quad \text{and} \quad P_C = \begin{pmatrix} 1/6 & 3/6 \\ 1/6 & 5/6 \end{pmatrix},$$

respectively, $K = 5$, $\Delta t = 1$ and $k^{\text{max}} = 4$. From these values we easily obtain $\mu_{R^{\text{out}}}$, $\sigma^2_{R^{\text{out}}}$ and $\Lambda_{s^w}$ for several values of the warm-up period. The result is illustrated in figure 2.8. We see from the pictures that the process $R^{\text{out}}$ is covariance-stationary if we warm-up the model for at least ten intervals. The speed with which this limiting behaviour occurs depends on the chosen parameters. We remark that we looked at other input parameters as well, and these examples lead to similar conclusions.

Now that we justified the covariance-stationary assumption for the process $R^{\text{out}}$, we denote the common mean and variation of this process by $\mu_{R^{\text{out}}}$ and $\sigma^2_{R^{\text{out}}}$, respectively. The ACF of the process $R^{\text{out}}$ at the $k$-th lag ($A_{R^{\text{out}}}(k)$), can then be found from the equality

$$A_{R^{\text{out}}}(k) = \frac{\mathbb{E} \left[ R_{s^w}^{\text{out}} R_{s^w+k}^{\text{out}} \right] - \mu^2_{R^{\text{out}}}}{\sigma^2_{R^{\text{out}}}}. \quad [k \geq 0, \ s > s^w]$$

Note that, we cannot rewrite the expectation in the preceding display using a similar approach as in (2.4), because the process $R^{\text{out}}$ is in general not a first-order Markov chain.
Another approach to compute the expectation in (2.6), is to express this expectation in $E[R_{in} R_{out}]$ as follows:

$$E[R_{s}^{out} R_{s+k}^{out}] = E[(R_{s}^{in} - \Delta R_{s})(R_{s+k}^{in} - \Delta R_{s+k})]$$

$$= E[R_{s}^{in} \Delta R_{s+k}] - E[R_{s+k}^{in} \Delta R_{s}]$$

$$+ E[\Delta R_{s} \Delta R_{s+k}] + E[R_{s}^{in} R_{s+k}^{in}].$$

But this amounts to computing $E[\Delta R_{s} \Delta R_{s+k}]$, and the process $\Delta R_{s}$ is not a Markov chain and not covariance-stationary, so we run into the same problem.

Thus, we have to compute the expectation in (2.6) from its definition, which is given by

$$E[R_{s}^{out} R_{s+k}^{out}] = \sum_{u} \sum_{v} u \cdot v \cdot P(R_{s}^{out} = u; R_{s+k}^{out} = v).$$

In order to compute the probability of the event \{${R}_{s}^{out} = u; {R}_{s+k}^{out} = v$\} we have to keep track of the whole history of the processes $X$, $R^{in}$ and $C$ until interval $s + k$, and this requires much computational effort. Fortunately, we only have to consider the three cases as discussed in subsection 2.3.2 for the process $R^{out}$. If we denote the events $D$, $E$ and $F
Matching procedure for a discrete fluid flow model

by

\[ D := \{ X_s = x_{l_1} = 0; R_s^{in} = r_{m_1}^{in}; C_s = c_{p_1}; \frac{r_{m_1}^{in}}{\Delta t} > x_{l_1} \}, \]
\[ E := \{ X_s = x_{l_2} > 0; R_s^{in} = r_{m_2}^{in}; C_s = c_{p_2}; \frac{x_{l_2}}{\Delta t} + r_{m_2}^{in} = u; (r_{m_2}^{in} - c_{p_2})\Delta t < x_{l_2} \}, \]
and \[ F := \{ C_s = c_{p_3} = u \}, \]

respectively, then, considering only the three cases for \( R^{out} \) implies that

\[ \mathbb{P}(R_{s}^{out} = u; R_{s+k}^{out} = v) > 0 \Leftrightarrow (1 \lor 2 \lor 3) \land (4 \lor 5 \lor 6), \]

where 1, 2 and 3 are exactly of the same form as given in subsection 2.3.2, and 4, 5 and 6 are given by

\[ 4 = \exists x_{l_1} \in \Omega_X \exists r_{m_1}^{in} \in \Omega_{R^{in}} \exists c_{p_1} \in \Omega_C : D, \]
\[ 5 = \exists x_{l_2} \in \Omega_X \exists r_{m_2}^{in} \in \Omega_{R^{in}} \exists c_{p_2} \in \Omega_C : E, \]
and \[ 6 = \neg 4 \land \neg 5 \land \exists c_{p_3} \in \Omega_C : F, \]

respectively.

\section{2.5 Loss statistics}

We end this chapter by discussing certain loss statistics.

From the iterative solution of (2.2), we can now derive the loss statistics in which we are interested. In this work we focus on the the loss probability per interval \( p_{loss} \) and the average amount of loss per interval \( L \). If we denote by \( I \) the indicator

\[ I = \mathbb{1}_{\{(r_{i}^{\text{in}} - c_j)\Delta t > K - x_k \}}, \quad (2.7) \]

then \( p_{loss} \) and \( L \) are given by

\[ p_{loss} = \sum_{x_k \in \Omega_X} \sum_{r_{i}^{\text{in}} \in \Omega_{R^{in}}} \sum_{c_j \in \Omega_C} l_s(x_k, r_{i}^{\text{in}}, c_j) \cdot I, \quad [s > s^w] \]

and

\[ L = \sum_{x_k \in \Omega_X} \sum_{r_{i}^{\text{in}} \in \Omega_{R^{in}}} \sum_{c_j \in \Omega_C} \frac{\Delta t - \frac{K - x}{(r_{i}^{\text{in}} - c_j)\Delta t}}{\Delta t} l_s(x_k, r_{i}^{\text{in}}, c_j) \cdot (r_{i}^{\text{in}} - c_j) \cdot I, \quad [s > s^w] \]

respectively.
Chapter 3

Influence of the parameters on the model

In this chapter we discuss how the model reacts to changes in the parameters. In the first section we discuss how the parameters influence the speed with which the model reaches stationarity, and we present several examples to support the derived statements. Then, in the next section, we derive the cases in which the BRD has a special structure. Here we first focus on the BRD at the input side, and then we use the obtained results to derive similar statements for the BRD at the output side. In section three we describe a similar approach for the ACF. Again the input side and the output side are presented separately. After that, we combine the input side and the output side, and investigate the impact of the model on the BRD and the ACF in a more general way. This provides the substance of sections four and five. We end this chapter by describing the influence of the parameters on the buffer content. Section six deals with this subject.

In the following, we restrict ourselves to cases where both the input Markov chain and the capacity Markov chain have less than three states, because analysis becomes much easier with this restriction. Luckily, in the second part of this work, where we discuss a matching procedure for the model, we show that two-state Markov chains are preferred over three-state Markov chains, so this restriction does not cause any problems.

3.1 Influence of the parameters

In this section we examine how the parameters influence the speed with which the model converges. In the first subsection we present general statements, and the second subsection deals with several examples to support the derived statements.
3.1.1 General statements

Initialization
Recall from subsection 2.2.1 that we can initialize the model in two ways. For the first initialization we start with an empty buffer a.s., and for the second initialization we assign to each possible starting value for the buffer content the same probability. Unless stated otherwise we focus on the second initialization.

The warm-up period $s^w$
As discussed in subsection 2.2.1, we warm-up the process for a certain amount of intervals $s^w$, until we reach stationarity. In general, choosing a large warm-up period leads to a more accurate model, but it also takes a longer time to reach stationarity. An appropriate choice for $s^w$ depends on all the other parameters involved and is a certain trade off between speed and accuracy of the model.

Grid value for the buffer content $\Delta x$
Recall from section 2.1.4 that $\Delta x$ is determined by the state spaces $\Omega_{R_{in}}$ and $\Omega_C$, the averaging interval $\Delta t$ and the buffer size $K$. We should be careful with choosing these values in such a way that $\Delta x$ becomes too small, because a small $\Delta x$ value has a negative effect on the speed with which the model converges. This follows from the fact that, in order to compute (2.2) for an arbitrary interval $s$, we need to compute (2.2) for all possible values in the previous interval $s-1$, so the smaller $\Delta x$, the more computations are needed. We now discuss the influence of the averaging interval, the buffer size and the state spaces on $\Delta x$, respectively.

Influence of the averaging interval on $\Delta x$
It follows from the construction of $\Delta x$ in section 2.1.4 that, increasing (decreasing) $\Delta t$ causes $\Delta x$ to increase (decrease) as well.

Influence of the buffer size on $\Delta x$
Taking another look at the construction of $\Delta x$, we see that $\Delta x$ is only influenced by the buffer size $K$ in the second construction step. Here we first verify if $K$ is a multiple of $\Delta x$, and whenever this is not the case, we change to a smaller grid. In the following we mainly consider cases where $\frac{K}{\Delta x} \in \mathbb{N}$.

Influence of the state spaces $\Omega_{R_{in}}$ and $\Omega_C$ on $\Delta x$
Finally, the state spaces $\Omega_{R_{in}}$ and $\Omega_C$ influence $\Delta x$ in the first step of the construction procedure. Recall that, we first define $\Delta x$ by $\Delta x := |r_{i_1}^{\text{in}} - c_1| \cdot \Delta t$, and then run through all combinations $(i,j)$ where $i \in \{1, \ldots, M_{R_{in}}\}$ and $j \in \{1, \ldots, M_C\}$, and replace $\Delta x$ by $\gcd(|r_{i_1}^{\text{in}} - c_j| \cdot \Delta t, \Delta x)$ for each combination. Now, in order to see how the state spaces $\Omega_{R_{in}}$ and $\Omega_C$ influence $\Delta x$, we look at three examples. For the three examples the state spaces $\Omega_{R_{in}}$ are given by
\[
\Omega_{R_{in}} = \{1, 2\}, \quad \Omega_{R_{in}} = \{1.1, 2\}, \quad \text{and} \quad \Omega_{R_{in}} = \{1.1, 2.1\},
\]
respectively, and the state spaces $\Omega_C$ are given by

$$\Omega_C = \{1, 3\}, \quad \Omega_C = \{1, 3\} \quad \text{and} \quad \Omega_C = \{1.1, 3.1\},$$

respectively. From this, it can easily be verified that $\Delta x$ is given by 1, 0.1 and 1, respectively. Comparing the first and the second example, we see that, choosing the values in $\Omega_{R_i}$ and $\Omega_C$ such that $|r_i^{\text{in}} - c_j| \cdot \Delta t \not\in \mathbb{Z}^+$, leads to a smaller grid. The third example shows that it is important to use the differences $|r_i^{\text{in}} - c_j| \cdot \Delta t$ to compute $\Delta x$, and not focus on the elements in $\Omega_{R_i}$ and $\Omega_C$ separately. Because, although none of the elements in $\Omega_{R_i}$ and $\Omega_C$ are in $\mathbb{Z}^+$, the grid is given by 1.

The buffer size

We have just discussed the influence of the buffer size $K$ on the grid $\Delta x$, which in turn has influence on the speed with which the model converges. But, the value of $K$ also has a more direct influence on the speed of convergence. Clearly, the larger the buffer size $K$, the more elements in the state space $\Omega_X$, and the more computations are needed. So, if we increase the buffer size, then more intervals are needed before the model reaches stationarity.

3.1.2 Examples

In the previous subsection we discussed how the model reacts to changes in the parameters. In order to support the derived statements, we now present several numerical examples, which are summarized in table 3.1. Here $s^w$ corresponds to the least number of warm-up intervals needed before $Y < \epsilon = 0.001$ (see subsection 2.2.1), and $t_{\text{comp}}$ denotes the computation time in seconds.

For all the examples discussed, $P_{R_i}$ and $P_C$ are given by

$$P_{R_i} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad P_C = \begin{pmatrix} \frac{1}{3} & \frac{3}{4} \\ \frac{1}{6} & \frac{1}{6} \end{pmatrix},$$

respectively. The first example (row 1) is treated as the standard example from which the other examples are obtained by changing the input parameters. In order to distinguish the changed parameters from the unchanged ones, we made them bold.

First, in the second and the third example we change $r_i^{\text{in}}$ to 1.5 and 1.1, respectively. Note that, this causes $\Delta x$ to change to 0.5 and 0.1, respectively. Now, it is visible from the last two columns that these changes have only a small effect on the warm-up period and the computation time. Next, in the fourth and the fifth example, we change $\Delta t$ to 0.5 and 0.1, respectively, and clearly this has the same effect on $\Delta x$ as in the previous two examples. Here, we derive from the table that, especially in the fifth example where $\Delta t = 0.1$, the warm-up period and the computation time both increase. This is due to the fact that a small $\Delta x$ value can have a negative effect on the speed with which the model
converges. Then, in the next two examples, we let $K$ be given by 10 and 50, respectively. Here, the first case in which $K = 10$, has only a small effect on the warm-up period. But, if we change $K$ to 50, then both the warm-up period and the computation time increase. As is explained in the previous subsection, this is due to the fact that, if we increase the buffer size, then more computation are needed before the model reaches stationarity. Finally, in the last two examples, we change several parameters of the standard example simultaneously. First, in example eight, we let $\Delta t = 0.1$ and $r^{in}_1 = 1.1$, which leads to an extremely small grid value for the buffer content ($\Delta x = 0.01$). Clearly, this leads to much computational effort. It is visible from the last two columns of the table that the warm-up period increases to 47 intervals and the computation time increases to 219.4 seconds. Then, in the ninth example we again change $r^{in}_1$ to 1.1, but now we let $K = 50$. Here, we see that the combination of a small $\Delta x$ value and a large $K$ value lead to much computational effort and we draw similar conclusions as in the previous example.

Table 3.1: Influence of the parameters on the speed with which the model converges.

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>$K$</th>
<th>$\Omega_{R^{in}}$</th>
<th>$\Omega_C$</th>
<th>$\Delta x$</th>
<th>$s^{w}$</th>
<th>$t_{comp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>1.00</td>
<td>11</td>
<td>0.1</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.5,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.50</td>
<td>12</td>
<td>0.1</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.10</td>
<td>12</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.50</td>
<td>16</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.10</td>
<td>46</td>
<td>2.2</td>
</tr>
<tr>
<td>1.0</td>
<td>10</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>1.00</td>
<td>16</td>
<td>0.1</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>1.00</td>
<td>46</td>
<td>2.2</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.01</td>
<td>47</td>
<td>219.4</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.10</td>
<td>47</td>
<td>219.3</td>
</tr>
</tbody>
</table>

3.2 BRDs with a special structure

In this section we derive the cases in which the BRD has a special structure. First, we discuss the cases in which the BRD at the input side is either constant or symmetric. After that, we use the obtained results to derive similar statements for the BRD at the output side.

3.2.1 A constant BRD at the input side

Note that, a constant BRD implies that the variation equals zero, and hence, that the ACF is undefined. For this reason we discuss the cases for which the BRD at the input side is constant and discard these cases in further investigation. We only consider the cases for which the input Markov chain $R^{in}$ consist of two states, because cases for which $R^{in}$ consist of one state clearly correspond to an constant BRD at the input side.
Matching procedure for a discrete fluid flow model

Note that, the BRD at the input side is constant if either \( \pi_{R_{in}} \to [1, 0] \), or if \( \pi_{R_{in}} \to [0, 1] \). We first focus on cases for which \( \pi_{R_{in}} \to [1, 0] \). Hereto, we look at two special Markov chains, for which the transition matrices are given by

\[
\begin{pmatrix}
1 - \epsilon & \epsilon \\
p_{21} & p_{22}
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
p_{11} & p_{12} \\
1 - \epsilon & \epsilon
\end{pmatrix}, \quad [0 < \epsilon < 1]
\]

respectively. Now, clearly, if we let \( \epsilon \) tend to zero, then for both chains \( \pi_{R_{in}} \to [1, 0] \). Next, we remark that cases for which \( \pi_{R_{in}} \to [0, 1] \) are obtained from cases for which \( \pi_{R_{in}} \to [1, 0] \) by exchanging the columns in the two Markov chains in the preceding display. Further, note that, for both chains discussed, the input Markov chain consist of only one state if \( \epsilon = 0 \).

### 3.2.2 A symmetric BRD at the input side

Next, we discuss cases for which the BRD at the input side is symmetric. Hereto, we first give the definition of a symmetric Markov chain.

**Definition 3.2.1** We call an two-state Markov chain symmetric if its transition matrix is a symmetric matrix.

We now discuss a well-known property of two-state symmetric Markov chains.

**Theorem 3.2.1** Let \( P \) denote the transition matrix of a two-state symmetric Markov chain, and let \( p \) be the probability of remaining in the current state \((0 < p < 1)\), then

\[
\lim_{l \to \infty} P^{(l)} = \begin{pmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2}
\end{pmatrix}.
\]

**Proof.** This follows immediately from the fact that the rows of an \( l \)-step transition matrix \( P^{(l)} \) all become identical to the vector of steady-state probabilities as \( l \to \infty \). \( \Box \)

It follows that, if we choose the transition probabilities in such a way that the transition matrix is symmetric, then the BRD at the input side is symmetric.

### 3.2.3 A constant BRD at the output side

Next, we discuss the cases in which the BRD at the output side is constant and discard these cases in further investigation.

If \( r_{in}^{j} \geq c_{j} \forall i \in \{1, \ldots, M_{R_{in}}\} \forall j \in \{1, \ldots, M_{C}\} \), then the input exceeds the capacity in each interval. As a result, the buffer remains full and \( R_{out} \equiv C \). Whenever the Markov chain \( C \) consist of only one state, this means that the BRD at the output side is
constant.

**Remark:** If \( r^\text{in}_i \leq c_j \ \forall i \in \{1, \ldots, M^\text{in}\} \ \forall j \in \{1, \ldots, M^\text{out}\} \), then the capacity exceeds the input in each interval. Here, the buffer remains empty and \( R^\text{out} \equiv R^\text{in} \). Because we already discarded the cases for which the input process is constant, this does not lead to any contributions.

### 3.2.4 Symmetric BRD at the output side

(1) Consider again the cases in which the buffer remains full and the output process equals the capacity process. It follows from Theorem 3.2.1 in subsection 3.2.2 that, if the transition matrix of the Markov chain \( C \) is symmetric, then the BRD at the output side is symmetric.

(2) Again, we look at opposite cases, where the buffer remains empty. In these cases \( R^\text{out} \) equals \( R^\text{in} \), so, if the transition matrix of the input process corresponds to a symmetric Markov chain, then the BRD at the output side is symmetric.

**Remark:** Of course, there are more cases for which the BRD at the output side is symmetric. We do not discuss all these cases here, but give an example to support this remark. Let \( \Omega^\text{in} = \Omega^\text{C} = \{1, 2\} \), \( \Delta t = 1 \) and \( K = 2 \). Then \( \Omega^\text{out} = \{1, 2\} \), and we can choose \( P^\text{in} \) and \( P^\text{C} \) in such a way that the BRD is indeed symmetric.

### 3.3 ACFs with a special structure

Now that we derived the cases in which the BRD has a special structure, we focus on the second bit rate statistic in which we are interested, namely the ACF. Again we first discuss the ACF at the input side, and use the obtained results to derive similar statements for the ACF at the output side. Here we present the cases for which the ACF is of the form \((1, 0, 0, \ldots), (1, 1, 1, \ldots) \) or \((1, -1, 1, \ldots) \), respectively.

#### 3.3.1 ACFs with a special structure at the input side

\( A_{R^\text{in}} = (1, 0, 0, \ldots) \): Clearly, if we choose the transition probabilities in such a way that we start in stationarity, i.e. such that \( p_{ij} = \pi_j \ \forall i \), then there is no dependency between the variables in the sequence \( R^\text{in} \). This implies that the sequence \( R^\text{in} \) is uncorrelated, which means that the ACF equals 0 for all lags larger than zero.

\( A_{R^\text{in}} = (1, 1, 1, \ldots) \): Recall from section 2.4.1 that the ACF at lag \( k \) is a measure of dependence between \( R^\text{in}_s \) and \( R^\text{in}_{s+k} \), and is close to one if \( R^\text{in}_s \) and \( R^\text{in}_{s+k} \) are highly positively correlated. Now, we ensure that the sequence \( R^\text{in} \) is highly positively correlated if we choose the probabilities of switching states close to zero.
The ACF at the input side is \( A_{R_{\text{in}}} = (1, -1, 1, \ldots) \). Here \( R_{s_{\text{in}}}^{\text{in}} \) and \( R_{s_{\text{in}}+k}^{\text{in}} \) are highly negatively correlated for \( k \) odd, and highly positively correlated for \( k \) even. Opposite from the previous case, this means that the probabilities of switching states should be close to 1.

### 3.3.2 ACFs with a special structure at the output side

1. If the buffer remains empty, then \( R_{\text{out}} \equiv R_{\text{in}}^{\text{in}} \), and thus \( A_{R_{\text{out}}} \equiv A_{R_{\text{in}}} \). Hence, in this case, all the statements derived for the ACF at the input side also apply for the ACF at the output side.

2. If the buffer remains full, then \( R_{\text{out}} \equiv C \), and hence \( A_{R_{\text{out}}} \equiv A_{C} \). Now, note that, when we discussed the ACF for the input process, we derived statements for arbitrary Markov chains. Hence, if \( R_{\text{out}} \equiv C \), then all the derived statements can again be applied for the ACF at the output side.

### 3.4 Combination of the BRD at the input side and at the output side

Now that we discussed the BRD and the ACF at the input side and the output side, we combine the input side and the output side in order to investigate these two bit rate statistics in a more general way. In this section we derive general statements for the BRD by looking at so called histogram difference plots (HDPs). These HDPs help us to visualize the changes perceived by traffic on its way through a network.

### 3.4.1 Shared and shaping bottleneck

As discussed in [15], we obtain the so-called throughput HDP by subtracting the corresponding input histogram values from the output histogram values. From these plots, it can be seen whether the network acts as a shared or shaping bottleneck, i.e. increases or decreases the burstiness of a packet stream.

Figure 3.1 refers to a stream whose initially constant throughput is changed in a shared bottleneck. As soon as the demand for resources of all streams exceeds the capacity, queuing occurs and the throughput decreases. As soon as the demand falls below the capacity, the queue is relaxed, which implies higher throughput at the output. Altogether, the variability of the traffic grows. The resulting difference plot has the shape of an “M” with negative values close to the original speed and positive values at both lower and higher speeds.
Figure 3.1: Anticipated time plot, throughput histograms at input and output and HDP (from left to right) in case of a shared bottleneck.

Figure 3.2 illustrates the effect of a shaping bottleneck on a stream. In this case, the throughput variations are smoothed, i.e. the variability of the traffic decreases. The difference plot has now the shape of a “W” with positive values close to the shaper’s throughput and negative values at lower and higher speeds.

