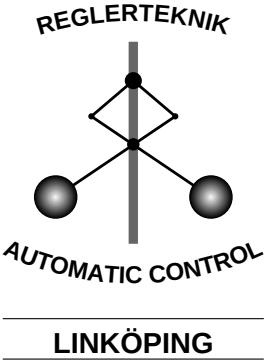
Linköping Studies in Science and Technology
Thesis No. 1048

On Modeling and Control of Network Queue Dynamics

Frida Gunnarsson



Division of Automatic Control and Communication Systems
Department of Electrical Engineering
Linköpings universitet, SE–581 83 Linköping, Sweden
WWW: http://www.control.isy.liu.se
Email: frida@isy.liu.se

Linköping 2003

On Modeling and Control of Network Queue Dynamics

Department of Electrical Engineering,
Linköpings universitet,
SE–581 83 Linköping,
Sweden.

ISBN 91-7373-764-X
ISSN 0280-7971
LiU-TEK-LIC-2003:47

Printed by UniTryck, Linköping, Sweden 2003

To Anders

Abstract

Internet flow control algorithms used today are separately designed and tuned for outdated setups and traffic mixes. Differnet algorithms exist in both end nodes an core routers of a network. This work presents improved techniques for network flow control. The solutions are adaptive and scalable to account for changing network structures and shifts in the traffic mix. The flow control is performed at the network routers by adaptive queue management. The problem is identified to be a control problem, which requires dynamic models to base control design on. System identification and automatic control are used to design adaptive queue management techniques. It is shown that biased autoregressive models of the queue length dynamics capture the main dynamics. The throughput is increased and the oscillations damped by using knowledge from the model, in the control.

The techniques for adaptive queue management are triggered by packet arrivals, which indicate nonuniform sampling rather than uniform. Techniques for modeling dynamics based on nonuniform samples are discussed. Accurate approximations of the signal spectra can be used to adopt continuous time models. Approximations of the Fourier transform based on nonuniform samples are developed and evaluated.

Continuous time models can be identified using standard identification techniques, from nonuniformly sampled data. Some of the considerations that has to made are presented. Filtering at nonuniform sample times is also discussed.

There exist many control problems on different levels in a network. An overview of network control is given, based on available specification. The control problems are also presented using block diagrams and connections between the problems are highlighted. It is shown that, is important to consider the structure of underlying network control when design is done.

Sammanfattning

En svensk översättning av avhandlingens titel är **Reglering och modellering