Figure 3.2: Anticipated time plot, throughput histograms at input and output and HDP (from left to right) in case of a shaping bottleneck.

### 3.4.2 Examples

In the second part of this work we describe a matching procedure involving the BRD at both the input side and the output side. For this reason, it is extremely important that the model is capable of reflecting the properties seen from the measurements. Hereto, we discuss two examples, one for which an shared bottleneck occurs and one for which an shaping bottleneck occurs. The shared and shaping bottleneck are typical behaviours seen from the measurements as described in [7].

In these two examples, the HDPs are characterized by the width $b$, the peak-to-peak value $h$ and the grade of deviation $g$. For $r_i \in \Omega_{R_{in}} \cup \Omega_{R_{out}}$ these values are defined
Matching procedure for a discrete fluid flow model

by

\[ b := \max \left\{ r_i : \pi_i^{\text{Rout}} - \pi_i^{\text{Rin}} \neq 0 \right\} - \min \left\{ r_i : \pi_i^{\text{Rout}} - \pi_i^{\text{Rin}} \neq 0 \right\} , \]

\[ h := \max \left\{ \pi_i^{\text{Rout}} - \pi_i^{\text{Rin}} \right\} + \left| \min \left\{ \pi_i^{\text{Rout}} - \pi_i^{\text{Rin}} \right\} \right| \]

and

\[ g := \sum_{r_i} \left| \frac{\pi_i^{\text{Rout}} - \pi_i^{\text{Rin}}}{2} \right| , \]

respectively.

**Example** In the first example we choose the parameters of the model in such a way that the throughput HDP corresponds to an shared bottleneck. Hereto, let \( P_{\text{Rin}} \) and \( P_C \) be of the form

\[ P_{\text{Rin}} = \begin{pmatrix} \frac{6}{10} & \frac{4}{10} \\ \frac{1}{10} & \frac{9}{10} \end{pmatrix} \quad \text{and} \quad P_C = \begin{pmatrix} \frac{7}{10} & \frac{3}{10} \\ \frac{3}{10} & \frac{7}{10} \end{pmatrix} , \]

respectively, and let \( \Omega_{\text{Rin}} = \{9,10\} \), \( \Omega_C = \{9,11\} \), \( K = 5 \) and \( \Delta t = 1 \). The resulting BRDs at the input side and at the output side are illustrated in figure 3.3 (left). Now, substraction of the input histogram values from the output histogram values leads to the HDP shown in figure 3.3 (right). Here the “M”-shape appears, which is typical for HDPs corresponding to an shared bottleneck.

Figure 3.3: BRDs at the input side and at the output side corresponding to an shared bottleneck (left) with corresponding throughput HDP (right).
Example The second example deals with an throughput HDP corresponding to an shaping bottleneck. Here we choose the parameters of the model in the following way: $P_{R_{in}}$ and $P_C$ are given by

\[ P_{R_{in}} = \begin{pmatrix} \frac{6}{10} & \frac{4}{10} \\ \frac{4}{10} & \frac{6}{10} \end{pmatrix} \quad \text{and} \quad P_C = \begin{pmatrix} \frac{1}{10} & \frac{9}{10} \\ \frac{9}{10} & \frac{1}{10} \end{pmatrix}, \]

respectively, $\Omega_{R_{in}} = \{9, 11\}$, $\Omega_C = \{9, 10\}$, $K = 5$ and $\Delta t = 1$. Figure 3.4 (left) illustrates the BRDs at both the input side and the output side. The resulting HDP, which is shown in figure 3.4 (right), has the shape of an “W” and hence corresponds to a shaping bottleneck.

![BRD at the input side](image1)

![BRD at the output side](image2)

![Histogram difference plot](image3)

Figure 3.4: BRDs at the input side and at the output side corresponding to a shaping bottleneck (left) with corresponding throughput HDP (right).

We conclude from the two examples that the model is perfectly capable of reflecting the properties seen from BRDs of the measurements.

3.5 Combination of the ACF at the input side and at the output side

Similarly, as done in the previous section for the BRD, we now derive general statements for the ACF. In the first subsection, we discuss the ACF in case of a shared and a shaping
bottleneck. The second subsection continues the numerical example started in subsection 3.4.2.

### 3.5.1 ACF in case of a shared and a shaping bottleneck

Here, we discuss the influence of a shared and a shaping bottleneck on the ACF. So, we again focus on the two situations where the model either increases or decreases the variation.

In the case of a shared bottleneck, the variability increases from input to output. This means that there is less dependency between the random variables in the sequence $R^{\text{out}}$ than in the sequence $R^{\text{in}}$. Hence, we expect the ACF to decrease from input to output.

In the case of a shaping bottleneck the variability decreases. Here the dependency grows from input to output and we expect the ACF to increase.

### 3.5.2 Examples

In the previous section we concluded that the model reflects the properties seen from BRDs of the measurements and remarked that this is important for the matching procedure. The second bit rate statistic that is dealt with in the matching procedure is the ACF. For this reason, we continue the two numerical examples started in subsection 3.4.2.

**Example** In example 3.4.2 we chose the parameters in such a way that the HDP corresponded to an shared bottleneck. Hence, as pointed out in the previous subsection, we expect the ACF to decrease from input to output. The ACFs at the input side and at the output side corresponding to the chosen parameters in example 3.4.2 are illustrated in figure 3.5. It is clearly visible from the picture that the ACF indeed decreases from input to output, and hence we conclude that the model changes the ACF as expected.

**Example** In the second example in subsection 3.4.2 we found parameters leading to an shaping bottleneck. Opposite from the previous example we expect the ACF to increase from input to output. Figure 3.6 illustrates the ACFs at both the input side and the output side and agrees with our expectation.

 Altogether we conclude that the model is perfectly capable of reflecting the properties seen from the measurements. This will be extremely important for the second part of this work where a matching procedure is presented involving the BRD and the ACF for the model.
3.6 Buffer content

In this section we focus on the behaviour of the buffer content. Hereto, we notate the BRD of the buffer content by \( \pi_X := (\pi^X_1, \ldots, \pi^X_M) \), where

\[
\pi^X_k := \mathbb{P}(X_s = x_k) \cdot [x_k \in \Omega_X, s > s^w]
\]

This section first considers the cases in which the buffer content is constant and symmetric, respectively. After that, we end this section with an interesting property of the buffer content.

3.6.1 A constant buffer content

(1) Recall from subsection 3.2.3 that cases for which \( r_i^{\text{in}} \geq c_j \ \forall i \in \{1, \ldots, M_{R^n}\} \ \forall j \in \{1, \ldots, M_C\} \) lead to a full buffer. This means that \( \pi^X_M = 1 \) a.s., hence the buffer content is constant.

(2) Opposite cases for which \( r_i^{\text{in}} \leq c_j \ \forall i \in \{1, \ldots, M_{R^n}\} \ \forall j \in \{1, \ldots, M_C\} \) lead to an empty buffer. Here, \( \pi^X_1 = 1 \) a.s., so the buffer content is again constant.

3.6.2 A symmetric buffer content

(1) If the transition matrix of the Markov chain \( R^{\text{in}} \) is symmetric and the Markov chain \( C \) consist of only one state, then the buffer content is symmetric iff \( |r_1^{\text{in}} - c_1| = |r_2^{\text{in}} - c_1| \). This can easily be verified with induction.
(2) If the Markov chain consist of two states and the transition matrix of both Markov chains $R^\text{in}$ and $C$ are symmetric, then the buffer content is symmetric iff $|r_1^\text{in} - c_1| = |r_2^\text{in} - c_2| \land |r_1^\text{in} - c_2| = |r_2^\text{in} - c_1|$. Again, this can be verified with induction.

### 3.6.3 General feature of the buffer content

In this section we discuss an striking property of the buffer content. We exchange the input Markov chain with the capacity Markov chain, i.e. exchange the source and the server, and compare the two buffer contents. The main result is summarized in the following

**Theorem 3.6.1** Let $\tilde{X}$ denote the process obtained from the process $X$ by exchanging the input Markov chain with the capacity Markov chain, and denote by $\pi^X$ its corresponding distribution. Then

$$
\pi^X_k = \pi^\tilde{X}_k - K.
$$

(3.2)
Proof. For simplicity of notation, denote the input Markov chain and the capacity Markov chain by $L^1$ and $L^2$, respectively. Further, let

$$A := \left\{ X_s = x_{k_1}; \bigcup_{l_1 \in \Omega_{L_1}} L^1_s = l_{i_1}; \bigcup_{l_j \in \Omega_{L_2}} L^2_s = l_{j_1} \right\}$$

$$B := \left\{ X_s = x_{k_1}; L^1_s = l_{i_1}; L^2_s = l_{j_1} \right\}$$

$$C := \left\{ X_{s-1} = x_{k_2}; L^1_{s-1} = l_{i_2}; L^2_s = l_{j_2} \right\}$$

$$D := \left\{ \dot{X}_{s-1} = K + x_{k_2}; L^1_{s-1} = l_{j_2}; L^2_{s-1} = l_{i_2} \right\}$$

$$E := \left\{ \dot{X}_s = K + x_{k_1}; L^2_s = l_{j_1}; L^1_s = l_{i_1} \right\}$$

and

$$F := \left\{ \dot{X}_s = K + x_{k_1}; \bigcup_{l_j \in \Omega_{L_1}} L^2_s = l_{j_1}; \bigcup_{l_i \in \Omega_{L_2}} L^1_s = l_{i_1} \right\}$$

and let the indicators $I^1$ and $I^2$ be denoted by

$$I^1 := \{(l_{i_2} - l_{j_2}) \Delta t \left\{ \begin{array}{ll} "\leq" & \text{if } x_{k_1} = 0 \\ "=" & \text{if } 0 < x_{k_1} < K \\ "\geq" & \text{if } x_{k_1} = K \end{array} \right\} x_{k_1} - x_{k_2} \}$$

and

$$I^2 := \{(l_{j_2} - l_{i_2}) \Delta t \left\{ \begin{array}{ll} "\leq" & \text{if } x_{k_1} = 0 \\ "=" & \text{if } 0 < x_{k_1} < K \\ "\geq" & \text{if } x_{k_1} = K \end{array} \right\} x_{k_1} - (K + x_{k_2}) \}.$$

Then, the probability on the right hand side of (3.2) is given by

$$\pi_{k_1}^X = \mathbb{P}(A)$$

$$= \sum_{l_1 \in \Omega_{L_1}} \sum_{l_2 \in \Omega_{L_2}} \mathbb{P}(B)$$

$$= \sum_{l_1 \in \Omega_{L_1}} \sum_{l_2 \in \Omega_{L_2}} \sum_{x_{k_2} \in \Omega_X} \sum_{l_{i_1} \in \Omega_{L_1}} \sum_{l_{j_2} \in \Omega_{L_2}} \mathbb{P}(C) \cdot p_{L^1}^{l_{i_1}} \cdot p_{L^2}^{l_{j_2}} \cdot I^1$$

$$= \sum_{l_1 \in \Omega_{L_1}} \sum_{l_2 \in \Omega_{L_2}} \sum_{x_{k_2} \in \Omega_X} \sum_{l_{i_1} \in \Omega_{L_1}} \sum_{l_{j_2} \in \Omega_{L_2}} \mathbb{P}(C) \cdot p_{L^1}^{l_{i_1}} \cdot p_{L^2}^{l_{j_2}} \cdot I^1$$

$$= \sum_{l_1 \in \Omega_{L_1}} \sum_{l_2 \in \Omega_{L_2}} \sum_{x_{k_2} \in \Omega_X} \sum_{l_{i_1} \in \Omega_{L_1}} \sum_{l_{j_2} \in \Omega_{L_2}} \sum_{l_{i_2} \in \Omega_{L_2}} \mathbb{P}(D) \cdot p_{L^1}^{l_{i_1}} \cdot p_{L^2}^{l_{j_2}} \cdot I^1$$

$$= \sum_{l_1 \in \Omega_{L_1}} \sum_{l_2 \in \Omega_{L_2}} \sum_{l_{i_1} \in \Omega_{L_1}} \mathbb{P}(F)$$

$$= \pi_{k_1}^X - K.$$


In order to clarify the result of Theorem 3.6.1, we discuss an numerical example in which we exchange the input Markov chain with the capacity Markov chain and plot the two buffer contents.

**Example** Let the transition matrices $P_{R_{in}}$ and $P_{C}$ be of the form

$$P_{R_{in}} = \begin{pmatrix} \frac{1}{10} & \frac{2}{10} \\ \frac{3}{5} & \frac{2}{5} \end{pmatrix} \quad \text{and} \quad P_{C} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix},$$

respectively, and let $\Omega_{R_{in}} = \{9, 10\}, \Omega_{C} = \{8, 11\}, K = 50$ and $\Delta t = 1$. The buffer content corresponding to these parameters is illustrated in figure 3.7 (left). Now, we exchange the input Markov chain with the capacity Markov chain, i.e. we exchange $\Omega_{R_{in}}$ with $\Omega_{C}$ and $P_{R_{in}}$ with $P_{C}$. The resulting buffer content is illustrated in figure 3.7 (right). Note that the peaks at $x_k = 2$ and $x_k = 3$ of the left picture, caused by cases in which we run into an empty buffer, are shifted to $x_k = 48$ and $x_k = 47$ in the right picture, corresponding to cases in which we run into an full buffer. The right picture typically corresponds to a situation of overload, which means that the average input rate exceeds the average capacity, i.e. $\mu_{R_{in}} > \mu_{C}$. In the sequel we mainly focus on cases where $\mu_{R_{in}} \leq \mu_{C}$. 

Figure 3.7: Buffer content before and after exchanging source and server.
Chapter 4

Simulation

In order to check the validity of the statements derived in the previous two chapters, we now discuss a corresponding simulation. In the first section we review some basic statistics particularly relevant to simulation. Then, in the next section, we start the simulation and obtain estimates for the simultaneous probability of $X$, $R^\text{in}$ and $C$. Section three describes estimates for the BRD and the ACF at both the input side and the output side. The last section focuses on estimates for the buffer content.

4.1 Introduction

In this introductory section we start with some preparations. We first describe the type of simulation that is dealt with and describe how to insure independence across runs. Then, in the next subsection, we discuss confidence intervals based on both the standard normal distribution and the t-distribution.

4.1.1 Preparation

In a simulation we use a computer to evaluate a model numerically, and data are gathered in order to estimate the desired true characteristics of the model. Now, because the state variables change instantaneously at separate points in time, the type of simulation that we focus on is called discrete-event simulation ([16] page 6). Although discrete-event simulation could conceptually be done by hand calculations, the amount of data that must be stored and manipulated dictates that discrete-event simulation be done on a digital computer.

Now, each run of a stochastic simulation model produces only estimates of a model’s true characteristics for a particular set of input parameters. Thus, several independent runs of the model are required for each set of input parameters to be studied. In order to describe the random nature of simulation output more precisely, we follow the strategy as explained in [16] on page 496. Here, $Y = \left\{Y_t\right\}_{t=1}^n$ is an output stochastic process, and
\( y_{ij} \) denotes the realization of the random variable \( Y_i \) in the \( j \)-th independent replication. Now, if each replication uses the same initial conditions, then the observations \( \{y_{ij}\}_{j=1}^n \) of the random variable \( Y_i \) are i.i.d. This is the key to relative simple output-data-analysis methods described in the remaining part of this chapter.

### 4.1.2 Confidence intervals

Let \( Y = \{Y_i\}_{i=1}^n \) be i.i.d. random variables with finite mean \( \mu_Y \) and finite variance \( \sigma_Y^2 \). It is known that, for \( n \) sufficiently large, an approximate 100(1 − \( \alpha \)) percent confidence interval for \( \mu_Y \) is given by

\[
\hat{\mu}_Y \pm z_{1-\alpha/2} \sqrt{\frac{\hat{\sigma}^2_Y}{n}},
\]

where \( \hat{\mu}_Y \) and \( \hat{\sigma}^2_Y \) are given by

\[
\hat{\mu}_Y = \frac{1}{n} \sum_{i=1}^n Y_i \quad \text{and} \quad \hat{\sigma}^2_Y = \frac{1}{n-1} \sum_{i=1}^n [Y_i - \hat{\mu}_Y]^2,
\]

respectively, and \( z_{1-\alpha/2} \) is the upper 1 − \( \alpha/2 \) critical point for a standard normal random variable. The correct interpretation to give to this confidence interval is as follows. If one constructs a very large number of independent 100(1 − \( \alpha \)) percent confidence intervals, each based on \( n \) observations, where \( n \) is sufficiently large, the proportion of these confidence intervals that contain \( \mu_Y \) should be 1 − \( \alpha \). We call this proportion the coverage for the confidence interval.

The difficulty in using (4.1) to construct a confidence interval for \( \mu_Y \) is in knowing what “\( n \) sufficiently large” means. If \( n \) is chosen too small, the actual coverage of a desired 100(1 − \( \alpha \)) percent confidence interval will generally be less than 1 − \( \alpha \). This is why the confidence interval given by (4.1) is stated to be only approximate.

In light of the above discussion, we now develop an alternative confidence interval expression. If the \( Y_i \)'s are normal random variables, the random variable \( t_n = [\hat{\mu}_Y - \mu_Y] \sqrt{\frac{n}{\hat{\sigma}^2_Y}} \) has a \( t \) distribution with \( n - 1 \) degrees of freedom, and an exact 100(1 − \( \alpha \)) percent confidence interval for \( \mu_Y \) is given by

\[
\hat{\mu}_Y \pm t_{n-1,1-\alpha/2} \sqrt{\frac{\hat{\sigma}^2_Y}{n}},
\]

where \( t_{n-1,1-\alpha/2} \) is the upper 1 − \( \alpha/2 \) critical point for the \( t \)-distribution with \( n - 1 \) degrees of freedom.

Plots of the density functions for the \( t \) distribution with four degrees of freedom and for the standard normal distribution are given in figure 4.1. Note that the \( t \)-distribution is less peaked and has longer tails than the normal distribution, so, for any finite \( n \), \( t_{n-1,1-\alpha/2} > z_{1-\alpha/2} \). In the remaining part of this work, we focus on both of the just mentioned confidence intervals.
4.2 Simulation of the simultaneous probability of $X$, $R_{\text{in}}$ and $C$

In this section we start the simulation and derive estimates for the simultaneous probability of $X$, $R_{\text{in}}$ and $C$. In the first subsection we discuss the design of a single simulation run. After that, we choose the simulation run length and the number of independent model replications, and introduce the necessary notation. The last subsection deals with several numerical examples in which we obtain accurate final estimates.

4.2.1 Single simulation run

We start again by initializing the random variables in the first interval. Hereto, we generate random variables in $\Omega_{R_{\text{in}}}$ and $\Omega_C$ given the distribution vectors $\pi_{R_{\text{in}}}$ and $\pi_C$. Further, we pick a random starting level for the buffer content, where each level is picked with the same probability $\frac{1}{K+1}$. After the initialization of the model, we obtain the values for the two Markov chains $R_{\text{in}}$ and $C$ in an arbitrary interval, from the values in the previous interval and the corresponding transition probabilities. The buffer content $X$ can again be found from equation (2.1). Figure 4.2 illustrates a possible outcome of one simulation run. Here the transition matrices $P_{R_{\text{in}}}$ and $P_C$ are of the form

$$P_{R_{\text{in}}} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad \text{and} \quad P_C = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix},$$

and $\Omega_{R_{\text{in}}} = \{1, 7\}$, $\Omega_C = \{3, 5\}$, $K = 2$ and $\Delta t = 1$. 

Figure 4.1: Density functions for the t-distribution with four degrees of freedom and for the standard normal distribution.
Figure 4.2: Possible simulation outcome (top). More detailed picture of the behaviour of the buffer (bottom).
4.2.2 Notation and preparation

Suppose we make \( \eta \) independent replications of a terminating simulation, where each replication is terminated by the event \( T \) and is begun with the ‘same’ initial conditions. Then, we recall from section 4.1.1 that the replications are independent of each other. Now, we notate the sequences obtained for \( X, R_{\text{in}}, C \) and \( R_{\text{out}} \) in the \( m \)-th replication by \( X^{(m)}, R_{\text{in}}^{(m)}, C^{(m)} \) and \( R_{\text{out}}^{(m)} \), respectively. The estimates of the simultaneous probability of \( X, R_{\text{in}} \) and \( C \) in the \( m \)-th replication \( ^{(m)}(x_k, c_j, r_i^{\text{in}}) \) are then given by

\[
\hat{\ell}^{(m)}(x_k, c_j, r_i^{\text{in}}) = \sum_{p=1}^{n} \frac{1\{X_s^{(m)} = x_k, R_{\text{in}}^{(m)} = r_i^{\text{in}}, C_{s+p} = c_j\}}{n}, \quad [x_k \in \Omega_X, r_i^{\text{in}} \in \Omega_{R_{\text{in}}}, c_j \in \Omega_C, s > s^w]
\]

and the final estimate, which is notated by \( \hat{\ell}(x_k, c_j, r_i^{\text{in}}) \), can be found from the equation

\[
\hat{\ell}(x_k, c_j, r_i^{\text{in}}) = \sum_{m=1}^{\eta} \frac{\hat{\ell}^{(m)}(x_k, c_j, r_i^{\text{in}})}{\eta}, \quad [x_k \in \Omega_X, r_i^{\text{in}} \in \Omega_{R_{\text{in}}}, c_j \in \Omega_C]
\]

Now, in order to measure the accuracy of the estimates, we define \( \Theta \) by

\[
\Theta := \sum_{x_k \in \Omega_X} \sum_{r_i^{\text{in}} \in \Omega_{R_{\text{in}}}} \sum_{c_j \in \Omega_C} |\ell(x_k, r_i^{\text{in}}, c_j) - \hat{\ell}(x_k, c_j, r_i^{\text{in}})|. \quad [x_k \in \Omega_X, r_i^{\text{in}} \in \Omega_{R_{\text{in}}}, c_j \in \Omega_C]
\]

Clearly, the closer \( \Theta \) is to zero the more accurate the estimates. The confidence intervals for \( \Theta \) based on the standard normal distribution and \( t \)-distribution are denoted by \( C_{\Theta,n} \) and \( C_{\Theta,t} \), respectively.

4.2.3 Results

Now that we introduced the necessary notation, we discuss several numerical examples and examine the accuracy of the obtained estimates. Hereto, we first need to specify the length of the sequences involved \( (n) \), the number of simulations \( (\eta) \) and the critical points for the confidence intervals based on the standard normal distribution and the \( t \)-distribution \( (\alpha) \). For all the examples discussed here, we let \( n = 1000, \eta = 10 \) and \( \alpha = 0.05 \). Depending on the parameters of the model we could choose larger values for \( n \) and \( \eta \), but the values \( n = 1000 \) and \( \eta = 10 \) already lead to extremely accurate estimates for all the examples discussed in this subsection.

The results are listed in table 4.1. Now, note that, the chosen parameters are exactly the same as the ones discussed in subsection 3.1.2, where we examined the speed of convergence of the model and obtained the required warm-up periods. In order to obtain stationarity results we used an warm-up period of fifty intervals for the examples obtained in table 4.1. Now, from the table it is clearly visible from the \( \Theta \)-values that the simulation results agree closely with the exact solutions. Furthermore, we derive from the last column that the computation time is extremely short for most of the examples.
discussed. But, in cases where we change several parameters of the standard example simultaneously, the computation time increases rapidly. The confidence intervals $C_{\Theta,n}$ and $C_{\Theta,t}$ are nearly the same and show that we obtained accurate statistical results.

Table 4.1: Accuracy of the estimates obtained for the simultaneous probability of $X$, $R^\text{in}$ and $C$ (left) with corresponding confidence intervals (right).

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>$K$</th>
<th>$\Omega_{R^\text{in}}$</th>
<th>$\Omega_C$</th>
<th>$\Theta$</th>
<th>$t_{\text{comp}}$</th>
<th>$C_{\Theta,n}$</th>
<th>$C_{\Theta,t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>5</td>
<td>${1.0,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.064</td>
<td>0.1</td>
<td>[0.06,0.07]</td>
<td>[0.06,0.07]</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>${1.5,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.070</td>
<td>0.2</td>
<td>[0.06,0.08]</td>
<td>[0.06,0.08]</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>${1.0,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.068</td>
<td>0.2</td>
<td>[0.06,0.08]</td>
<td>[0.06,0.08]</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>${1.0,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.055</td>
<td>2.5</td>
<td>[0.04,0.07]</td>
<td>[0.04,0.07]</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>${1.0,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.060</td>
<td>2.6</td>
<td>[0.05,0.07]</td>
<td>[0.05,0.07]</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>${1.1,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.082</td>
<td>237.5</td>
<td>[0.05,0.07]</td>
<td>[0.05,0.07]</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>${1.1,3.0}$</td>
<td>${2.0,4.0}$</td>
<td>0.071</td>
<td>235.1</td>
<td>[0.05,0.09]</td>
<td>[0.06,0.09]</td>
</tr>
</tbody>
</table>

4.3 Simulation of the BRD and the ACF at the input side and at the output side

Next, we obtain estimates for the BRD and the ACF. Again, we first introduce the necessary notation, and after that, we discuss several numerical examples to support the derived statements.

4.3.1 Notation and preparation

In order to simulate the BRD and the ACF at both the input side and the output side, we first need to introduce some notation. We start by introducing notation for these two bit rate statistics at the input side of the model.

The estimates of the BRD at the input side in the $m$-th replication, which are denoted by $\tilde{\pi}_R^{(m)} = (\pi_1^{R^\text{in}(m)}, \ldots, \pi_M^{R^\text{in}(m)})$, can be found from the equation

$$\pi_i^{R^\text{in}(m)} = \frac{\sum_{p=1}^{n} 1_{\{R_i^{\text{in}(m)} = p\}}}{n}, \quad s > s^w, r_i^{\text{in}} \in \Omega_{R^\text{in}}$$
Matching procedure for a discrete fluid flow model

Further, we notate the estimates of the BRD at the input side by \( \hat{\pi}_{R_{in}} \), and these are obtained from the preceding display as follows:

\[
\hat{\pi}_{R_{in}}^{(m)} = \frac{\sum_{i=1}^{\eta} \hat{\pi}_{i}^{R_{in},(m)}}{\eta}.
\]

Next, for the estimates of the ACF at the input side in the \( m \)-th replication \( \hat{A}_{R_{in}}^{(m)} \) we first estimate the expectation and the variance in the \( m \)-th replication from

\[
\hat{\mu}_{R_{in}}^{(m)} = \frac{\sum_{i=1}^{n} R_{in,i}^{(m)}}{n} \quad \text{and} \quad \hat{\sigma}^{2(m)}_{R_{in}} = \frac{\sum_{i=1}^{n} \left[ R_{in,i}^{(m)} - \hat{\mu}_{R_{in}}^{(m)} \right]^2}{n-k},
\]

respectively, and then we estimate the covariance function in the \( m \)-th replication from

\[
\hat{C}_{R_{in}}^{(m)}(k) = \frac{\sum_{p=1}^{n} \left[ R_{in,i+p}^{(m)} - \hat{\mu}_{R_{in}}^{(m)} \right] \left[ R_{in,i+p+k}^{(m)} - \hat{\mu}_{R_{in}}^{(m)} \right]}{n-k}. \quad [k \geq 0]
\]

With these estimates, \( \hat{A}_{R_{in}}^{(m)} \) is given by

\[
\hat{A}_{R_{in}}^{(m)}(k) = \frac{\hat{C}_{R_{in}}^{(m)}(k)}{\hat{\sigma}^{2(m)}_{R_{in}}}, \quad [k \geq 0]
\]

and we find the final estimate of the ACF at the input side \( \hat{A}_{R_{in}} \) from

\[
\hat{A}_{R_{in}}(k) = \frac{\sum_{m=1}^{\eta} \hat{A}_{R_{in}}^{(m)}(k)}{\eta}. \quad [k \geq 0]
\]

Note that, other estimates of the ACF can also be used. For example, one could replace the \( n-k \) in the denominator of \( \hat{C}_{R_{in}}^{(m)}(k) \) by \( n \). The difficulty with estimators of the ACF is that they are biased, they have a large variance unless \( n \) is very large, and they are correlated with other correlation estimators. In particular, \( \hat{A}_{R_{in}}^{(m)}(n-1) \) will be a poor estimator of \( A_{R_{in}}(n-1) \), since it is based on the single product \( [R_{in,i+1}^{(m)} - \hat{\mu}_{R_{in}}^{(m)}][R_{in,i+n}^{(m)} - \hat{\mu}_{R_{in}}^{(m)}] \). Thus, in general, 'good' estimates of the \( A_{R_{in}}(k) \)'s are difficult to obtain unless \( n \) is very large and \( k \) is small relative to \( n \).

Now, in order to measure the accuracy of the estimates obtained for the BRD and the ACF at the input side, we define \( \hat{\Pi}_{in} \) and \( \hat{A}_{in} \) by

\[
\hat{\Pi}_{in} := \sum_{i=1}^{M_{R_{in}}} |\pi_i^{R_{in}} - \hat{\pi}_i^{R_{in}}| \quad \text{and} \quad \hat{A}_{in} := \sum_{k=0}^{k_{\text{max}}} |A_{R_{in}} - \hat{A}_{R_{in}}|,
\]

respectively, where \( k_{\text{max}} \) denotes the number of lags to be included. Further, we notate the confidence intervals for \( \hat{\Pi}_{in} \) (\( \hat{A}_{in} \)) based on the standard normal distribution and the
t-distribution by \( C^\text{in},n \) and \( C^\text{in},t \) (\( C^\text{in},n \) and \( C^\text{in},t \)).

Finally, we remark that the estimates of the BRD and the ACF at the output side \((\hat{\Pi}^{(m)}_{\text{out}}, \hat{A}^{(m)}_{\text{out}} \text{ and } \hat{A}_{\text{out}})\) are obtained in a similar way as the estimates at the input side, and all the necessary notation is obtained by replacing the process \( R^\text{in} \) with the process \( R^\text{out} \).

### 4.3.2 Results

Again, we want to examine how accurate the model estimates the true characteristics for particular sets of input parameters. Hereto, we focus on the same input parameters as discussed in subsection 4.2.3.

The results are listed in table 4.2 (top). Here, it is visible that the estimates of the BRD and ACF at the input side agree closely with the exact values for all nine examples. But, turning our attention to the estimates of these two bit rate statistics at the output side, we conclude that, especially in cases where \( \Delta x \) becomes small (\( \Delta x \approx 0.01 \)) the estimates become less accurate. Furthermore, cases for which we change several parameters of the standard example simultaneously lead to a long computation time. Of course, we can increase the length of the sequences involved \( (n) \) and the number of simulations \( (\eta) \) in order to obtain better estimates, but this increases the computation time repeatedly. For instance, if we let \( n = 5000 \) and \( \eta = 50 \) and focus on the estimates corresponding to the parameters in example eight, then we almost halve the errors in the estimates \( \hat{\Pi}^\text{in}, \hat{\Pi}^\text{out}, \hat{A}^\text{in} \) and \( \hat{A}^\text{out} \), but it takes the computer almost an hour to compute the results. The confidence intervals for the two bit rate statistics at both the input side and the output side, which are shown in table 4.2 (bottom), indicate that we obtained accurate estimates.

<table>
<thead>
<tr>
<th>( \Delta t )</th>
<th>( K )</th>
<th>( \Omega^\text{in} )</th>
<th>( \Omega^\text{C} )</th>
<th>( \hat{\Pi}^\text{in} )</th>
<th>( \hat{\Pi}^\text{out} )</th>
<th>( \hat{A}^\text{in} )</th>
<th>( \hat{A}^\text{out} )</th>
<th>( t_{\text{comp}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.007</td>
<td>0.020</td>
<td>0.136</td>
<td>0.158</td>
<td>0.3</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.5,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.032</td>
<td>0.036</td>
<td>0.065</td>
<td>0.100</td>
<td>0.2</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.005</td>
<td>0.048</td>
<td>0.080</td>
<td>0.090</td>
<td>7.0</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.021</td>
<td>0.190</td>
<td>0.095</td>
<td>0.062</td>
<td>0.2</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.019</td>
<td>0.199</td>
<td>0.071</td>
<td>0.131</td>
<td>3.2</td>
</tr>
<tr>
<td>1.0</td>
<td>10</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.019</td>
<td>0.049</td>
<td>0.096</td>
<td>0.134</td>
<td>0.2</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.024</td>
<td>0.049</td>
<td>0.100</td>
<td>0.149</td>
<td>3.2</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.013</td>
<td>0.202</td>
<td>0.120</td>
<td>0.178</td>
<td>632.6</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.043</td>
<td>0.064</td>
<td>0.091</td>
<td>0.105</td>
<td>628.6</td>
</tr>
</tbody>
</table>
4.4 Simulation of the buffer content

Finally, we obtain estimates for the buffer content. The structure of this section is similar to that of the previous two sections.

4.4.1 Notation and preparation

Recall from section 3.6 that the buffer content is determined by (3.1). Now, the estimates of the buffer content in the \( m \)-th replication, which are notated by \( \hat{\pi}^{(m)}_X \), are given by

\[
\hat{\pi}^{(m)}_X = \frac{\sum_{p=1}^{n} I\{X^{(m)}_p = x_k\}}{n}. \quad [s > s^w, x_k \in \Omega_X]
\]

The final estimates \((\hat{\pi}_X = (\hat{\pi}^X_1, \ldots, \hat{\pi}^X_{M_{\Omega}}))\) are obtained from the preceding display in the following way:

\[
\hat{\pi}^X_k = \frac{\sum_{m=1}^{\eta} \hat{\pi}^{X,(m)}_k}{\eta}.
\]

Now, we define \( \hat{\Pi}^X \) by

\[
\hat{\Pi}^X := \sum_{k=1}^{M_{\Omega}} |\hat{\pi}^X_k - \hat{\pi}^X_k|,
\]

and the confidence intervals based on the standard normal distribution and the \( t \)-distribution are notated by \( C_{\hat{\Pi}^X,n} \) and \( C_{\hat{\Pi}^X,t} \), respectively.
4.4.2 Results

Again, we consider the sets of input parameters as discussed in table 4.1 and table 4.2, and compare the estimates obtained for the buffer content with the corresponding exact values.

The results are presented in table 4.3. Here we draw similar conclusions as for the results obtained in table 4.1, namely, the estimates agree closely with the exact values, and cases for which we change several input parameters of the standard example simultaneously lead to an large computation time. Again, the confidence intervals indicate that we obtained accurate statistical results.

Table 4.3: Accuracy of the estimates obtained for the buffer content (left) with corresponding confidence intervals (right).

<table>
<thead>
<tr>
<th>$\Delta t$</th>
<th>$K$</th>
<th>$\Omega_{Po}$</th>
<th>$\Omega_C$</th>
<th>$\hat{X}$</th>
<th>$t_{comp}$</th>
<th>$\mathcal{C}_{\hat{X},n}$</th>
<th>$\mathcal{C}_{\hat{X},d}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.024</td>
<td>2.5</td>
<td>[0.02,0.03]</td>
<td>[0.02,0.03]</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.5,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.031</td>
<td>0.4</td>
<td>[0.02,0.04]</td>
<td>[0.02,0.04]</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.026</td>
<td>7.0</td>
<td>[0.02,0.03]</td>
<td>[0.02,0.03]</td>
</tr>
<tr>
<td>0.5</td>
<td>5</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.025</td>
<td>0.3</td>
<td>[0.01,0.04]</td>
<td>[0.01,0.04]</td>
</tr>
<tr>
<td>0.1</td>
<td>10</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.049</td>
<td>3.2</td>
<td>[0.03,0.06]</td>
<td>[0.04,0.06]</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.0,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.032</td>
<td>0.3</td>
<td>[0.02,0.04]</td>
<td>[0.02,0.04]</td>
</tr>
<tr>
<td>0.1</td>
<td>5</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.049</td>
<td>622.6</td>
<td>[0.03,0.07]</td>
<td>[0.03,0.07]</td>
</tr>
<tr>
<td>1.0</td>
<td>50</td>
<td>{1.1,3.0}</td>
<td>{2.0,4.0}</td>
<td>0.059</td>
<td>622.4</td>
<td>[0.04,0.07]</td>
<td>[0.05,0.07]</td>
</tr>
</tbody>
</table>
Part II

Matching procedure
Chapter 5

Genetic Algorithm

Recall from the first chapter that we created the discrete fluid flow model in order to obtain results that are hard to reach via measurements and to interpretate measurement results. Now that the model is essentially complete, we describe a procedure to find the values of the parameters such that the model reflects the real world as closely as possible. In order to find the best possible solution we use two optimization algorithms, namely the Genetic Algorithm (GA) and Sequential Quadratic Programming (SQP). In this chapter we focus on the GA, and the next chapter deals with SQP.

In the first section we explain the terminology associated with GAs and discuss the basic concepts. The most important concept discussed here is the fitness function, which enables the GA to determine the accuracy of a solution. For this reason, we discuss the fitness function in more detail in sections two and three, where we describe the fitness functions used for matching the input side and the output side, respectively. Other important input parameters for the GA are the boundary values. The boundary values provide the subject of the fourth section. Finally, the last sections deals with some modifications to the standard GA.

5.1 Basic concepts

In this section, concepts and terminology associated with GAs are defined and explained.

GAs were introduced by John Holland in 1975 in his book *Adaption in Natural and Artificial Systems* [9] and belong to the class of stochastic search optimization methods. The algorithm uses only the function in the search process to make progress towards a solution without regard to how the functions are evaluated. Continuity or differentiability of the problem functions is neither required nor used in calculations of the algorithms. Therefore, the algorithms are very general and can be applied to all kinds of problems. The method is easy to use since it does not require use of gradients of cost or constraint functions. A drawback of the algorithm is that there is no absolute guarantee
that a global solution has been obtained.

Now, because the GA is influenced by biology, this is reflected in the terminology. The five basic terms associated with the algorithm are the population, chromosomes, genes, generations and the fitness function, which are defined as follows:

**Population** The set of design points at the current iteration is called a population. It represents a group of designs as potential solution points. The number of designs in a population is called the population size ($N$).

**Chromosome** This term is used to represent a design point, thus a chromosome represents a design of the system.

**Gene** This term represents the value of a particular design variable.

**Generation** An iteration of the GA is called a generation.

**Fitness function** The fitness function defines the relative importance of a design. A higher fitness value implies a better design.

Figure 5.1 provides a better understanding of these concepts.

Figure 5.1: GA terminology: a population is formed by chromosomes and each chromosome has a set of genes.
The basic idea of the GA is to start with a set of designs, randomly generated using the allowable values for each design variable, and assign a fitness value to each design. Then, we generate a new set of designs (population) from the current set such that the average fitness of the population is improved. Since more fit members of the set are used to create new designs, the successive sets of designs have a higher probability of having designs with better fitness values. The process is continued until a stopping criterion is satisfied or the number of iterations exceeds a specified limit.

Three genetic operators are used to accomplish this task: reproduction, crossover and mutation. Reproduction is an operator where an old design is copied into the new population according to the design’s fitness. There are many different strategies to implement this reproduction operator. This is also called the selection process. The crossover operator corresponds to allowing selected members of a new population to exchange characteristics of their design among themselves. A mutation alters several genes in one design, determining a location at random.

The foregoing three steps are repeated for successive generations of the population until no further improvement in fitness is attainable. The member in this generation with the highest level of fitness is taken as optimum design. For a more detailed description of the GA please see [9] or [10].

5.2 Fitness function for matching the input side

In the matching procedure we match the input side and the output side separately, because a matching procedure for a combined input side and output side becomes too time-consuming. This means that we first derive the optimal parameters for the input process, and use the values found here to derive the optimal parameters for the capacity process. In this section we focus on the fitness function for matching the input side, and the next section deals with the fitness function for matching the output side.

The fitness function used for matching the input side compares the proposed BRDs and ACFs at the input side to the desired ones, where the desired ones are obtained from measurements. We first calculate the fitness values for the BRD and the ACF separately, and then we obtain the total fitness of the solution as a weighted sum of these two fitness values.

5.2.1 Fitness value for the BRD at the input side

As mentioned in the introduction of this section, we want to compare the estimated BRD obtained from the measurements to the computed BRD for the model. Now, recall that, we modelled the input process $R_{in}$ as an Markov chain with a fixed number of states $M_{Rin}$. But, the number of possible rates of the BRD obtained from the measurements may not
be equal to $M_{R\text{in}}$. Clearly, this makes it difficult to get an appropriate comparison between the two BRDs. To overcome this problem, we describe a procedure to adjust the BRD obtained from the measurements in such a way that the adjusted BRD has $M_{R\text{in}}$ possible rates.

**Adjusting the BRD obtained from the measurements**

Denote the possible rates and the corresponding probabilities for the BRD obtained from the measurements before the adjustments by

$$
\Omega_{R\text{in}}^\prime = \left\{ \tilde{r}_1^\text{in}, \ldots, \tilde{r}_{M_{R\text{in}}}^\text{in} \right\} \quad \text{and} \quad \pi_{R\text{in}}^\prime = \left( \tilde{\pi}_1^R_{\text{in}}, \ldots, \tilde{\pi}_{M_{R\text{in}}}^R_{\text{in}} \right),
$$

respectively, and let the corresponding expectation and variation be notated by $\tilde{\mu}_{R\text{in}}$ and $\tilde{\sigma}_{R\text{in}}^2$, respectively. Similarly, the possible rates and the corresponding probabilities for the BRD obtained from the measurements after the adjustments are notated by

$$
\tilde{\Omega}_{R\text{in}} = \left\{ \tilde{r}_1^\text{in}, \ldots, \tilde{r}_{M_{R\text{in}}}^\text{in} \right\} \quad \text{and} \quad \tilde{\pi}_{R\text{in}} = \left( \tilde{\pi}_1^R_{\text{in}}, \ldots, \tilde{\pi}_{M_{R\text{in}}}^R_{\text{in}} \right),
$$

with corresponding expectation and variation given by $\tilde{\mu}_{R\text{in}}^\prime$ and $\tilde{\sigma}_{R\text{in}}^2$.

Now, three different cases can occur, namely, $M_{R\text{in}} = \tilde{M}_{R\text{in}}$, $M_{R\text{in}} < \tilde{M}_{R\text{in}}$ or $M_{R\text{in}} > \tilde{M}_{R\text{in}}$. For all three cases we discuss how to adjust the BRD for the measurements, and we compare $\tilde{\mu}_{R\text{in}}$ and $\tilde{\sigma}_{R\text{in}}^2$ with $\tilde{\mu}_{R\text{in}}^\prime$ and $\tilde{\sigma}_{R\text{in}}^2$, respectively.

**$M_{R\text{in}} = \tilde{M}_{R\text{in}}$:** Here we do not have to adjust the BRD obtained from the measurements. Clearly, $\tilde{\mu}_{R\text{in}} = \tilde{\mu}_{R\text{in}}^\prime$ and $\tilde{\sigma}_{R\text{in}}^2 = \tilde{\sigma}_{R\text{in}}^2$.

**$M_{R\text{in}} > \tilde{M}_{R\text{in}}$:** In this case we need to enlarge the state space $\tilde{\Omega}_{R\text{in}}$ and adjust the probabilities $\tilde{\pi}_{R\text{in}}$ accordingly. We do this as follows:

$$
\tilde{\Omega}_{R\text{in}}^\prime := \tilde{\Omega}_{R\text{in}} \cup \left\{ \tilde{r}_{M_{R\text{in}}}^\text{in}, \ldots, \tilde{r}_{M_{R\text{in}}-M_{R\text{in}}+1}^\text{in} \right\}_{\text{times} M_{R\text{in}} - M_{R\text{in}}} \quad \text{and} \quad \tilde{\pi}_{R\text{in}}^\prime := \left\{ \begin{array}{ll}
\tilde{\pi}_{R\text{in}} & \text{for } i \in \{1,\ldots,M_{R\text{in}}-1\} \\
\frac{\tilde{\pi}_{M_{R\text{in}}}^R_{\text{in}}}{M_{R\text{in}} - M_{R\text{in}} + 1} & \text{for } i = M_{R\text{in}}.
\end{array} \right.
$$

With these adjustments, the expectation $\tilde{\mu}_{R\text{in}}^\prime$ is given by

$$
\tilde{\mu}_{R\text{in}}^\prime = \sum_{i=1}^{M_{R\text{in}}-M_{R\text{in}}+1} \tilde{r}_i^\text{in} \tilde{\pi}_i^R_{\text{in}} = \sum_{i=1}^{M_{R\text{in}}-1} \tilde{r}_i^\text{in} \tilde{\pi}_i^R_{\text{in}} + \tilde{r}_{M_{R\text{in}}}^\text{in} \cdot \left( M_{R\text{in}} - \tilde{M}_{R\text{in}} + 1 \right) \frac{\tilde{\pi}_{M_{R\text{in}}}^R_{\text{in}}}{M_{R\text{in}} - M_{R\text{in}} + 1} = \mu_{R\text{in}},
$$
and the variance $\tilde{\sigma}_{R_{in}}^2$ is given by

$$
\tilde{\sigma}_{R_{in}}^2 = \sum_{i=1}^{M_{R_{in}}-1} \left( \tilde{\pi}_{i}^R \right)^2 - \tilde{\mu}_{R_{in}}^2
= \sum_{i=1}^{M_{R_{in}}-1} \left( \tilde{\pi}_{i}^R \right)^2 - \tilde{\pi}_{R_{in}}^R \cdot \left( M_{R_{in}} - \tilde{M}_{R_{in}} + 1 \right) - \tilde{\mu}_{R_{in}}^2
= \tilde{\sigma}_{R_{in}}^2.
$$

$M_{R_{in}} < \tilde{M}_{R_{in}}$: Opposite from the previous case, we need to shrink the state space $\tilde{\Omega}_{R_{in}}$ and adjust the corresponding probabilities $\tilde{\pi}_{R_{in}}$ accordingly. Now, until the number of elements in the adjusted state space equals $M_{R_{in}}$, we do the following. First, we find the minimum value in $\tilde{\Omega}_{R_{in}}$, notation $\tilde{r}_{(1)}^R$, and the minimum value in $\tilde{\Omega}_{R_{in}}/\tilde{r}_{(1)}^R$, notation $\tilde{r}_{(2)}^R$.

The corresponding probabilities are denoted by $\tilde{\pi}_{(1),R_{in}}^R$ and $\tilde{\pi}_{(2),R_{in}}^R$, respectively. Next, we let

$$
\tilde{\pi}_{(1,2),R_{in}}^R : = \tilde{\pi}_{(1),R_{in}}^R + \tilde{\pi}_{(2),R_{in}}^R \quad \text{and} \quad \tilde{r}_{(1,2),R_{in}}^R : = \tilde{r}_{(1),R_{in}}^R + \frac{\tilde{\pi}_{(2),R_{in}}^R}{\tilde{\pi}_{(1,2),R_{in}}^R} \left( \tilde{r}_{(2),R_{in}}^R - \tilde{r}_{(1),R_{in}}^R \right).
$$

Finally, we adjust $\tilde{\pi}_{R_{in}}$ and $\tilde{\Omega}_{R_{in}}$ to

$$
\tilde{\pi}_{R_{in}} : = \left( \frac{\tilde{\pi}_{R_{in}}}{\tilde{\pi}_{(1),R_{in}}^R \vee \tilde{\pi}_{(2),R_{in}}^R} \right) \tilde{\pi}_{(1,2),R_{in}}^R \quad \text{and} \quad \tilde{\Omega}_{R_{in}} : = \left\{ \tilde{\Omega}_{R_{in}} \cup \tilde{r}_{(2),R_{in}}^R \cup \tilde{r}_{(1,2),R_{in}}^R \right\}.
$$

We first show that, in each step of this procedure the expectation remains the same, and after that, we show that the variance never increases. The state space and the corresponding probabilities before (after) an arbitrary step are denoted by $\tilde{\Omega}_{R_{in}}^{(b)}$ and $\tilde{\pi}_{R_{in}}^{(b)}$ ($\tilde{\Omega}_{R_{in}}^{(a)}$ and $\tilde{\pi}_{R_{in}}^{(a)}$), respectively. Simiarly, we denote the expectation and the variance before (after) an arbitrary step by $\tilde{\mu}_{R_{in}}^{(b)}$ and $\tilde{\sigma}_{R_{in}}^{2(b)}$ ($\tilde{\mu}_{R_{in}}^{(a)}$ and $\tilde{\sigma}_{R_{in}}^{2(a)}$), respectively.

Now, the expectation after an arbitrary step is given by

$$
\tilde{\mu}_{R_{in}}^{(a)} = \sum_{i=1}^{M_{R_{in}}-1} \tilde{r}_{i}^{R_{in}} \tilde{\pi}_{i}^{R_{in}} - \tilde{r}_{(1)}^{R_{in}} \tilde{\pi}_{(1)}^{R_{in}} - \tilde{r}_{(2)}^{R_{in}} \tilde{\pi}_{(2)}^{R_{in}} + \tilde{r}_{(1,2)}^{R_{in}} \tilde{\pi}_{(1,2)}^{R_{in}}
= \sum_{i=1}^{M_{R_{in}}-1} \tilde{r}_{i}^{R_{in}} \tilde{\pi}_{i}^{R_{in}} - \tilde{r}_{(1)}^{R_{in}} \tilde{\pi}_{(1)}^{R_{in}} - \tilde{r}_{(2)}^{R_{in}} \tilde{\pi}_{(2)}^{R_{in}} + \left( \tilde{r}_{(1)}^{R_{in}} + \frac{\tilde{\pi}_{(2)}^{R_{in}}}{\tilde{\pi}_{(1)}^{R_{in}}} \left( \tilde{r}_{(2)}^{R_{in}} - \tilde{r}_{(1)}^{R_{in}} \right) \right) \left( \tilde{\pi}_{(1)}^{R_{in}} + \tilde{\pi}_{(2)}^{R_{in}} \right)
= \tilde{\mu}_{R_{in}}^{(b)}.
$$
Next, in order to show that the variance never increases in an arbitrary step, we show that the difference between $\tilde{\sigma}_{R^{in}}^2(a)$ and $\tilde{\sigma}_{R^{in}}^2(b)$ is smaller or equal to zero. This difference can be written as
\[
\tilde{\sigma}_{R^{in}}^2(a) - \tilde{\sigma}_{R^{in}}^2(b) = \left( \tilde{r}^{in}(1,2) \right)^2 \tilde{\pi}_{R^{in}}^{(1,2)} - \left( \tilde{r}^{in}(1,1) \right)^2 \tilde{\pi}_{R^{in}}^{(1,1)} - \left( \tilde{r}^{in}(1,2) \right)^2 \tilde{\pi}_{R^{in}}^{(1,2)}
\]
\[
= \left( \tilde{r}^{in}(1) + \frac{\tilde{\pi}_{R^{in}}^{(2)}}{\tilde{\pi}_{R^{in}}^{(1,2)}} \left( \tilde{r}^{in}(2) - \tilde{r}^{in}(1) \right) \right)^2 \left( \tilde{\pi}_{R^{in}}^{(1,1)} + \tilde{\pi}_{R^{in}}^{(2)} \right) - \left( \tilde{r}^{in}(1) \right)^2 \tilde{\pi}_{R^{in}}^{(1,1)} - \left( \tilde{r}^{in}(2) \right)^2 \tilde{\pi}_{R^{in}}^{(2)}
\]
\[
= -\left( \tilde{r}^{in}(2) - \tilde{r}^{in}(1) \right)^2 \left( \frac{\tilde{\pi}_{R^{in}}^{(1,2)}}{\tilde{\pi}_{R^{in}}^{(1,1)} + \tilde{\pi}_{R^{in}}^{(2)}} \right),
\]
and because $\tilde{\pi}_{R^{in}}^{(1,1)} > 0$, $\tilde{\pi}_{R^{in}}^{(2)} > 0$ and $\tilde{r}^{in}(2) \leq \tilde{r}^{in}(1)$, the obtained value is indeed smaller or equal to zero.

In order to provide a better understanding of this procedure, we discuss an example in which $M_{R^{in}} < \bar{M}_{R^{in}}$.

**Example** Suppose that the state space and the corresponding probabilities obtained from the measurements are given by
\[
\tilde{\Omega}_{R^{in}} = \{2, 3, 4, 5\} \quad \text{and} \quad \tilde{\pi}_{R^{in}} = \left( \frac{1}{10}, \frac{1}{5}, \frac{2}{5}, \frac{3}{10} \right),
\]
respectively, and that the number of states in the input Markov chain $R^{in}$ equals two, i.e. $M_{R^{in}} = 2$. These values lead to an expectation and an variation of $\tilde{\mu}_{R^{in}} = 3\frac{9}{10}$ and $\tilde{\sigma}_{R^{in}}^2 = 6\frac{8}{100} = 0.89$, respectively. Now, in the first step of the procedure described above $\tilde{r}^{in}(1) = 2$ and $\tilde{r}^{in}(2) = 3$, and the corresponding probabilities are given by $\tilde{\pi}_{R^{in}}^{(1,2)} = \frac{1}{10}$ and $\tilde{\pi}_{R^{in}}^{(2)} = \frac{1}{5}$, respectively, hence $\tilde{\pi}_{R^{in}}^{(1,2)} = \frac{3}{10}$ and $\tilde{r}^{in}(1,2) = 2\frac{2}{3}$. This means that the state space and the corresponding probabilities after the first step are given by
\[
\tilde{\Omega}_{R^{in}}^{(a)} = \left\{ \frac{2}{3}, 4, 5 \right\} \quad \text{and} \quad \tilde{\pi}_{R^{in}}^{(a)} = \left( \frac{3}{10}, \frac{2}{5}, \frac{3}{10} \right),
\]
respectively, and these values lead to an expectation and an variation of $\tilde{\mu}_{R^{in}} = 3\frac{9}{10}$ and $\tilde{\sigma}_{R^{in}}^{2(a)} = 2\frac{47}{300} \approx 0.82$, respectively. In the next step $\tilde{r}^{in}(1) = 2\frac{2}{3}$ and $\tilde{r}^{in}(2) = 4$ and the corresponding probabilities are given by $\tilde{\pi}_{R^{in}}^{(1)} = \frac{3}{10}$ and $\tilde{\pi}_{R^{in}}^{(2)} = \frac{4}{5}$, thus $\tilde{\pi}_{R^{in}}^{(1,2)} = \frac{7}{10}$ and $\tilde{r}^{in}(1,2) = 3\frac{2}{5}$. Here we have that
\[
\tilde{\Omega}_{R^{in}}^{(a)} = \left\{ \frac{3}{5}, 4, 5 \right\} \quad \text{and} \quad \tilde{\pi}_{R^{in}}^{(a)} = \left( \frac{7}{10}, \frac{3}{10} \right),
\]
which again leads to an expectation of \( \bar{\mu}_{R^{in}}^{(a)} = \frac{9}{10} \) and the variation decreases to \( \sigma_{R^{in}}^{2(a)} = \frac{363}{700} \approx 0.52 \). Now, because the number of elements in \( \tilde{\gamma}_{R^{in}}^{(a)} \) equals \( M_{R^{in}} \), we stop the procedure. Note that in each step of the procedure the expectation equals \( \frac{9}{10} \), and the variance decreases first from 0.89 to 0.82, and then from 0.82 to 0.52.

Now that we adjusted the BRD of the measurements, we are ready to calculate the fitness value for the BRD at the input side.

The fitness value for the BRD at the input side \( (F_{R^{in}}, \pi_{R^{in}}) \) is a combination of a fitness value obtained for the rates \( (F_{R^{in}}) \) and a fitness value obtained for the corresponding probabilities \( (F_{\pi_{R^{in}}}) \). For this reason, we obtain \( F_{R^{in}, \pi_{R^{in}}} \) in three steps. In the first step we discuss how to obtain \( F_{R^{in}} \), in the second step we use a similar approach to obtain \( F_{\pi_{R^{in}}} \), and in the final step we combine these two values and obtain \( F_{R^{in}, \pi_{R^{in}}} \).

**Step 1: Fitness value for the rates \( F_{R^{in}} \)**

Let the error vector for the rates \( U_{R^{in}} \) be defined by \( U_{R^{in}} := (U_{1}^{R^{in}}, \ldots, U_{M_{R^{in}}}^{R^{in}}) \), where \( U_{i}^{R^{in}} := |r_{i}^{in} - \bar{r}_{i}^{in}| \). From this, the fitness for each state \( i \) is obtained as:

\[
F_{i}^{R^{in}} = \begin{cases} 
F_{R^{in}}^{\text{max}} & \text{if } U_{i}^{R^{in}} < L_{1}^{R^{in}}; \\
\Delta_{R^{in}} \cdot U_{i}^{R^{in}} + a_{R^{in}} & \text{if } L_{1}^{R^{in}} \leq U_{i}^{R^{in}} \leq L_{2}^{R^{in}}; \\
(U_{i}^{R^{in}} - (L_{2}^{R^{in}} - v_{R^{in}})^{p_{R^{in}}})^{-1} & \text{if } U_{i}^{R^{in}} > L_{2}^{R^{in}}.
\end{cases}
\]  

(5.1)

The fitness function is divided in three parts, and we discuss the shape of the fitness function for each of the parts. Figure 5.2 shows a sketch of the fitness function defined in the preceding display. Here we discard the process \( R^{in} \) from the notation because this fitness function will be used for other processes as well.

First, if \( U_{i}^{R^{in}} < L_{1}^{R^{in}} \), then the fitness function is constant. The reason to use an maximum fitness value is the following. It might be the case that we have an almost perfect match for the first state. If we don’t use a maximum fitness value, this value might dominate the total fitness.

Next, if \( L_{1}^{R^{in}} \leq U_{i}^{R^{in}} \leq L_{2}^{R^{in}} \), then the fitness function is linear. \( \Delta_{R^{in}} \) and \( a_{R^{in}} \) specify a line that goes from \( (L_{1}^{R^{in}}, F_{R^{in}}^{\text{max}}) \) to \( (L_{2}^{R^{in}}, a_{R^{in}} + \Delta_{R^{in}} \cdot L_{2}^{R^{in}}) \) and satisfy the equation

\[
a_{R^{in}} = F_{R^{in}}^{\text{max}} - \Delta_{R^{in}} \cdot L_{1}^{R^{in}}.
\]  

(5.2)
Finally, if $U_{R_i} > L_{R_i}^2$, then the fitness function is reciprocal. $v_{R_i}$ and $p_{R_i}$ allow us to specify the shape of the reciprocal part and satisfy the equation

$$v_{R_i} = \left( a_{R_i} + \Delta_{R_i} \cdot L_{R_i}^2 \right)^{-\frac{1}{p_{R_i}}}.$$ \hspace{1cm} (5.3)

We use a reciprocal part for the following reason. Recall that the GA tries to find solutions with a higher fitness value for each generation. Now, if we only use a linear part, then the fitness value equals zero for a large set of designs. Our experience is that the GA finds it harder to make progress towards an optimal solution in these cases.

Figure 5.3 illustrates the shape of the fitness function for several choices of the parameters.

Now that we evaluated the fitness value for each rate separately, we calculate $F_{R_i}$ as a weighted sum of the values $F_{i,R_i}$. If we denote the weights by $w_{R_i} = (w_{1,R_i}^{R_i}, \ldots, w_{M_{R_i}}^{R_i})$, where $w_{i,R_i}$ controls the influence of the $i$-th rate in the weighted sum, then $F_{R_i}$ is given by

$$F_{R_i} = \sum_{i=1}^{M_{R_i}} w_{i,R_i}^{R_i} F_{i,R_i}.$$  

**Step 2: Fitness value for the probabilities $F_{\pi_{R_i}}$**

The fitness value for the corresponding probabilities is obtained in exactly the same way as the fitness value for the rates as discussed above.
Step 3: Fitness value for the BRD $F_{R_{in}, \pi_{R_{in}}}$

The fitness value for the BRD at the input side ($F_{R_{in}, \pi_{R_{in}}}$) is a weighted sum of the fitness value for the rates ($F_{R_{in}}$) and the fitness value for the corresponding probabilities ($F_{\pi_{R_{in}}}$). If we let the weight $w_{R_{in}, \pi_{R_{in}}}$ control the influence of $F_{R_{in}}$ in the weighted sum, then $F_{R_{in}, \pi_{R_{in}}}$ is given by

$$F_{R_{in}, \pi_{R_{in}}} = w_{R_{in}, \pi_{R_{in}}} \cdot F_{R_{in}} + \left(1 - w_{R_{in}, \pi_{R_{in}}} \right) \cdot F_{\pi_{R_{in}}}.$$ 

Figure 5.3: Shape of the fitness function for several choices of the parameters.
5.2.2 Fitness value for the ACF at the input side

Here, we discuss how to calculate the fitness value for the ACF at the input side. Again we use a similar approach as the one described in subsection 5.2.1, where we describe how to obtain the fitness value for the rates $F_{R_{\text{in}}}$. So, we first calculate the fitness per lag using the fitness function illustrated in figure 5.2, and then we calculate the total fitness value for the ACF as a weighted sum of the fitness values of the different lags.

5.2.3 Total fitness value

Now that we evaluated the fitness value for the BRD at the input side ($F_{R_{\text{in}},\pi_{R_{\text{in}}}}$) and the fitness value for the ACF at the input side ($F_{A_{R_{\text{in}}}}$), we obtain the total fitness value ($F_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}}$) as a weighted sum of these values. If we let the weight $w_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}}$ control the influence of $F_{R_{\text{in}},\pi_{R_{\text{in}}}}$ in the weighted sum, then $F_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}}$ is given by

$$F_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}} = w_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}} \cdot F_{R_{\text{in}},\pi_{R_{\text{in}}}} + \left(1 - w_{(R_{\text{in}},\pi_{R_{\text{in}}}),A_{R_{\text{in}}}}\right) \cdot F_{A_{R_{\text{in}}}}.$$ 

5.3 Fitness function for matching the output side

As mentioned in the introduction of the previous section, we match the input side and the output side separately. For the output side we use a similar approach as for the input side. So, we compare the proposed BRDs and ACFs at the output side to the desired ones and calculate the fitness values using the fitness function illustrated in figure 5.2. The total fitness is again obtained as a combination of the fitness values obtained for the BRD and the ACF, and we use weights to control the influence of the two bit rate statistics.

Because the fitness value for the output side is obtained in exactly the same way as the fitness value for the input side, we leave out all further details. Only the method described in subsection 5.2.1, where we adjust the BRD of the measurements, differs slightly from the method used for the output side. For this reason, we shortly discuss the method used for the output side.

5.3.1 Adjusting the BRD of the model

Recall from subsection 5.2.1 that, in order to obtain the fitness value for the BRD at the input side, we adjusted the BRD obtained from the measurements. This was necessary because we modeled the input Markov chain with a fixed number of states. For the output side we do not have this limitation. Of course, we still need to overcome the problem that the number of possible rates of the BRD obtained from the measurements may not be equal to $M_{R_{\text{out}}}$. Instead of adjusting the BRD obtained from the measurements, we now adjust the BRD computed from the model. With this approach we mimic the real world
as closely as possible. We use a similar approach as the one described in subsection 5.2.1, and similar conclusion can be drawn for the expectation and the variance.

5.4 Boundaries

Ideally the GA should have no boundaries, besides the number of parameters, but allowing a parameter to assume values in the range of \([-\infty, \infty]\) can cause problems. By bounding the parameters we reduce the fitness landscape that the GA has to search, without removing too many possible solutions. In the next two subsections we focus on the boundaries for the parameters at the input side and the output side, respectively.

5.4.1 Boundaries at the input side

At the input side of the model, the GA is given the task to find the elements in the state space \(\Omega_{R_{in}}\) and the probabilities in the transition matrix \(P_{R_{in}}\) such that the model matches the observations as closely as possible. Clearly, these parameters need to satisfy boundary constraints.

For the probabilities in the transition matrix the elements in each row should add up to one and the probabilities lie between zero and one, so the following constraints need to be satisfied

\[
\sum_{i_2=1}^{M_{R_{in}}} p_{i_1i_2}^{R_{in}} = 1 \quad [i_1 \in \{1, \ldots, M_{R_{in}}\}]
\]

and

\[
0 < p_{i_1i_2}^{R_{in}} < 1 \quad [i_1, i_2 \in \{1, \ldots, M_{R_{in}}\}]
\]

In order to reduce the number of transition parameters from \(M_{R_{in}}^2\) to \((M_{R_{in}} - 1)^2\), we replace the equality constraints in the preceding display by the following inequality constraints

\[
0 < \sum_{i_2=1}^{M_{R_{in}}-1} p_{i_1i_2}^{R_{in}} < 1, \quad [i_1 \in \{1, \ldots, M_{R_{in}}\}]
\]

and define \(p_{i_1, M_{R_{in}}}^{R_{in}}\) as one minus this sum. For the elements in the state space \(\Omega_{R_{in}}\) we use the following constraints

\[
r_{i_1}^{in} < r_{i_2}^{in} \quad [i_1 < i_2]
\]

and

\[
b_{r_{i_1},l}^{in} < r_{i_1}^{in} < b_{r_{i_1},u}, \quad [r_{i_1}^{in} \in \Omega_{R_{in}}]
\]

where the boundaries \(b_{r_{i_1},l}\) and \(b_{r_{i_1},u}\) should be based on the measured data.

Unfortunately, the GA captures only the boundary constraints for each variable separately, and cannot be used directly for other constraints. Now, there are two ways of
implementing other constraints. Either we give a penalty whenever the fitness function is called with a vector which does not satisfy one of the constraints, or we modify the code in such a way that we start with a population of designs which satisfy all the constraints, and everytime we modify one (or a part of a) design we make sure that the constraints are satisfied. The first method has as drawback that, with a certain probability “strange” values can occur, but as advantage that we do not have to change the code. The second method never produces an outcome which does not satisfy the constraints but slows down the algorithm. We implement both methods and compare them.

5.4.2 Boundaries at the output side

At the output side of the model we focus on constraints for the elements in the state space $\Omega_C$ and constraints for the probabilities in the transition matrix $\Omega_C$. The constraints are of the same form as the ones described in the previous subsection for $\Omega_{\text{Pin}}$ and $P_{\text{Pin}}$. Again, we discuss a penalty method and a method where we modify the code, and compare the two methods.

5.5 Modifications to the standard GA

In this section we discuss four modifications to the standard GA. First, we discuss the amount of crossover and mutation used to generate a new set of designs. Then, we point out the different stopping criteria used in this work. After that, we discuss a discretization method for the obtained optimal design. Finally, we describe a fast method to obtain a start population. For the first three modifications we focus only on the input side since we use a similar approach for the output side. The last modification is only used at the output side.

5.5.1 Amount of crossover and mutation

Clearly, the speed of the GA depends on the number of parameters, and hence on the number of states in the Markov chain $\mathcal{R}_{\text{Pin}}$. We should be careful with choosing too many states for the process $\mathcal{R}_{\text{Pin}}$, because this can lead to a large amount of calculations. In order to fine-tune the performance of the GA, we increase the population size, the number of crossovers ($\#c$) and the number of mutations ($\#m$) whenever the number of states increases. Table 5.1 illustrates this dependency. Here we set the number of crossovers to a small number, because too many crossovers can result in a poorer performance of the algorithm. This is due to the fact that a large amount of crossovers can produce designs that are far away from the mating designs. The mutation, however, changes designs in the neighborhood of the current design. Therefore a larger amount of mutation may be allowed. Finally, the populationsize is set to a reasonable number for each problem. The
values in table 5.1 are obtained from the following equations:

\[ N := \min(50 \cdot M_{\text{Rin}}, 200) \]
\[ \#c := \min(\max(2, M_{\text{Rin}}), 4) \]
\[ \#m := \min(\max(3, M_{\text{Rin}}), 10) \]

Table 5.1: Dependency between the number of states \( (M_{\text{Rin}}) \) and the population size \( (N) \), the number of mutations \( (\#m) \) and the number of crossovers \( (\#c) \).

<table>
<thead>
<tr>
<th>( M_{\text{Rin}} )</th>
<th>( N )</th>
<th>( #c )</th>
<th>( #m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>( \geq 4 )</td>
<td>200</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

5.5.2 Stopping criteria

At each generation, the member having the highest fitness value among all the designs is defined as the leader of the population. If several members have the highest fitness value, then one of them is chosen as the leader. The leader is replaced if another member with a higher fitness value appears. This way, it is safeguarded from extinction (as a result of reproduction, crossover and mutation). In addition, the leader is guaranteed a higher probability of selection for reproduction.

Now, the question is when we should terminate the GA and take the member with the highest level of fitness as the optimum design. We now describe the three stopping criteria that are used in this work.

For the first stopping criterion we describe a doubling strategy. If the member with the highest fitness remains the same for the past two consecutive generations, then we double \( \#m \). Now, if we not make any progress for another two generations, we also double \( \#c \). If this still does not lead to a design with a higher fitness in two generations, then the progress towards the solution point is too slow. In this case we randomly select fifty percent of the population and introduce completely new designs into the population in effort to increase diversity. For the new population we use a similar approach as before, i.e. doubling \( \#m \) and \( \#c \) after two and four generations without improvement, respectively. If there is still no progress, then we stop.

The \( \#m \) and \( \#c \) return to their original values as soon as the fitness increases. The concept behind this is that we do not want too much crossover or mutation to ruin the good designs as long as they keep producing better offspring. On the other hand, we
need more crossover and mutation to trigger changes when progress stops.

A second reason to stop is that an optimal value is found. This means that the difference between the fitness value of the leader of the population and the maximum attainable fitness function is smaller than a specified number.

Finally, we also stop if the number of iterations exceeds a specified value.

5.5.3 Discretization

As pointed out in subsection 2.1.3, we do not allow all possible values for the rates, but we assume the rates to be multiples of certain base rates. This means that the elements in \( \Omega_{R_{\text{in}}} \) are restricted to be members of a finite set of values. For this reason, we first solve the continuous optimization problem and then adjust the optimal design using the nearest discrete values.

Denote the parameters of the continuous problem and the corresponding optimal discrete values by

\[
\Omega^{*,R,GA}_{R_{\text{in}}} := \{ r_{1,\text{in},*,R,GA}^{\text{in}}, \ldots, r_{M_{\text{in}},\text{in},*,R,GA}^{\text{in}} \} \quad \text{and} \quad \Omega^{*,D,GA}_{R_{\text{in}}} := \{ r_{1,\text{in},*,D,GA}^{\text{in}}, \ldots, r_{M_{\text{in}},\text{in},*,D,GA}^{\text{in}} \}
\]

respectively. Now, the goal is to find the optimal discrete values \( r_{i,\text{in},*,D,GA} \) which are multiples of \( \frac{1}{100} \), i.e. such that

\[
h_i := r_{i,\text{in},*,D,GA}^{\text{in}} - \frac{1}{100} \cdot \left[ 100 \cdot r_{i,\text{in},*,D,GA}^{\text{in}} \right] = 0 \quad [i \in \{ 1, \ldots, M_{\text{in}} \}]
\]

We obtain these optimal discrete values from the optimal continuous values using the following strategy. We first check for each of the continuous values if \( h_i \) equals zero. For each of the continuous values that satisfy this property, we let \( r_{i,\text{in},*,D,GA} := r_{i,\text{in},*,R,GA} \). For the remaining values we either round \( r_{i,\text{in},*,R,GA} - h_i \) to its upper bound or to its lower bound, i.e.

\[
r_{i,\text{in},*,D,GA} := r_{i,\text{in},*,R,GA}^{\text{in}} - h_i - \frac{1}{100} \quad \text{or} \quad r_{i,\text{in},*,D,GA} := r_{i,\text{in},*,R,GA}^{\text{in}} - h_i + \frac{1}{100}.
\]

Say that this leads to \( \zeta \) solutions \( (0 \leq \zeta \leq 2^{M_{\text{in}}}) \). Now, for all \( \zeta \) solutions we compute the corresponding fitness value and we choose the design with the highest score. Typically, the solution of the discrete problem is close to the solution of the continuous problem.

5.5.4 Start population

Initialization of a population to provide the GA a starting point is usually done by generating random strings within the search space, and this is the default behaviour. Now,
because we need to determine the fitness for each of the design points in this start population, and hence calculate the BRD and the ACF, producing the start population already takes a long time. For this reason, we preconstructed twenty start populations and each time we run the GA, we pick one of these populations at random.
Chapter 6

Sequential Quadratic Programming

GAs are capable of quickly finding promising regions of the search space but may take a relatively long time to reach the optimal solution. For this reason, we enhance the GA with a local improvement operator. The local improvement operator used in this work is called SQP and provides the subject of this chapter.

In the first section we explain the basic concepts associated with SQP. The second section deals with three important input parameters for the SQP algorithm, namely the fitness function, the boundary constraints and the merit function. The last section discusses modifications to the standard SQP algorithm.

6.1 Basic concepts

Since its popularization in the late 1970s, SQP has become one of the most successful methods for solving constrained optimization problems. As with most optimization methods, SQP is not a single algorithm, but rather a conceptual method from which numerous specific algorithms have evolved. The programming problem to be solved is

\[
\min_{x \in A} f(x)
\]

\[
A = \{ x \in \mathbb{R}^m : u_i(x) = 0, \ i \in \mathcal{E}, \ i \in \mathcal{I} \},
\]

where \( u_i(x) \) is a set of equality and inequality constraints (denoted by \( \mathcal{E} \) and \( \mathcal{I} \), respectively).

The basic idea of SQP is that at each step, a local model of the optimization problem is constructed and solved, yielding a step (hopefully) toward the solution of the original problem. Now, because SQP methods can be viewed as a generalization of Newton’s method, they share the characteristics of Newton-like methods, namely, rapid convergence when the iterates are close to the solution but possible erratic behaviour that needs to be carefully controlled when the iterates are far from a solution. While this correspondence is
valid in general, the presence of constraints makes both the analysis and implementation of SQP methods significantly more complex.

Two additional properties of the SQP method should be pointed out. First, SQP is not a feasible-point method, that is, neither the initial point nor any of the subsequent iterates need be feasible (a feasible point satisfies all the constraints). This is a major advantage since we start the SQP from the point determined by the genetic string. Second, the success of the SQP methods depends on the existence of rapid and accurate algorithms for solving quadratic programs. Fortunately, quadratic programs are easy to solve.

6.2 Input parameters

In this section we discuss three important input parameters for the SQP algorithm, namely the fitness function, the boundary constraints and the merit function.

6.2.1 Fitness function

The fitness functions used for the SQP algorithm are of the same form as the ones described in the previous chapter for the GA. So, we again compare the desired BRDs and ACFs to the proposed ones, and obtain fitness values using the fitness function illustrated in figure 5.2.

6.2.2 Boundary constraints

Recall from subsection 5.4.1 that the GA only captures the boundary constraints for each variable separately and cannot be used directly for other constraints. To overcome this problem, we discussed two methods, namely, a penalty method and a method where we modified the code. Fortunately, the SQP algorithm does not have this limitation.

Further, we decreased the number of variables for the transition matrix $P_{R_{in}}$ from $M_{R_{in}}^2$ to $(M_{R_{in}} - 1)^2$ by replacing the equality constraints displayed in equation (5.4) by the inequality constraints displayed in equation (5.5). Now, since the SQP algorithm can be improved using equality constraints instead of inequality constraints, we implement both type of constraints. Again, we compare both methods and use the method which leads to better numerical results.

6.2.3 Merit function

For constrained problems, we would like a next iterate not only to decrease $f$ but also to come closer to satisfying the constraints. Often these two aims conflict, so it is necessary to weight their relative importance. Now, as a criterion for determining whether or not
one point is better than another, we define a merit function. The merit function that we use is given by

$$
\psi(x) := f(x) + \sum_{i \in \mathcal{E}} v_i |u_i(x)| + \sum_{i \in \mathcal{I}} v_i \max(0, u_i(x)),
$$

where $v_i > 0$ are penalty parameters. By convention the merit function is small when the agreement is good.

### 6.3 Modifications to the standard SQP algorithm

This section deals with modifications to the standard SQP algorithm. We first consider the stopping criteria used in the SQP algorithm, and then we discuss a discretization strategy.

#### 6.3.1 Stopping criteria

Apart from the three stopping rules discussed in subsection 5.5.2 for the GA, we equip the SQP algorithm with a fourth stopping rule involving the number of function evaluations. So, we stop the algorithm if the number of function evaluations exceeds a specified number. We implement this stopping rule because each time we call the fitness function, we need to calculate the BRD and the ACF and this can lead to a long computation time (see subsection 4.3.2).

#### 6.3.2 Discretization

For the solution obtained from the SQP algorithm we use a similar discretization strategy as the one described in subsection 5.5.3 for the GA. The parameters of the continuous problem and the corresponding optimal discrete values are notated by

$$
\Omega^*, \mathbb{R}, \text{SQP}_{\text{Rin}} := \left\{ r_{1,1}^{\text{in,*}, \mathbb{R}, \text{SQP}}, \ldots, r_{M_{\text{Rin}}}^{\text{in,*}, \mathbb{R}, \text{SQP}} \right\}
$$

and

$$
\Omega^*, \mathbb{D}, \text{SQP}_{\text{Rin}} := \left\{ r_{1,1}^{\text{in,*}, \mathbb{D}, \text{SQP}}, \ldots, r_{M_{\text{Rin}}}^{\text{in,*}, \mathbb{D}, \text{SQP}} \right\},
$$

respectively.
Chapter 7

Matching preparations

This chapter provides an introduction to the use of experimental designs and optimization techniques. All the concepts discussed here are used in the next chapter to obtain the desired information. In experimental design terminology, the input parameters and structural assumptions composing a model are called factors, and the output performance measures are called responses.

In the first section we discuss so-called factorial designs. These designs provide a way of deciding which combination of factors optimize a given model. A second technique to attain this goal is called meta-modeling and is described in the next section. The last section deals with an procedure to determine the best out of several methods.

7.1 Factorial designs

The first design to estimate how changes in input factors effect the responses of the experiment is called a $2^m$ factorial design. Here, it is assumed that just two values, or levels, of each of the input factors can be identified. There is no general prescription on how to set these levels, but they should be set “opposite” in nature but not so extreme that they are unrealistic. In the case of $m$ input factors, there are thus $2^m$ different combinations of the input factors. Since we only consider factorial designs with two input factors, we set $m$ to two.

Refering to the two levels of each of the two factors as the “-” and the “+” level, a design matrix can be formed, which describes the responses in terms of the input factor levels. If we denote the response from the $i$th combination by $S_i$, then the design matrix corresponding to a $2^2$ factorial design is represented by table 7.1.

Now, the main effect of a factor $i$ is defined as the average change in the response due to moving factor $i$ from its “-” level to its “+” level, while holding all other factors fixed. It can be computed by applying the signs in the factor column to the corresponding
Table 7.1: Design matrix for a $2^2$ factorial design.

<table>
<thead>
<tr>
<th>Factor combination</th>
<th>Factor 1</th>
<th>Factor 2</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>$S_1$</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>-</td>
<td>$S_2$</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>+</td>
<td>$S_3$</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>+</td>
<td>$S_4$</td>
</tr>
</tbody>
</table>

responses, adding, and then dividing by four. The main effects of factor one and two are given by

$$
\mathcal{E}_1 = \frac{-S_1 + S_2 - S_3 + S_4}{4} \quad \text{and} \quad \mathcal{E}_2 = \frac{-S_1 - S_2 + S_3 + S_4}{4},
$$

respectively. Further, the effect of one of the factors might depend in some way on the level of the other factor, which is called interaction. The interactions are computed by “multiplying” the columns of the involved factors row by row (like signs multiply to “+”, unlike signs multiply to “-”), applying the resulting signs to the corresponding responses, adding, and dividing by four. The interaction between factors one and two is given by

$$
\mathcal{E}_3 = \frac{S_1 - S_2 - S_3 + S_4}{4}.
$$

If an interaction appears to be present between two factors, then the main effects of the factors cannot be interpreted as simply the effect in general of moving these factors from their “-” level to their “+” level. Another limitation of factorial designs is that the responses are random variables, thus the main effects and interactions are subject to variation. To overcome this difficulty a good approach is to replicate the whole design many times to get many estimates of main effects and interactions, which could then be combined to form confidence intervals on the expected main effects and interactions in the usual way. If the confidence intervals do not contain zero, then the effects appear to be really present.

### 7.2 Meta-models

This section deals with a second technique to measure how changes in input factors effect responses, called meta-modeling.

Most experimental designs, are based on an algebraic regression-model assumption about the way the input factors affect the outputs. For instance, the affect of the two factors $y_1$ and $y_2$ on an output response $y$ might be approximated by the regression model

$$
y = \beta_0 + \beta_1 y_1 + \beta_2 y_2 + \beta_3 y_1 y_2 + \beta_4 y_1^2 + \beta_5 y_2^2.
$$
where the $\beta_i$ coefficients are unknown and must be estimated. Since in this case the above regression model is an approximation to another model, the regression is a 'model of a model' and so is called a meta-model. The parameters of the model are estimated by making runs at various input values for the $y_i$, recording the corresponding responses, and then using standard least-squares regression to estimate the coefficients. If the coefficients for the meta-model are found, then the fitted model gives a sense of where the best input-factor-combinations might be. An obvious difficulty on the use of meta-models is that they the responses, from which meta-models are constructed, are subject to variation.

### 7.3 Determine the best of several methods

In this section we describe a procedure whose goal is to select one out of several methods as being the best one, in some sense, and to control the probability that the selected method really is the best one.

Let $S_{ij}$ be the response of the $j$th replication using the $i$th method. It is assumed that the $S_{ij}$'s are independent of each other. The goal is to select the one with the largest expected response.

Now, there are two important issues that need to be discussed. First, since the $S_{ij}$'s are random variables, we can never be absolutely sure that we shall make the correct selection. Further, if the difference between the largest expected response and the second-largest expected response is small, we might not care if we choose the second-best method. The exact problem formulation, then, is that we want the probability of a correct selection larger than $p_{\Box}$ provided that the just mentioned difference is larger than $d^*$, where $p^*$ and $d^*$ can be specified.

The statistical procedure for solving this problem involves 'two-stage' sampling for each of the methods. First we make a fixed number of replications for each method and determine the sample means and variances. Then, we use the resulting variance estimates to determine how many more replications for each method are necessary in a second stage of sampling in order to reach a decision.

The procedure is proposed as follows. Initially make $\varphi(0) \geq 2$ replications using each of the methods and let $\mu^{(1)}_{S_i}$ and $\sigma^2_{S_i}$, respectively be the first-stage sample means and the corresponding variance estimates. Next, define $\varphi_i$ by

$$\varphi_i := \max \left\{ \varphi(0) + 1, \left[ \frac{q \sigma^2_{S_i}}{(d^*)^2} \right] \right\},$$

where $q$ is a constant that depends on the number of methods, $p^*$ and $\varphi(0)$, and make $\varphi_i - \varphi(0)$ additional observations using method $i$. The second-stage sample means $\mu^{(2)}_{S_i}$ are
obtained from these additional observations. Then define the weights $w_{S_i}^{(1)}$ and $w_{S_i}^{(2)}$ by

$$w_{S_i}^{(1)} := \frac{\varphi(0)}{\varphi_i} \left( 1 + \sqrt{1 - \frac{\varphi_i}{\varphi(0)} \left( 1 - \frac{\varphi_i - \varphi(0)}{q^2 \sigma_{S_i}^2} \right)} \right) \quad \text{and} \quad w_{S_i}^{(2)} := 1 - w_{S_i}^{(1)},$$

respectively, and define the weighted sample means by

$$\mu_{S_i}^{(1,2)} := w_{S_i}^{(1)} \mu_{S_i}^{(1)} + w_{S_i}^{(2)} \mu_{S_i}^{(2)}.$$

Finally, select the method with the smallest $\mu_{S_i}^{(1,2)}$. 
Chapter 8

Results

This chapter is divided in three main parts. In the first part, which runs from section one to section seven, we focus on the obtained results for the input side. In the first section we choose the parameters for the model and discuss the eight possible methods used in this work. In the next two sections we derive the optimal shape of the fitness function for each of the eight methods. Then, in section four, we select the best method. The accuracy of the selected method is dealt with in section five. In section six we use the selected method to obtain the optimal parameters for the input Markov chain. We conclude the first part of this chapter in section seven where we discuss the impact of the fitness weights and the matching problems. The second part runs from section eight to section fourteen. The first six sections are exactly of the same form as the first part of this chapter. Here we focus on the obtained results for the output side and derive the optimal parameters for the capacity Markov chain. In the last section we again derive the optimal capacity parameters, but under extra restrictions. In the third part (section fifteen) we combine the input side and the output side and shortly discuss the obtained model.

Input side

8.1 Choosing the parameters

In this section we choose the parameters of the model. We first discuss the parameters obtained from the measurements, then we choose the parameters for the GA and the SQP algorithm, and finally we choose the parameters for the fitness function. In the last subsection we discuss eight possible methods.

8.1.1 Parameters obtained from the measurements

The measurements used in this work are taken from the Mobihealth project described in [1], where the authors explore the ability of 3G networks to support demanding m-health services. In this project traces for different saturation factors are obtained. We focus on
traces obtained for the lowest saturation factor of 0.5, which corresponds to sending seven packets per second, because the observed network behaviour is stable for these traces. Further, the authors obtained both short and long traces. The short traces have a duration of forty seconds and are repeated thirty times. The long traces are a combination of the thirty (individual) short traces. As discovered in [8], the long trace exposes some abnormalities and is therefore useless. For this reason we focus on the short traces in this work.

Now, in order to derive the BRD and the ACF of the measurements, we use the strategy as described in [15]. The idea is to collect the contributions of packets observed during short averaging intervals over an observation window, and then calculate the corresponding throughput time series. From the throughput time series we obtain the bit rate statistics in which we are interested using well-known formulas.

The rates and the corresponding probabilities derived from the measurements are given by

\[ \tilde{\Omega}_{Rin} = \{6, 7\} \quad \text{and} \quad \tilde{\pi}_{Rin} = (0.07, 0.93), \]

respectively, and the ACF is given by

\[ \tilde{A}_{Rin} = (1, -0.38, -0.03, -0.01). \]

Independent of \( M_{Rin} \), we let the bounds \( b_{Rin, l} \) and \( b_{Rin, u} \) be given by \( b_{Rin, l} = 5 \) and \( b_{Rin, u} = 8 \).

### 8.1.2 Parameters for the GA and the SQP algorithm

The GA and the SQP algorithm are both called with fifteen iterations. An optimal solution is found if the difference between the obtained fitness and the maximum fitness is smaller than one, where the maximum fitness is set to one hundred. For the SQP algorithm we also stop if the number of function valuations exceeds \( 10 \cdot M_{Rin}^2 \).

#### 8.1.3 Parameters for the fitness function

Recall from subsection 5.2.1 that we obtained the total fitness of an arbitrary design as a weighted sum of fitnesses obtained for the rates, the corresponding probabilities and the ACF. Now, because we choose the same parameters for each of the three fitness functions, we make no distinction between them and discard \( R_{in}^{\text{in}}, \pi_{Rin} \) and \( A_{Rin} \) from the notation.

We want to choose the parameters in such a way that the obtained fitness function produces accurate optimal designs. The four parameters which have an influence on the shape of the fitness function are \( L_1, L_2, \Delta \) and \( p \). The parameters \( a \) and \( v \) can then be found from equations (5.2) and (5.3), respectively.
Now, since an investigation involving all four parameters becomes too time-consuming, we set $L_1$ and $L_2$ to reasonable values and drop them from consideration. Here, we let $L_1 = \frac{1}{1000}$ and $L_2 = \frac{1}{100}$. In the next two sections we discuss how to obtain the values $\Delta$ and $p$ using a $2^2$ factorial design and a three-dimensional response surface.

Further, we mainly consider uniformly distributed fitness weights. Only the weights $w_{A_{Rin}}$ are chosen differently, because we want to put more weight on the lower lags than on the higher lags. The following set of weights emphasise the lower lags:

$$w_{A_{Rin}} = \left( \frac{k_{\text{max}}}{v}, \frac{k_{\text{max}} - 1}{v}, \ldots, \frac{1}{v} \right) \quad \text{where} \quad v = \frac{k_{\text{max}} + 1}{2}.$$  

### 8.1.4 Several methods

The number of states for the input Markov chain ($M_{Rin}$) can be set to any integer larger than one. Now, because the number of parameters increases quadratically in $M_{Rin}$, a large $M_{Rin}$ value can lead to too long computations. In this work we compare the two cases in which $M_{Rin}$ either equals two or three.

Further, recall from subsections 5.4.1 and 6.2.2 that we discussed two methods for both the GA and the SQP algorithm. The constraints in the GA are satisfied using either a penalty method or a method where we modify the code. For the SQP we either use equality or inequality constraints. Altogether, this leaves us with the eight different methods illustrated in table 8.1.

<table>
<thead>
<tr>
<th>Method</th>
<th>$M_{Rin}$</th>
<th>GA method</th>
<th>SQP method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>penalty</td>
<td>inequality constraints</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>penalty</td>
<td>equality constraints</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>no penalty</td>
<td>inequality constraints</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>no penalty</td>
<td>equality constraints</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>penalty</td>
<td>inequality constraints</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>penalty</td>
<td>equality constraints</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>no penalty</td>
<td>inequality constraints</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>no penalty</td>
<td>equality constraints</td>
</tr>
</tbody>
</table>

### 8.2 Determination of $\Delta$ and $p$ from a $2^2$ factorial design

In this section we try to derive the optimal $\Delta$ and $p$ values for each of the eight methods in table 8.1 using a $2^2$ factorial design.
A $2^2$ factorial design for $\Delta$ and $p$ requires that we choose just two levels for each of the factors and then compute the average response at each of the $2^2$ possible factor-level combinations. Now, recall from subsection 5.2.1 that $\Delta$ and $p$ respectively control the slope of the linear and the reciprocal part in the fitness functions. A large negative $\Delta$ value leads to a steep linear part and a large positive $p$ value to a steep reciprocal part. Further, recall that the levels should be opposite in nature. The low and high levels we choose for these two factors are given in the coding chart in table 8.2.

Table 8.2: Coding chart for $\Delta$ and $p$.  

<table>
<thead>
<tr>
<th>Factor</th>
<th>-/+</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>-300</td>
</tr>
<tr>
<td>$p$</td>
<td>2</td>
</tr>
</tbody>
</table>

Now, we replicated the entire $2^2$ factorial design corresponding to the first method ten times and calculated the ten independent replicates of each of the responses $S_i$ as follows:

$$S_i = \tilde{R}^{in} + \tilde{\Pi}^{in} + \tilde{A}^{in},$$

where

$$\tilde{R}^{in} = \sum_{i=1}^{M_{Rin}} |\tilde{r}^{in}_i - r^{in}_i|,$$

$$\tilde{\Pi}^{in} = \sum_{i=1}^{M_{Rin}} |\tilde{\pi}^{Rin}_i - \pi^{Rin}_i|$$

and

$$\tilde{A}^{in} = \sum_{k=1}^{k_{\text{max}}} |\tilde{A}_{Rin}(k) - A_{Rin}(k)|.$$ 

The design matrix and corresponding average response variables are given in table 8.3 (top). From the average responses we derive the expected main effects using the equations from section 7.1. Furthermore, we formed approximate 95 percent confidence intervals for the expected effects using the standard normal distribution and the t-distribution with nine degrees of freedom. The expected main effects with corresponding confidence intervals are given in table 8.3 (bottom). Here, we denote the confidence intervals for effect $E_i$ based on the standard normal distribution and the t-distribution by $C_{E_i,n}$ and $C_{E_i,t}$, respectively.

Unfortunately, the confidence intervals corresponding to factor one ($\Delta$) contain the value zero, so we cannot conclude that the corresponding effect is real. Next, because $E_2$ is given by -0.0114, we expect the average response to decrease if we move $p$ from two to four. Hence, we expect more accurate results for large $p$ values. Because the confidence intervals for $p$ are close to zero, it is not clear if this effect is really present. Finally, since
\( E_3 \) is nearly zero, there appears to be no interaction.

We also tried to find \( \Delta \) and \( p \) values for the other seven methods using a \( 2^2 \) factorial design. Here we draw similar conclusions as for the first method. Because the influence of \( \Delta \) and \( p \) is not clear, we use a different approach to obtain the \( \Delta \) and \( p \) values, which will be discussed in the next section.

Table 8.3: Design matrix and expected responses for a \( 2^2 \) factorial design on \( \Delta \) and \( p \) (top). Expected main effects with corresponding confidence intervals (bottom).

<table>
<thead>
<tr>
<th>Factor combination ( i )</th>
<th>( \Delta )</th>
<th>( p )</th>
<th>( S_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-300</td>
<td>2</td>
<td>0.4830</td>
</tr>
<tr>
<td>2</td>
<td>-100</td>
<td>2</td>
<td>0.4835</td>
</tr>
<tr>
<td>3</td>
<td>-300</td>
<td>4</td>
<td>0.4589</td>
</tr>
<tr>
<td>4</td>
<td>-100</td>
<td>4</td>
<td>0.4621</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \mathcal{E}_i )</th>
<th>( C_{\mathcal{E}_{i,n}} )</th>
<th>( C_{\mathcal{E}_{i,t}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i = 1 )</td>
<td>0.0009</td>
<td>[-0.0003, 0.0021]</td>
<td>[-0.0001, 0.0018]</td>
</tr>
<tr>
<td>( i = 2 )</td>
<td>-0.0114</td>
<td>[-0.0120, -0.0108]</td>
<td>[-0.0119, -0.0109]</td>
</tr>
<tr>
<td>( i = 3 )</td>
<td>0.0008</td>
<td>[0.0001, 0.0011]</td>
<td>[0.0001, 0.0012]</td>
</tr>
</tbody>
</table>

### 8.3 Determination of \( \Delta \) and \( p \) from a response surface

In this section we try to derive the optimal \( \Delta \) and \( p \) values for each of the eight methods in table 8.1 using meta-modeling.

The parameters of the meta-model are estimated by making multiple runs at various input values for \( \Delta \) and \( p \), recording the corresponding responses, and using standard least squares regression to estimate the coefficients. Again, we first focus on method one. Here, we made five replications at the sixteen combinations of \( \Delta = -100, -200, -300 \) and \( -400 \) and \( p = 1, 2, 3 \) and 4. This required a total of eighty runs of the model. Fitting a full quadratic model to these eighty points leads to the fitted meta-model

\[
y = 0.5528 + 0.0001\Delta - 0.0921p + 0.0006\Delta p + 0.0362p^2,
\]

which is shown in figure 8.1. This figure clearly shows that large negative \( \Delta \) values and large positive \( p \) values lead to the best results. So, it appears that the slopes of the linear and the reciprocal part should be chosen as steep as possible. This is probably due to the fact that the GA and the SQP algorithm find it easier to produce designs with an higher fitness value from one generation to the next if the slopes in the fitness function are steep.
Figure 8.1: Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the first method.

For the other seven methods we created a fitted meta-model using the same sixteen combinations of $\Delta$ and $p$. Although not all fitted meta-models are as pronounced as the one shown in figure 8.1 for the first method, there is a tendency that higher $\Delta$ and $p$ values lead to more accurate results. For instance, the fitted meta-model for method five is given by

$$y = 1.5078 + 0.0005\Delta - 0.4118p + 0.0001\Delta p + 0.0680p^2,$$

and is nearly of the same form as the meta-model obtained for the first method (see figure 8.2). For this reason we use the same fitness function for all of the eight methods, with slopes of the linear and reciprocal part given by $\Delta = -400$ and $p = 4$, respectively.

Figure 8.2: Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the fifth method.
8.4 Determination of the best method

Now that we obtained the shape of the fitness function for each of the eight methods, we determine the best method on the basis of their corresponding expected responses.

Our goal is to select the method with the highest response and to be 90 percent sure that we have made the correct selection. We made \( \varphi(0) = 20 \) initial independent replications of each method and the results of the first-stage sampling are given in the first two columns of table 8.4. Then we made \( \varphi_i = 20 \) additional replications for each method and computed the second stage sample means, as shown. Finally, we calculated the weights for each method and the weighted sample means. Since \( \mu_{S_2}^{(1,2)} \) is the largest weighted sample mean, we select method two as the best method.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \mu_{S_1}^{(1)} )</th>
<th>( \sigma_{S_1}^2 )</th>
<th>( \varphi_i )</th>
<th>( \mu_{S_1}^{(2)} )</th>
<th>( w_{S_1}^{(1)} )</th>
<th>( w_{S_1}^{(2)} )</th>
<th>( \mu_{S_1}^{(1,2)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69.84</td>
<td>3.53</td>
<td>21</td>
<td>70.26</td>
<td>1.78</td>
<td>-0.78</td>
<td>69.51</td>
</tr>
<tr>
<td>2</td>
<td><strong>69.18</strong></td>
<td><strong>9.39</strong></td>
<td>21</td>
<td><strong>58.89</strong></td>
<td><strong>1.43</strong></td>
<td><strong>-0.43</strong></td>
<td><strong>73.60</strong></td>
</tr>
<tr>
<td>3</td>
<td>69.84</td>
<td>3.53</td>
<td>21</td>
<td>70.26</td>
<td>1.78</td>
<td>-0.78</td>
<td>69.51</td>
</tr>
<tr>
<td>4</td>
<td>69.57</td>
<td>6.48</td>
<td>21</td>
<td>70.23</td>
<td>1.54</td>
<td>-0.54</td>
<td>69.21</td>
</tr>
<tr>
<td>5</td>
<td>64.63</td>
<td>98.10</td>
<td>37</td>
<td>62.61</td>
<td>0.60</td>
<td>0.40</td>
<td>63.82</td>
</tr>
<tr>
<td>6</td>
<td>24.72</td>
<td>94.91</td>
<td>36</td>
<td>29.92</td>
<td>0.62</td>
<td>0.38</td>
<td>26.68</td>
</tr>
<tr>
<td>7</td>
<td>70.02</td>
<td>10.91</td>
<td>21</td>
<td>71.34</td>
<td>1.39</td>
<td>-0.39</td>
<td>69.51</td>
</tr>
<tr>
<td>8</td>
<td>32.21</td>
<td>72.64</td>
<td>28</td>
<td>40.72</td>
<td>0.80</td>
<td>0.20</td>
<td>33.92</td>
</tr>
</tbody>
</table>

8.5 Accuracy of the obtained method

Now that we obtained the best method to match the model to the measurements, we check the accuracy of this method. In order to check the accuracy we use the following strategy.

We first choose arbitrary values for the state space \( \Omega_{Rin} \) and the transition matrix \( P_{Rin} \), and calculate the corresponding BRD and ACF. Next, we pretend that these two bit rate statistics where derived from the measurements, i.e. \( \bar{\Omega}_{Rin} := \Omega_{Rin} \), \( \bar{\pi}_{Rin} := \pi_{Rin} \) and \( \bar{A}_{Rin} := A_{Rin} \). Then, we check if the GA and the SQP algorithm are capable of reproducing the state space \( \Omega_{Rin} \) and the transition matrix \( P_{Rin} \).

We discuss two examples, for which the state spaces \( \Omega_{Rin} \) are both given by \( \{6, 7\} \) and the transition matrices \( P_{Rin} \) by

\[
\begin{pmatrix}
0.01 & 0.99 \\
0.90 & 0.10
\end{pmatrix}
\quad \text{and} \quad
\begin{pmatrix}
0.60 & 0.40 \\
0.11 & 0.89
\end{pmatrix}.
\]
Now, the steady-state probabilities of the two examples are given by 

\[(0.48, 0.52) \quad \text{and} \quad (0.22, 0.78),\]

respectively, and the ACFs by 

\[(1, -0.89, 0.79, -0.71) \quad \text{and} \quad (1, 0.49, 0.24, 0.12).\]

For both examples we made ten runs, and the runs with the highest fitness are illustrated in table 8.5. Comparing the values in the table with the ones displayed above, it is clearly visible that the combination of the two algorithms leads to extremely accurate results.

Table 8.5: The obtained state spaces \(\Omega_R\), steady-state probabilities \(\pi_R\), ACFs \(A_R\), transition matrices \(P_R\), and fitness values \(F(R,\pi_R,A_R)\) for the two examples discussed in this section.

<table>
<thead>
<tr>
<th>(\Omega_R)</th>
<th>(\pi_R)</th>
<th>(A_R)</th>
<th>(P_R)</th>
<th>(F(R,\pi_R,A_R))</th>
</tr>
</thead>
<tbody>
<tr>
<td>({6.00, 7.00})</td>
<td>(0.48, 0.52)</td>
<td>(1, -0.89, 0.79, -0.70)</td>
<td>(\begin{pmatrix} 0.01 &amp; 0.99 \ 0.90 &amp; 0.10 \end{pmatrix})</td>
<td>100</td>
</tr>
<tr>
<td>({6.00, 7.00})</td>
<td>(0.2, 0.8)</td>
<td>(1, 0.49, 0.24, 0.12)</td>
<td>(\begin{pmatrix} 0.60 &amp; 0.40 \ 0.10 &amp; 0.90 \end{pmatrix})</td>
<td>100</td>
</tr>
</tbody>
</table>

### 8.6 Determination of the optimal input parameters

In this section we obtain the state space \(\Omega_R\) and the transition matrix \(P_R\) for which the model matches the measurements as closely as possible.

The algorithms ran ten times. In table 8.6 we list the fitness values obtained for the GA and the SQP algorithm for each of the ten runs. Run five produces the best result and the corresponding state space and transition matrix are given by 

\[\{6, 7\} \quad \text{and} \quad \begin{pmatrix} 0.03 & 0.97 \\ 0.07 & 0.93 \end{pmatrix},\]

respectively. Figure 8.3 illustrates the behaviour of the GA and the SQP for the best run. It is clearly visible from this figure that the SQP greatly enhances the performance of the GA.

The state space and the transition matrix in the previous display are used as input parameters in the second part of this chapter, where we derive the optimal capacity parameters.
Table 8.6: Ten runs for the GA and the SQP algorithm.

<table>
<thead>
<tr>
<th>Run</th>
<th>Fitness GA</th>
<th>Fitness SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59.9</td>
<td>69.4</td>
</tr>
<tr>
<td>2</td>
<td>69.9</td>
<td>70.1</td>
</tr>
<tr>
<td>3</td>
<td>63.8</td>
<td>70.1</td>
</tr>
<tr>
<td>4</td>
<td>56.8</td>
<td>65.3</td>
</tr>
<tr>
<td>5</td>
<td>65.9</td>
<td>70.3</td>
</tr>
<tr>
<td>6</td>
<td>62.3</td>
<td>65.8</td>
</tr>
<tr>
<td>7</td>
<td>64.5</td>
<td>70.2</td>
</tr>
<tr>
<td>8</td>
<td>69.8</td>
<td>70.1</td>
</tr>
<tr>
<td>9</td>
<td>65.7</td>
<td>67.3</td>
</tr>
<tr>
<td>10</td>
<td>66.8</td>
<td>70.2</td>
</tr>
</tbody>
</table>

Figure 8.3: Performance of the GA and the SQP algorithm for the best run.

8.7 Matching in practice

In this section we first discuss the influence of the weight $w_{(R_{in},R_{in},A_{in})}$, and after that we describe the matching problems.

8.7.1 Influence of the weights

Recall from subsection 8.1.3 that we mainly used uniformly fitness weights. In order to examine the influence of the weights, we shortly discuss cases for which the weights are
not uniformly distributed. Now, because we do not wish to give preferential treatment to any rate or steady-state probability, we do not modify \( w_R \). Here, we only focus on the fitness weight \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \), which controls the influence of \( F^{R^{\text{in}}, \pi^{\text{in}}} \) in the weighted sum of \( F^{R^{\text{in}}, \pi^{\text{in}}} \) and \( F^{A^{\text{in}}} \). Clearly, we expect \( F^{R^{\text{in}}, \pi^{\text{in}}} \) to increase (decrease) if \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) increases (decreases), and we expect the opposite behaviour for \( F^{A^{\text{in}}} \).

In the following example we again determine the optimal input parameters for several \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) values.

**Example** Because a \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) value of 0.5 already leads to a perfect match for the BRD \( (F^{R^{\text{in}}, \pi^{\text{in}}} = 100) \), increasing \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) will only lead to less accurate results. For this reason, we only discuss cases for which \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \geq 0.5 \).

In table 8.7 we supplement the case for which \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} = 0.5 \) with the two cases for which \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) is given by 0.65 and 0.75, respectively. First, a \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) value of 0.65 leads to exactly the same fitness values as a \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) value of 0.5. Next, if we increase \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) to 0.75, then \( F^{A^{\text{in}}} \) increases from 40.5 to 66.8, and \( F^{R^{\text{in}}, \pi^{\text{in}}} \) decreases dramatically. The steady-state probability for the model, \( \pi^{\text{in}} = (0.71, 0.29) \), differs significantly from the steady-state probability obtained from the measurements, \( \tilde{\pi}^{\text{in}} = (0.07, 0.93) \), which is unacceptable.

**Table 8.7: Influence of the weight \( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} \) on the fitness values \( F^{R^{\text{in}}, \pi^{\text{in}}} \) and \( F^{A^{\text{in}}} \).**

<table>
<thead>
<tr>
<th>( w(R^{\text{in}}, \pi^{\text{in}}), A^{\text{in}} )</th>
<th>( R^{\text{in}} )</th>
<th>( \pi^{\text{in}} )</th>
<th>( A^{\text{in}} )</th>
<th>( F^{R^{\text{in}}, \pi^{\text{in}}} )</th>
<th>( F^{A^{\text{in}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>{6,7}</td>
<td>(0.07, 0.93)</td>
<td>(1, −0.04, 0, 0)</td>
<td>100.0</td>
<td>40.5</td>
</tr>
<tr>
<td>0.65</td>
<td>{6,7}</td>
<td>(0.07, 0.93)</td>
<td>(1, −0.03, 0, 0)</td>
<td>100.0</td>
<td>40.5</td>
</tr>
<tr>
<td>0.75</td>
<td>{6,7}</td>
<td>(0.71, 0.29)</td>
<td>(1, −0.38, 0.15, −0.06)</td>
<td>50.6</td>
<td>66.8</td>
</tr>
</tbody>
</table>

8.7.2 Matching problems

The SQP method solves a quadratic programming (QP) subproblem at each iteration. The solution of the QP subproblem is used to form a search direction for a line search procedure. In order to determine this search direction, the Cholesky factor needs to be calculated (if it exists), and this amounts to computing the inverse of a square matrix.
Matching procedure for a discrete fluid flow model

During this computation we sometimes get the following error messages:

**Matrix is close to singular or badly scaled. RCOND=4.103e-021.**

Now, recall that a square matrix is singular if its determinant is zero, so the matrix is not invertible. This means that the computed inverse is unlikely to be of much use and the results are inaccurate. If the error message goes away, then it may not cause any problems.

This error message can be due to a variety of things, but in most cases its due to one of the following reasons:

- The fitness function is very sensitive to one variable, but almost completely insensitive to another.
- Extraneous independent degree of freedom.

Since we did not find a solution to this problem, we just discarded the values obtained with error messages and made some extra replications.

### Output side

#### 8.8 Choosing the parameters

##### 8.8.1 Parameters obtained from the measurements

The rates and the corresponding probabilities derived from the measurements are given by

\[
\tilde{\Omega}_{\text{out}} = \{6, 7\} \quad \text{and} \quad \tilde{\pi}_{\text{out}} = (0.133, 0.867),
\]

respectively, and the ACF is given by

\[
\tilde{A}_{\text{out}} = (1, -0.035, -0.037, -0.038).
\]

We let the bounds \(b_{\text{out},l}\) and \(b_{\text{out},u}\) be given by \(b_{\text{out},l} = 5\) and \(b_{\text{out},u} = 9\), respectively.

Further, we should choose a reasonable value for the buffersize \(K\). Too small \(K\) values lead to too little variation in the buffer level, whereas too big values lead to a long computation time and hence to inaccurate results. We recommend setting \(K\) to

\[
K := 4 \cdot [\max\{\tilde{r}_i^{\text{in}} : \tilde{r}_i^{\text{in}} \in \tilde{\Omega}_{\text{out}}\} - \min\{\tilde{r}_i^{\text{out}} : \tilde{r}_i^{\text{out}} \in \tilde{\Omega}_{\text{out}}\}].
\]

Since, for the MobiHealth traces, the maximum and the minimum in the previous display are given by seven and six, respectively, we set \(K\) to four.
8.8.2 Remaining parameters
At the output side, the GA and the SQP algorithm are both called with ten iterations.
The SQP algorithm is stopped if the number of function evaluations exceeds \(10 \cdot M_C^2\). All
the other parameters are set to the same values as in section 8.1.

8.8.3 Several methods
In subsection 8.1.4 we discussed eight possible methods for matching the input side, in-
volving the number of states for the input Markov chain \(M_{R_{in}}\) and the constraints for
both the GA and the SQP algorithm. At the output side we follow a similar approach,
again with eight possible methods, but we replace the value \(M_{R_{in}}\) with the value \(M_C\). This
means that we compare the two cases in which \(M_C\) either equals two or three.

8.9 Determination of \(\Delta\) and \(p\)
In this section we obtain the optimal \(\Delta\) and \(p\) values for each of the eight methods at the
output side using both a 2\(^2\) factorial design and meta-modeling.

8.9.1 Determination of \(\Delta\) and \(p\) from a 2\(^2\) factorial design
The low and the high values for the factors \(\Delta\) and \(p\) are again chosen as in table 8.3, and
we replicate the entire 2\(^2\) factorial design corresponding to the first method ten times. The
expected responses and the expected main effects with corresponding confidence intervals
are illustrated in table 8.8.

Table 8.8: Design matrix and expected responses for a 2\(^2\) factorial design on \(\Delta\) and \(p\) (top).
Expected main effects with corresponding confidence intervals (bottom).

<table>
<thead>
<tr>
<th>Factor combination (i)</th>
<th>(\Delta)</th>
<th>(p)</th>
<th>(S_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-300</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>-100</td>
<td>2</td>
<td>0.34</td>
</tr>
<tr>
<td>3</td>
<td>-300</td>
<td>4</td>
<td>0.59</td>
</tr>
<tr>
<td>4</td>
<td>-100</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(\mathcal{E}_i)</th>
<th>(C_{\mathcal{E}_{i,n}})</th>
<th>(C_{\mathcal{E}_{i,t}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i = 1)</td>
<td>-0.06 [-0.19,0.07]</td>
<td>[-0.12, 0.01]</td>
</tr>
<tr>
<td>(i = 2)</td>
<td>0.06 [-0.07,0.19]</td>
<td>[ 0.01,-0.12]</td>
</tr>
<tr>
<td>(i = 3)</td>
<td>-0.11 [-0.40,0.19]</td>
<td>[-0.24, 0.03]</td>
</tr>
</tbody>
</table>

Here, most confidence intervals for the average effects contain the value zero, so we cannot
conclude that the effects are real. Again, we looked at the other seven methods, and here
we draw similar conclusions. For this reason we derive the optimal $\Delta$ and $p$ values using meta-modeling.

### 8.9.2 Determination of $\Delta$ and $p$ from a response surface

For the first method we made five replications at the same sixteen combinations for $\Delta$ and $p$ as discussed in section 8.3. From these eighty points we derived the fitted meta-model

$$y = 0.75 + 0.0002\Delta - 0.25p + 0.001\Delta p + 0.0056p^2,$$

which is shown in figure 8.4.

![Figure 8.4: Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the first method.](image)

Similarly as for the input side, we derive from the picture that steep slopes for the linear part and the reciprocal part lead to the best numerical results. Again, we set $\Delta$ and $p$ to -400 and 4, respectively. The meta-models obtained for the other seven methods are of the same form as the one displayed above. For instance, the meta-model obtained for the fifth method is given by

$$y = 1.32 + 0.0002\Delta - 0.31p + 0.002\Delta p + 0.0068p^2,$$

which is shown in figure 8.5. Hence, we again use the same fitness function for all the eight methods.
Figure 8.5: Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the fifth method.

8.10 Determination of the best method

Now that we determined the shape of the fitness function for the eight methods, we select one out of eight methods as being the best. Now, we used exactly the same approach as discussed in section 8.4, and the results are illustrated in table 8.9. Since $\mu_{S_i}^{(1,2)}$ is maximal for $i = 2$, we again select method two as the best method.

Table 8.9: Selecting the best of the 8 possible methods.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\mu_{S_i}^{(1)}$</th>
<th>$\sigma_{S_i}^2$</th>
<th>$\varphi_i$</th>
<th>$\mu_{S_i}^{(2)}$</th>
<th>$w_{S_i}^{(1)}$</th>
<th>$w_{S_i}^{(2)}$</th>
<th>$\mu_{S_i}^{(1,2)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75.54</td>
<td>78.42</td>
<td>30</td>
<td>73.40</td>
<td>0.74</td>
<td>-0.26</td>
<td>74.99</td>
</tr>
<tr>
<td>2</td>
<td><strong>78.91</strong></td>
<td><strong>1.37</strong></td>
<td>21</td>
<td><strong>79.12</strong></td>
<td>2.30</td>
<td>-1.30</td>
<td>78.64</td>
</tr>
<tr>
<td>3</td>
<td>72.16</td>
<td>50.99</td>
<td>21</td>
<td>79.12</td>
<td>1.02</td>
<td>-0.02</td>
<td>72.01</td>
</tr>
<tr>
<td>4</td>
<td>57.00</td>
<td>9.47</td>
<td>21</td>
<td>79.12</td>
<td>1.43</td>
<td>-0.43</td>
<td>47.57</td>
</tr>
<tr>
<td>5</td>
<td>67.50</td>
<td>6.58</td>
<td>21</td>
<td>79.12</td>
<td>1.54</td>
<td>-0.54</td>
<td>61.24</td>
</tr>
<tr>
<td>6</td>
<td>69.16</td>
<td>16.51</td>
<td>21</td>
<td>71.02</td>
<td>1.28</td>
<td>-0.28</td>
<td>68.63</td>
</tr>
<tr>
<td>7</td>
<td>67.50</td>
<td>59.21</td>
<td>23</td>
<td>79.12</td>
<td>0.94</td>
<td>0.06</td>
<td>68.20</td>
</tr>
<tr>
<td>8</td>
<td>46.73</td>
<td>19.24</td>
<td>21</td>
<td>79.12</td>
<td>1.25</td>
<td>-0.25</td>
<td>38.68</td>
</tr>
</tbody>
</table>

8.11 Accuracy of the obtained method

In this section we check the accuracy of the selected method ($M_C = 2$, penalty method for the GA and equality constraints for the SQP algorithm). In order to do this, we use a similar strategy as discussed in section 8.5 for the input side. So, we first choose $\Omega_C$
and $P_C$ and compute the BRD and the ACF at the output side using the optimal input parameters derived in section 8.6. Then, we set $\Omega_{Rout}$, $\pi_{Rout}$ and $A_{Rout}$ to $\Omega_{Rout}$, $\pi_{Rout}$ and $A_{Rout}$, respectively, and check if the GA and the SQP algorithm are capable of reproducing the state space $\Omega_C$ and the transition matrix $P_C$.

Again, we discuss two examples, for which the state spaces $\Omega_C$ are both given by \{5, 8\} and the transition matrices $P_C$ by

\[
\begin{pmatrix}
0.2 & 0.8 \\
0.8 & 0.2
\end{pmatrix}
\text{ and }\begin{pmatrix}
0.6 & 0.4 \\
0.83 & 0.17
\end{pmatrix}.
\]

Now, the state spaces $\Omega_{Rout}$ for the two examples are given by \{5, 8\} and \{5, 7, 8, 9\}, respectively, the corresponding steady-state probabilities $\pi_{Rout}$ by

\[
(0.5, 0.5) \text{ and } (0.68, 0.01, 0.01, 0.30),
\]

and the ACFs $A_{Rout}$ by

\[
(1, -0.60, 0.36, -0.22) \text{ and } (1, -0.23, 0.05, -0.01).
\]

For both examples we made ten runs, and the runs with the highest fitness are illustrated in table 8.10. Comparing the values in the table with the ones displayed above, it is clearly visible that the combination of the two algorithms leads to accurate results.

Table 8.10: The obtained state spaces $\Omega_{Rout}$, steady-state probabilities $\pi_{Rout}$, ACFs $A_{Rout}$, state spaces $\Omega_C$, transition matrices $P_C$, and fitness values $F(R_{out}, \pi_{Rout}, A_{Rout})$ for the two examples discussed in this section.

<table>
<thead>
<tr>
<th>$\Omega_{Rout}$</th>
<th>$\pi_{Rout}$</th>
<th>$A_{Rout}$</th>
<th>$\Omega_C$</th>
<th>$P_C$</th>
<th>$F(R_{out}, \pi_{Rout}, A_{Rout})$</th>
</tr>
</thead>
</table>
| \{5.0, 7.5\}   | (0.5, 0.5)  | (1, -0.60, 0.36, -0.22) | \{5.0, 7.5\} | \begin{pmatrix}
0.2 & 0.8 \\
0.8 & 0.2
\end{pmatrix} | 87.6          |
| \{5.9\}        | (0.70, 0.30)| (1, -0.24, 0.03, -0.02) | \{5.0, 9\}  | \begin{pmatrix}
0.61 & 0.39 \\
0.81 & 0.19
\end{pmatrix} | 97.3          |

### 8.12 Determination of the optimal capacity parameters

In this section we obtain the state space $\Omega_C$ and the transition matrix $P_C$ for which the model matches the measurements as closely as possible.
The algorithms ran ten times and the obtained fitness values are illustrated in table 8.11. The highest fitness is obtained in run two, in which the state space $\Omega_C$ and the transition matrix $P_C$ are given by

$$\{8.5, 8.8\} \quad \text{and} \quad \begin{pmatrix} 0.77 & 0.23 \\ 0.81 & 0.19 \end{pmatrix},$$

respectively. Figure 8.6 illustrates the behaviour of the GA and the SQP for the best run. As was the case in section 8.6, we derive from the figure that the SQP algorithm improves the fitness value obtained from the GA.

Note that the GA and the SQP algorithm are both called with five iterations instead of ten. This is because the obtained results are exactly the same for these two cases. The reason for this is as follows. Recall from subsection 8.1.1 that we focus on traces for which the observed network behaviour is stable. The BRD and the ACF at the input side differ slightly from the corresponding statistics at the output side. Matching the model to these observations, it comes as no surprise that we obtain a model for which the buffer remains empty with probability one, i.e. a model for which $r_{in}^i \leq c_j \forall i \in \{1, \ldots, M_{Rin}\} \forall j \in \{1, \ldots, M_C\}$. For most runs this result is already obtained after a few iterations.

In order to come to a more realistic model, we determine the optimal capacity parameters under extra restrictions in section 8.14.

### Table 8.11: Ten runs for the GA and the SQP algorithm.

<table>
<thead>
<tr>
<th>Run</th>
<th>Fitness GA</th>
<th>Fitness SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>57.1</td>
<td>66.9</td>
</tr>
<tr>
<td>2</td>
<td>71.0</td>
<td>79.1</td>
</tr>
<tr>
<td>3</td>
<td>55.9</td>
<td>70.1</td>
</tr>
<tr>
<td>4</td>
<td>76.3</td>
<td>79.1</td>
</tr>
<tr>
<td>5</td>
<td>46.7</td>
<td>60.0</td>
</tr>
<tr>
<td>6</td>
<td>79.1</td>
<td>79.1</td>
</tr>
<tr>
<td>6</td>
<td>62.3</td>
<td>65.8</td>
</tr>
<tr>
<td>7</td>
<td>79.1</td>
<td>79.1</td>
</tr>
<tr>
<td>8</td>
<td>79.1</td>
<td>79.1</td>
</tr>
<tr>
<td>9</td>
<td>65.7</td>
<td>67.3</td>
</tr>
<tr>
<td>10</td>
<td>79.1</td>
<td>79.1</td>
</tr>
</tbody>
</table>
Matching in practice

In this section we first discuss the influence of the weight $w(R_{out}, \pi_{R_{out}}, A_{R_{out}})$, and after that we describe the matching problems.

8.13.1 Influence of the weights

The results at the output side are obtained using uniform fitness weights. Similarly as in subsection 8.7.1, we discuss an example in which we obtain the optimal capacity parameters for several $w(R_{out}, \pi_{R_{out}}, A_{R_{out}})$ values.

**Example** In table 8.12 we supplement the case for which $w(R_{out}, \pi_{R_{out}}, A_{R_{out}})$ equals 0.5 with the five cases for which $w(R_{out}, \pi_{R_{out}}, A_{R_{out}})$ is given by 0.25, 0.35, 0.65 and 0.75, respectively.

As is visible from the table (top), all the five cases lead to the same state space $\Omega_{R_{out}}$, steady-state probability $\pi_{R_{out}}$ and ACF $A_{R_{out}}$. Further, the fitness values $F_{R_{out}, \pi_{R_{out}}}$ and $F_{A_{R_{out}}}$ are all the same, which means that the total fitness values $F(R_{out}, \pi_{R_{out}}, A_{R_{out}})$ are nearly the same (bottom). The reason for this is as follows. The model which matches best to the observations is the model for which the buffer remains empty. Because this result is easy to obtain for all five cases, we end-up with similar results.
Table 8.12: Influence of the weight \( w_{(\text{Rout}, \pi_{\text{Rout}}), A_{\text{Rout}}} \) on the state space \( \Omega_{\text{Rout}} \), the steady-state probability \( \pi_{\text{Rout}} \) and the ACF \( A_{\text{Rout}} \) (top), and on the state space \( \Omega_C \), the transition matrix \( P_C \) and the fitness values \( F_{\text{Rout}, \pi_{\text{Rout}}}, F_{A_{\text{Rout}}} \) and \( F_{(\text{Rout}, \pi_{\text{Rout}}), A_{\text{Rout}}} \) (bottom).

<table>
<thead>
<tr>
<th>( w_{(\text{Rout}, \pi_{\text{Rout}}), A_{\text{Rout}}} )</th>
<th>( \Omega_{\text{Rout}} )</th>
<th>( \pi_{\text{Rout}} )</th>
<th>( A_{\text{Rout}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>{6, 7}</td>
<td>(0.07, 0.93)</td>
<td>(1, -0.040, 0.002, 0)</td>
</tr>
<tr>
<td>0.35</td>
<td>{6, 7}</td>
<td>(0.07, 0.93)</td>
<td>(1, -0.040, 0.002, 0)</td>
</tr>
<tr>
<td>0.50</td>
<td>{6, 7}</td>
<td>(0.07, 0.93)</td>
<td>(1, -0.040, 0.002, 0)</td>
</tr>
<tr>
<td>0.65</td>
<td>{6, 7}</td>
<td>(0.07, 0.93)</td>
<td>(1, -0.040, 0.002, 0)</td>
</tr>
<tr>
<td>0.75</td>
<td>{6, 7}</td>
<td>(0.07, 0.93)</td>
<td>(1, -0.040, 0.002, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \Omega_C )</th>
<th>( P_C )</th>
<th>( F_{\text{Rout}, \pi_{\text{Rout}}} )</th>
<th>( F_{A_{\text{Rout}}} )</th>
<th>( F_{(\text{Rout}, \pi_{\text{Rout}}), A_{\text{Rout}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{7.7, 8.8}</td>
<td>\begin{pmatrix} 0.60 &amp; 0.40 \ 0.01 &amp; 0.99 \end{pmatrix}</td>
<td>75.20</td>
<td>83.00</td>
<td>81.10</td>
</tr>
<tr>
<td>{7.5, 8.4}</td>
<td>\begin{pmatrix} 0.42 &amp; 0.58 \ 0.85 &amp; 0.15 \end{pmatrix}</td>
<td>75.20</td>
<td>83.00</td>
<td>80.30</td>
</tr>
<tr>
<td>{8.5, 8.8}</td>
<td>\begin{pmatrix} 0.77 &amp; 0.23 \ 0.81 &amp; 0.19 \end{pmatrix}</td>
<td>75.20</td>
<td>83.00</td>
<td>79.10</td>
</tr>
<tr>
<td>{7.6, 8.2}</td>
<td>\begin{pmatrix} 0.04 &amp; 0.96 \ 0.99 &amp; 0.01 \end{pmatrix}</td>
<td>75.20</td>
<td>83.00</td>
<td>78.00</td>
</tr>
<tr>
<td>{8.0, 8.1}</td>
<td>\begin{pmatrix} 0.69 &amp; 0.31 \ 0.80 &amp; 0.20 \end{pmatrix}</td>
<td>75.20</td>
<td>83.00</td>
<td>77.20</td>
</tr>
</tbody>
</table>

8.13.2 Matching problems

In subsection 8.7.2 we discussed the matching problems associated with the SQP algorithm while matching the input side. Fortunately, we did not get error messages at the output.
side, so no extra replications were needed.

8.14 Determination of the optimal capacity parameters under extra restrictions

In sections 8.6 and 8.12 we determined the optimal input and capacity parameters for the model. Unfortunately, $r_i^{\text{in}} < c_j \forall i \in \{1, \ldots, M_R\} \forall j \in \{1, \ldots, M_C\}$, which means that the buffer remains empty for the obtained model. Because this model is not very interesting, we put extra restrictions on the model in order to come to a more realistic result. In this section we determine the optimal capacity parameters under these extra restrictions. We first describe the extra restrictions. After that, we make five runs and choose the capacity parameters corresponding to the run with the highest fitness value.

We discuss two extra restrictions. The first restriction eliminates the cases for which the buffer remains empty. This means that

$$\min \{c_j : c_j \in \Omega_C\} < \max \{r_i^{\text{in}} : r_i^{\text{in}} \in \Omega_R\},$$

so the results obtained in section 8.12 not apply here. The second restriction eliminates the opposite cases in which the buffer remains full, and is of the form

$$\max \{c_j : c_j \in \Omega_C\} > \min \{r_i^{\text{in}} : r_i^{\text{in}} \in \Omega_R\}.$$

With these extra restrictions we made five runs using method two, and the fitness values are summarized in table 8.13. Clearly, the obtained fitness values are smaller than the ones illustrated in table 8.11, which are the values obtained without the extra restrictions. The highest fitness is obtained in run four, in which the state space $\Omega_C$ and the transition matrix $P_C$ are given by

$$\{6, 2, 8.7\} \text{ and } \begin{pmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \end{pmatrix},$$

respectively. Because the optimal state space for the input Markov chain is given by \{6, 7\}, both of the extra restrictions are satisfied. Figure 8.7 illustrates the behaviour of the GA and the SQP for the best run.

Table 8.13: Five runs for the GA and the SQP algorithm under the extra restrictions.

<table>
<thead>
<tr>
<th>Run</th>
<th>fitness GA</th>
<th>fitness SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>53.1</td>
<td>61.9</td>
</tr>
<tr>
<td>2</td>
<td>56.3</td>
<td>60.1</td>
</tr>
<tr>
<td>3</td>
<td>55.9</td>
<td>60.1</td>
</tr>
<tr>
<td>4</td>
<td><strong>57.5</strong></td>
<td><strong>63.9</strong></td>
</tr>
<tr>
<td>5</td>
<td>46.7</td>
<td>53.0</td>
</tr>
</tbody>
</table>
Combined input side and output side

8.15 The obtained model

In this section we combine the input side and the output side and shortly discuss the obtained model.

Recall from sections 8.6 and 8.14 that the optimal input and capacity parameters are given by

$$\{6, 7\} \text{ and } \begin{pmatrix} 0.03 & 0.97 \\ 0.07 & 0.93 \end{pmatrix},$$

and

$$\{6.2, 8.7\} \text{ and } \begin{pmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \end{pmatrix},$$

respectively. With these values the obtained discrete fluid flow model can be summarized as in figure 8.8.

The BRD at the input side and at the output side with corresponding throughput HDP for the obtained model, the corresponding ACFs at the input side and at the output
side and the corresponding buffer content are illustrated in figure 8.9 (top and middle). Here, the impact of the model is clearly visible. The throughput histogram plot at the input side is much more compact than the one at the output side, and the autocorrelation structure is also changed significantly by the model. Changes of the ACF from input side to output side reflect changes of after-effects and periodicities within the throughput process imposed by the network.

Finally, recall from section 2.5 that we are interested in the loss probability per interval $p_{\text{loss}}$ and the average amount of loss per interval $L$. Figure 8.9 (bottom) displays the different amount of losses per interval with the corresponding probabilities. These values lead to a $p_{\text{loss}}$ value of 0.06 and a $L$ value of 0.05.
Figure 8.8: The obtained discrete fluid flow model.
Figure 8.9: BRDs at the input side and at the output side with corresponding throughput HDP for the obtained model (top left and right). Corresponding ACFs at the input side and at the output side (middle left). Corresponding buffer content (middle right). Loss probability per interval and average amount of loss (bottom).
Chapter 9

Outlook

There are three important issues that are left as future work. First of all, recall from the preface that we created the model for scaling studies and for the interpretation of measurements results. This is our ultimate research goal.

Next, we might want to replace the SQP algorithm with a different algorithm because of the matching problems discussed in subsection 8.7.2. A possible substitute for the SQP algorithm is the Simulated Annealing (SA) algorithm, which is a mature and well developed optimization procedure. An important advantage is its ability to move away from local optima, thus the ability to find the global optimum is not related to the initial conditions. For more information about SA algorithms we refer to [3].

Finally, we put a lot of effort in automating the matching procedure, which makes its useability easier. But, since the model and the corresponding matching procedure are implemented using over eight hundred matlab files, the next step is to write a report describing a user-friendly tool for carrying the fluid flow analysis described in this work.
Appendix A

Most of the figures and tables presented in this thesis are constructed with the aid of matlab files, which means that the matlab files immediately display the necessary latex commands. In this appendix we list the matlab files used in this work.

<table>
<thead>
<tr>
<th>Figure</th>
<th>Matlab file</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8 (left)</td>
<td>THESIS_makePictureExpAndVarForSeveralWarmupPeriods.m</td>
</tr>
<tr>
<td>2.8 (right)</td>
<td>THESIS_makePictureAutocovForSeveralWarmupPeriods.m</td>
</tr>
<tr>
<td>3.3-3.7 and 8.9</td>
<td>THESIS_BRD_ACF_Buf_Loss_pictures.m</td>
</tr>
<tr>
<td>4.1</td>
<td>THESIS_makePictureNormAndTDistr.m</td>
</tr>
<tr>
<td>4.2</td>
<td>THESIS_generateChainsandBufferlevel.m</td>
</tr>
<tr>
<td>5.2</td>
<td>THESIS_SketchFitnessfunctionNotLogLog.m</td>
</tr>
<tr>
<td>5.3</td>
<td>THESIS_sixSmallFitnessfunctionPicturesNotLogLog.m</td>
</tr>
<tr>
<td>8.1, 8.2, 8.4 and 8.5</td>
<td>THESIS_makeMetamodelPicture.m</td>
</tr>
<tr>
<td>8.3, 8.6 and 8.7</td>
<td>THESIS_printBestGAandSQP.m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Matlab file</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>THESIS_makeTableForUpsilon.m</td>
</tr>
<tr>
<td>4.1</td>
<td>THESIS_makeTableForTheta.m</td>
</tr>
<tr>
<td>4.2</td>
<td>THESIS_makeTableForBandA.m</td>
</tr>
<tr>
<td>4.3</td>
<td>THESIS_makeTableForD.m</td>
</tr>
<tr>
<td>5.1</td>
<td>THESIS_makeTableForGAPopMutCro.m</td>
</tr>
<tr>
<td>7.1</td>
<td>THESIS_make2kFactdesignMatrix.m</td>
</tr>
<tr>
<td>8.1</td>
<td>THESIS_makeTableForMethods.m</td>
</tr>
<tr>
<td>8.2</td>
<td>THESIS_makeCodingChart.m</td>
</tr>
<tr>
<td>8.3 and 8.8</td>
<td>THESIS_make22FactdesignMatrixForDeltaAndP.m</td>
</tr>
<tr>
<td>8.4 and 8.9</td>
<td>THESIS_makeTableForBestMethod.m</td>
</tr>
<tr>
<td>8.5</td>
<td>THESIS_makeTableTwoExamples.m</td>
</tr>
<tr>
<td>8.6, 8.11 and 8.13</td>
<td>THESIS_makeTableGA_and_SQP.m</td>
</tr>
<tr>
<td>8.7</td>
<td>THESIS_makeTableForSeveralWeights.m</td>
</tr>
<tr>
<td>8.10</td>
<td>THESIS_makeTableTwoExamples_outputside.m</td>
</tr>
<tr>
<td>8.12</td>
<td>THESIS_makeTableForSeveralWeights_outputside.m</td>
</tr>
</tbody>
</table>
Appendix B

Here we describe three supplemental matlab files. The first file provides a graphical understanding of the simultaneous probability of $X$, $R^{\text{in}}$ and $C$. The next two files are movies of the buffer content and the fitness function, respectively.

**THESIS_threeDimProbMatrixPicture.m**

In this file the simultaneous probability of $X$, $R^{\text{in}}$ and $C$ is presented for fixed buffer content values $x_k$ with three-dimensional vertical bar charts. The $x$-axis displays the input rates $r_i^{\text{in}}$ and the $y$-axis displays the capacity rates $c_j$. With these three values we compute the probability as discussed in section 2.2, and display the result on the $z$-axis.

Figure 9.1 displays the probabilities for $x_k = 1$ corresponding to the standard example discussed in subsection 3.1.2. For instance, the probability corresponding to $r_i^{\text{in}} = 1$ and $c_j = 2$ is given by 0.309. With the two sliders displayed under the bar chart we can rotate the picture.

![Simultaneous probability of $R^{\text{in}}$ and $C$ for $x_k = 1$](image)

Figure 9.1: Simultaneous probabilities of $R^{\text{in}}$ and $C$ for $x_k = 1$ corresponding to the standard example.
This file plays a movie of the behaviour of the buffer as discussed in subsection 4.2.1.

Finally, we discuss a movie for the fitness function, which was mainly dealt with in subsection 5.2.1. Here we successively change the values of $L_2$, $\Delta$ and $p$. 
List of Figures

2.1 Fluid flow funnel. ................................................................. 18
2.2 Observation window. ............................................................. 19
2.3 Fluid flow funnel with mathematical notation. ......................... 19
2.4 Markov chains with two states (left) and three states (right) with transition probabilities and rates denoted by \( p \) and \( r \), respectively. .................. 20
2.5 Behaviour of the buffer for \( \Omega_{R,w} = \{0,6\}, \Omega_C = \{3\}, P_{R,w} > 0, K = 5, \Delta t = 1 \) and \( X_0 = 0 \). .................................................. 22
2.6 Multidimensional Markov chain in the case of two input states and two capacity states. ......................................................... 24
2.7 Behaviour of the buffer in an underload situation (left) and an overload situation (right). ......................................................... 28
2.8 Expectation and variation of the output process for several values of the warm-up period (left). Value of lambda for several values of the warm-up period (right). .................................................. 32
3.1 Anticipated time plot, throughput histograms at input and output and HDP (from left to right) in case of a shared bottleneck. ......................... 42
3.2 Anticipated time plot, throughput histograms at input and output and HDP (from left to right) in case of a shaping bottleneck. ......................... 42
3.3 BRDs at the input side and at the output side corresponding to an shared bottleneck (left) with corresponding throughput HDP (right). ................ 43
3.4 BRDs at the input side and at the output side corresponding to an shaping bottleneck (left) with corresponding throughput HDP (right). ................ 44
3.5 ACFs at the input side and at the output side in case of a shared bottleneck. 46
3.6 ACFs at the input side and at the output side in case of a shaping bottleneck. 47
3.7 Buffer content before and after exchanging source and server. .......... 49
4.1 Density functions for the t-distribution with four degrees of freedom and for the standard normal distribution. .................................. 53
4.2 Possible simulation outcome (top). More detailed picture of the behaviour of the buffer (bottom). .................................................. 54
5.1 GA terminology: a population is formed by chromosomes and each chromosome has a set of genes. .......................... 64
5.2 Sketch of the fitness function. ........................................ 70
5.3 Shape of the fitness function for several choices of the parameters. ..... 71

8.1 Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the first method. .................. 92
8.2 Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the fifth method. .................. 92
8.3 Performance of the GA and the SQP algorithm for the best run. ....... 95
8.4 Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the first method. .................. 99
8.5 Contour map (left) and response-surface plot (right) of the fitted meta-model corresponding to the fifth method. .................. 100
8.6 Performance of the GA and the SQP algorithm for the best run. ....... 103
8.7 Performance of the GA and the SQP algorithm for the best run under the extra restrictions. ................................. 106
8.8 The obtained discrete fluid flow model. ................................ 108
8.9 BRDs at the input side and at the output side with corresponding through-
put HDP for the obtained model (top left and right). Corresponding ACFs at the input side and at the output side (middle left). Corresponding buffer content (middle right). Loss probability per interval and average amount of loss (bottom). .................. 109

9.1 Simultaneous probabilities of $R_{in}$ and $C$ for $x_k = 1$ corresponding to the standard example. .......................... 115
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Influence of the parameters on the speed with which the model converges.</td>
<td>38</td>
</tr>
<tr>
<td>4.1</td>
<td>Accuracy of the estimates obtained for the simultaneous probability of $X$, $R_{in}$ and $C$ (left) with corresponding confidence intervals (right).</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Accuracy of the estimates obtained for the the BRD and the ACF (top) with corresponding confidence intervals (bottom).</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Accuracy of the estimates obtained for the buffer content (left) with corresponding confidence intervals (right).</td>
<td>60</td>
</tr>
<tr>
<td>5.1</td>
<td>Dependency between the number of states ($M_{in}$) and the population size ($N$), the number of mutations ($#m$) and the number of crossovers ($#c$).</td>
<td>75</td>
</tr>
<tr>
<td>7.1</td>
<td>Design matrix for a $2^2$ factorial design.</td>
<td>84</td>
</tr>
<tr>
<td>8.1</td>
<td>The eight different methods compared in this work.</td>
<td>89</td>
</tr>
<tr>
<td>8.2</td>
<td>Coding chart for $\Delta$ and $p$.</td>
<td>90</td>
</tr>
<tr>
<td>8.3</td>
<td>Design matrix and expected responses for a $2^2$ factorial design on $\Delta$ and $p$ (top). Expected main effects with corresponding confidence intervals (bottom).</td>
<td>91</td>
</tr>
<tr>
<td>8.4</td>
<td>Selecting the best of the 8 possible methods.</td>
<td>93</td>
</tr>
<tr>
<td>8.5</td>
<td>The obtained state spaces $\Omega_{in}$, steady-state probabilities $\pi_{in}$, ACFs $A_{in}$, transition matrices $P_{in}$, and fitness values $F_{(\pi_{in},A_{in})}$ for the two examples discussed in this section.</td>
<td>94</td>
</tr>
<tr>
<td>8.6</td>
<td>Ten runs for the GA and the SQP algorithm.</td>
<td>95</td>
</tr>
<tr>
<td>8.7</td>
<td>Influence of the weight $w_{(\pi_{in},A_{in})}$ on the fitness values $F_{in}$ and $F_{A_{in}}$.</td>
<td>96</td>
</tr>
<tr>
<td>8.8</td>
<td>Design matrix and expected responses for a $2^2$ factorial design on $\Delta$ and $p$ (top). Expected main effects with corresponding confidence intervals (bottom).</td>
<td>98</td>
</tr>
<tr>
<td>8.9</td>
<td>Selecting the best of the 8 possible methods.</td>
<td>100</td>
</tr>
<tr>
<td>8.10</td>
<td>The obtained state spaces $\Omega_{out}$, steady-state probabilities $\pi_{out}$, ACFs $A_{out}$, state spaces $\Omega_C$, transition matrices $P_C$, and fitness values $F_{(\pi_{out},A_{out})}$ for the two examples discussed in this section.</td>
<td>101</td>
</tr>
<tr>
<td>8.11</td>
<td>Ten runs for the GA and the SQP algorithm.</td>
<td>102</td>
</tr>
</tbody>
</table>
8.12 Influence of the weight $w_{(R_{out},\pi_{R_{out}},A_{R_{out}})}$ on the state space $\Omega_{R_{out}}$, the steady-state probability $\pi_{R_{out}}$ and the ACF $A_{R_{out}}$ (top), and on the state space $\Omega_{C}$, the transition matrix $P_{C}$ and the fitness values $F_{R_{out},\pi_{R_{out}}}$, $F_{A_{R_{out}}}$ and $F_{(R_{out},\pi_{R_{out}},A_{R_{out}})}$ (bottom). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104

8.13 Five runs for the GA and the SQP algorithm under the extra restrictions. . 105
## List of Notation

### Part I: Discrete fluid flow model

#### Notation for the input process

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^{\text{in}}$</td>
<td>input process</td>
<td>19</td>
</tr>
<tr>
<td>$R^{\text{in},(m)}$</td>
<td>input process in the $m$-th simulation run</td>
<td>55</td>
</tr>
<tr>
<td>$\Omega_{R^{\text{in}}}$</td>
<td>state space for the input process</td>
<td>21</td>
</tr>
<tr>
<td>$r^{\text{in}}$</td>
<td>arbitrary element of $\Omega_{R^{\text{in}}}$</td>
<td>21</td>
</tr>
<tr>
<td>$\Delta r$</td>
<td>base rate for the input rates</td>
<td>21</td>
</tr>
<tr>
<td>$M_{R^{\text{in}}}$</td>
<td>number of elements in $\Omega_{R^{\text{in}}}$</td>
<td>21</td>
</tr>
<tr>
<td>$P^{(l)}_{R^{\text{in}}}$</td>
<td>$l$-step transition matrix for the input process</td>
<td>21</td>
</tr>
<tr>
<td>$p_{i_1, i_2}^{(l), R^{\text{in}}}$</td>
<td>$l$-step transition probabilities for the input process</td>
<td>21</td>
</tr>
<tr>
<td>$\pi_{R^{\text{in}}}$</td>
<td>steady-state probability vector for the input process</td>
<td>21</td>
</tr>
<tr>
<td>$\mu_{R^{\text{in}}}$</td>
<td>expectation for the input process</td>
<td>30</td>
</tr>
<tr>
<td>$\mu^{(m)}_{R^{\text{in}}}$</td>
<td>estimate of the expectation for the input process in the $m$-th simulation run</td>
<td>57</td>
</tr>
<tr>
<td>$\sigma^2_{R^{\text{in}}}$</td>
<td>variance for the input process</td>
<td>30</td>
</tr>
<tr>
<td>$\hat{\sigma}^2_{R^{\text{in}}, m}$</td>
<td>estimate of the variance for the input process in the $m$-th simulation run</td>
<td>57</td>
</tr>
<tr>
<td>$\hat{\pi}_{R^{\text{in}}}$</td>
<td>estimate of the steady-state probability vector for the input process</td>
<td>57</td>
</tr>
<tr>
<td>$\hat{\pi}^{(m)}_{R^{\text{in}}}$</td>
<td>estimate of the BRD for the input process in the $m$-th simulation run</td>
<td>56</td>
</tr>
<tr>
<td>$\hat{\Pi}^{\text{in}}$</td>
<td>measure for the accuracy of the obtained estimates of $\pi_{R^{\text{in}}}$</td>
<td>57</td>
</tr>
<tr>
<td>$C_{\hat{\Pi}^{\text{in}}, n}$</td>
<td>confidence interval for $\hat{\Pi}^{\text{in}}$ based on the standard normal distribution</td>
<td>58</td>
</tr>
<tr>
<td>$C_{\hat{\Pi}^{\text{in}}, t}$</td>
<td>confidence interval for $\hat{\Pi}^{\text{in}}$ based on the $t$-distribution</td>
<td>58</td>
</tr>
<tr>
<td>$A_{R^{\text{in}}}$</td>
<td>ACF for the input process</td>
<td>30</td>
</tr>
<tr>
<td>$A^{(m)}_{R^{\text{in}}}$</td>
<td>estimate of the ACF for the input process</td>
<td>57</td>
</tr>
<tr>
<td>$\hat{A}^{(m)}_{R^{\text{in}}}$</td>
<td>estimate of the ACF for the input process in the $m$-th simulation run</td>
<td>57</td>
</tr>
<tr>
<td>$\hat{A}^{\text{in}}$</td>
<td>measure for the accuracy of the obtained estimates of $A_{R^{\text{in}}}$</td>
<td>57</td>
</tr>
<tr>
<td>$C_{\hat{A}^{\text{in}}, n}$</td>
<td>confidence interval for $\hat{A}^{\text{in}}$ based on the standard normal distribution</td>
<td>58</td>
</tr>
<tr>
<td>$C_{\hat{A}^{\text{in}}, t}$</td>
<td>confidence interval for $\hat{A}^{\text{in}}$ based on the $t$-distribution</td>
<td>58</td>
</tr>
</tbody>
</table>

#### Notation for the output process

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R^{\text{out}}$</td>
<td>output process</td>
<td>19</td>
</tr>
<tr>
<td>$R^{\text{out}, (m)}$</td>
<td>output process in the $m$-th simulation run</td>
<td>55</td>
</tr>
</tbody>
</table>
$\Omega_{\text{out}}$ state space for the output process ........................................ 22
$r_{i_{\text{out}}}$ arbitrary element of $\Omega_{\text{out}}$ ........................................ 22
$M_{\text{out}}$ number of elements in $\Omega_{\text{out}}$ ........................................ 22
$\mu_{\text{out}}$ expectation for the output process ........................................ 31
$\sigma_{\text{out}}^2$ variance for the output process ........................................ 31
$\pi_{\text{out}}$ steady-state probability vector for the output process .......... 26
$\hat{\pi}_{\text{out}}$ estimate of the steady-state probability vector for the output process . 58
$\hat{\pi}_{\text{out}}^{(m)}$ estimate of the steady-state probability vector for the output process in the $m$-th simulation run ................. 58
$\hat{\Pi}_{\text{out}}$ measure for the accuracy of the obtained estimates of $\pi_{\text{out}}$ .... 58
$\hat{\Pi}_{\text{out},n}$ confidence interval for $\hat{\Pi}_{\text{out}}$ based on the standard normal distribution 59
$\hat{\Pi}_{\text{out},t}$ confidence interval for $\hat{\Pi}_{\text{out}}$ based on the t-distribution .......... 59
$A_{\text{out}}$ ACF for the output process ........................................ 31
$A_{\text{out}}$ estimate of the ACF for the output process ............................... 58
$A_{\text{out}}^{(m)}$ estimate of the ACF for the output process in the $m$-th simulation run ........................................ 58
$\hat{A}_{\text{out}}$ measure for the accuracy of the obtained estimates of $A_{\text{out}}$ .......... 58
$\hat{A}_{\text{out},n}$ confidence interval for $\hat{A}_{\text{out}}$ based on the standard normal distribution 59
$\hat{A}_{\text{out},t}$ confidence interval for $\hat{A}_{\text{out}}$ based on the t-distribution .......... 59

**Notation for the capacity process**

$C$ capacity process .................................................. 19
$C^{(m)}$ capacity process in the $m$-th simulation run ................. 55
$\Omega_C$ state space for the capacity process .................................. 21
$c_j$ arbitrary element of $\Omega_C$ ........................................ 21
$\Delta \rho$ base rate for the capacity rates .................................... 21
$M_C$ number of elements in $\Omega_C$ ........................................ 21
$P^{(l)}_c$ $l$-step transition matrix for the capacity process .......... 21
$P^{(l),C}_{11,12}$ $l$-step transition probabilities for the capacity process .......... 21
$\pi_C$ steady-state probability vector for the capacity process .......... 21

**Notation for the buffer content process**

$X$ the buffer content process ........................................ 19
$X^{(m)}$ buffer content process in the $m$-th simulation run .......... 55
$\Omega_X$ state space for the buffer content process .................. 21
$x_k$ arbitrary element of $\Omega_X$ ........................................ 21
$\Delta \rho$ grid for the buffer content ........................................ 21
$M_X$ number of elements in $\Omega_X$ ........................................ 21
$K$ size of the buffer ........................................ 19
$\pi_X$ steady-state probability vector for the buffer content process .......... 46
$\hat{\pi}_X$ estimate of the buffer content process .................. 59
Matching procedure for a discrete fluid flow model

$\hat{\pi}_X^{(m)}$ estimate of the steady-state probability vector for the buffer content process in the $m$-th simulation run ........................................ 59
$\hat{\Pi}^X$ measure for the accuracy of the obtained estimates of $\pi^X$ ................. 59
$C_{\hat{\Pi}^X,n}$ confidence interval for $\hat{\Pi}^X$ based on the standard normal distribution . 59
$C_{\hat{\Pi}^X,t}$ confidence interval for $\hat{\Pi}^X$ based on the $t$-distribution ...................... 59

**Notation for the simultaneous probability of $X$, $R^{in}$ and $C$**

$l(x_k,r^{in}_i,c_j)$ simultaneous probability of $X$, $R^{in}$ and $C$ ............................................. 23
$\hat{l}(x_k,r^{in}_i,c_j)$ estimate of the simultaneous probability of $X$, $R^{in}$ and $C$ .......................... 55
$\hat{l}^{(m)}(x_k,r^{in}_i,c_j)$ estimate of the simultaneous probability of $X$, $R^{in}$ and $C$ in the $m$-th simulation run .................................................................................. 55
$\Upsilon$ maximum error in the simultaneous probability of $X$, $R^{in}$ and $C$ ............... 23
$\Theta$ measure for the accuracy of the obtained estimates of the simultaneous probability of $X$, $R^{in}$ and $C$ ............................................................................. 55
$C_{\Theta,n}$ confidence interval for $\Theta$ based on the standard normal distribution .... 55
$C_{\Theta,t}$ confidence interval for $\Theta$ based on the $t$-distribution ........................................ 55

**Extra notation**

$\Delta R$ difference between the input process and the output process ........... 19
$\Delta w$ observation window ................................................................. 19
$\Delta t$ averaging interval ................................................................. 19
$n$ number of intervals ...................................................................... 19
$s$ arbitrary interval ........................................................................ 19
$s^w$ warm-up period ........................................................................ 23
$b$ width of the HDP ........................................................................ 42
$h$ peak-to-peak value of the HDP ..................................................... 42
$g$ grade of deviation of the HDP ......................................................... 42
$k_{max}$ number of lags ........................................................................ 57
$t^{s,e}$ time that the buffer becomes empty in interval $s$ ......................... 27
$t_{comp}$ computation time .................................................................. 37
$\eta$ number of independent replications of the simulation .................... 55
$T$ termination event for the simulation ............................................. 55
$p_{loss}$ loss probability per interval .................................................... 33
$L$ average amount of loss per interval ............................................. 33

**Part II: Matching procedure**

**Input side**

**Notation for the measurements**

$\tilde{\Omega}_{R^{in}}$ state space for the input process obtained from the measurements .... 66
\( \tilde{r}_1^{in} \) arbitrary element of \( \tilde{\Omega}_{R^{in}} \) ......................... 66
\( M_{R^{in}} \) number of elements in \( \tilde{\Omega}_{R^{in}} \) ........................................ 66
\( \mathcal{P}^{in} \) measure for the accuracy of the obtained input rates ................. 90
\( \bar{\pi}_{R^{in}} \) steady-state probability vector for the input process obtained from
the measurements ........................................ 66
\( \tilde{\Pi}^{in} \) measure for the accuracy of the obtained steady-state probabilities
for the input process ........................................ 90
\( \bar{\mu}_{R^{in}} \) expectation for the input process obtained from the measurements .. 66
\( \bar{\sigma}^2_{R^{in}} \) variation for the input process obtained from the measurements ...... 66
\( \bar{A}_{R^{in}} \) ACF for the input process obtained from the measurements ............ 88
\( \mathcal{A}^{in} \) measure for the accuracy of the obtained ACF for the input process . . 90

Notation for the GA and the SQP algorithm
\( \Omega^{*,R,GA}_{R^{in}} \) state space for the optimal (continuous) input rates obtained from the
GA ................................................................. 76
\( r_i^{in,**,R,GA} \) arbitrary element of \( \Omega^{*,R,GA}_{R^{in}} \) ............................................. 76
\( \Omega^{*,D,GA}_{R^{in}} \) state space for the optimal (discrete) input rates obtained from the
GA ................................................................. 76
\( r_i^{in,**,D,GA} \) arbitrary element of \( \Omega^{*,D,GA}_{R^{in}} \) ............................................. 76
\( \Omega^{*,R,GA}_{R^{in}} \) state space for the optimal (continuous) input rates obtained from
the SQP algorithm ............................................. 81
\( r_i^{in,**,R,SQP} \) arbitrary element of \( \Omega^{*,R,SQP}_{R^{in}} \) ............................................. 81
\( \Omega^{*,D,SQP}_{R^{in}} \) state space for the optimal (discrete) input rates obtained from the
SQP algorithm .................................................. 81
\( r_i^{in,**,D,SQP} \) arbitrary element of \( \Omega^{*,D,SQP}_{R^{in}} \) ............................................. 81

Notation for the fitness function
\( F_{R^{in}} \) fitness value obtained from the input rates ................................. 69
\( F_{\pi_{R^{in}}} \) fitness value obtained from the steady-state probability vector for
the input process ............................................. 69
\( F_{R^{in},\pi_{R^{in}}} \) fitness value obtained from the BRD for the
input process .................................................. 69
\( F_{A_{R^{in}}} \) fitness value obtained from the ACF for the input process ............... 72
\( F_{(R^{in},\pi_{R^{in}}),A_{R^{in}}} \) total fitness value obtained for the input process ............ 72
\( w_{R^{in}} \) weights for \( w_{R^{in}} \) ........................................... 70
\( w_{R^{in},\pi_{R^{in}}} \) weight for \( w_{R^{in},\pi_{R^{in}}} \) ........................................... 71
\( w_{(R^{in},\pi_{R^{in}}),A_{R^{in}}} \) weight for \( w_{(R^{in},\pi_{R^{in}}),A_{R^{in}}} \) ........................................... 72
\( F_{\text{max}}^{R^{in}} \) maximum fitness value for the input rates ................................ 69
\( U_{R^{in}} \) error vector for the input rates ........................................ 69
\( L_{1}^{R^{in}} \) seperation point between the constant part and the linear part in
the fitness function ........................................... 69
\( L_{2}^{R^{in}} \) seperation point between the linear part and the reciprocal part in
Matching procedure for a discrete fluid flow model

the fitness function ................................................................. 69
\( \Delta_{R_{in}} \) slope of the linear part in the fitness function ...................... 69
\( p_{R_{in}} \) slope of the reciprocal part in the fitness function ..................... 69
\( b_{R_{in},l} \) lower bound on the elements in \( \Omega_{R_{in}} \) ................................. 73
\( b_{R_{in},u} \) upper bound on the elements in \( \Omega_{R_{in}} \) ................................. 73

Output side

Notation for the measurements
\( \hat{\Omega}_{R_{out}} \) state space for the output process obtained from the measurements . . . . 97
\( \hat{r}_{i} \) arbitrary element of \( \hat{\Omega}_{R_{out}} \) ................................................. 66
\( \hat{\pi}_{R_{out}} \) steady-state probability vector for the output process obtained from the measurements ................................................. 97
\( \hat{A}_{R_{out}} \) ACF for the output process obtained from the measurements ........... 97

Notation for the fitness function
\( F_{R_{out},\pi_{R_{out}}} \) fitness value obtained from the BRD for the output process ................................................................. 103
\( F_{A_{R_{out}}} \) fitness value obtained from the ACF for the output process ............. 103
\( F_{(R_{out},\pi_{R_{out}}),A_{R_{out}}} \) total fitness value obtained for the output process ................................................................. 101
\( w_{(R_{out},\pi_{R_{out}}),A_{R_{out}}} \) weight for \( w_{(R_{out},\pi_{R_{out}}),A_{R_{out}}} \) ........................................... 103

Input side and Output side

Notation for the GA and the SQP algorithm
\( N \) population size used in the GA .............................................. 64
\#c number of crossovers used in the GA ...................................... 74
\#m number of mutations used in the GA ..................................... 74
\( \psi(x) \) merit function used in the SQP algorithm ......................... 81
\( \mathcal{E} \) set of equality constraints used in the SQP algorithm ............. 79
\( \mathcal{T} \) set of inequality constraints used in the SQP algorithm .......... 79

Notation for the experimental designs and for selecting the best method
\( \mathcal{E}_1 \) main effect of factor one ............................................... 84
\( \mathcal{E}_2 \) main effect of factor two ............................................... 84
\( \mathcal{E}_3 \) interaction between factors one and two ......................... 84
\( \mathcal{C}_{\mathcal{E}_i,n} \) confidence interval for \( \mathcal{E}_i \) based on the standard normal distribution .. 90
\( \mathcal{C}_{\mathcal{E}_i,t} \) confidence interval for \( \mathcal{E}_i \) based on the t-distribution .......... 90
\( \mathcal{S}_i \) expected response from the \( i \)th factor level ..................... 83
\( \mu_{\mathcal{S}_i}^{(1)} \) first-stage sample mean for method \( i \) ......................... 85
\( \mu_{\mathcal{S}_i}^{(2)} \) second-stage sample mean for method \( i \) ......................... 85
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_{S_i}^{(1,2)}$</td>
<td>weighted sample mean for method $i$</td>
<td>86</td>
</tr>
<tr>
<td>$\sigma^2_{S_i}$</td>
<td>variance estimate corresponding to the first-stage sample mean for</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>method $i$</td>
<td></td>
</tr>
<tr>
<td>$w_{S_i}^{(1)}$</td>
<td>weight for the first-stage sample mean for method $i$</td>
<td>86</td>
</tr>
<tr>
<td>$w_{S_i}^{(2)}$</td>
<td>weight for the second-stage sample mean for method $i$</td>
<td>86</td>
</tr>
</tbody>
</table>
Bibliography


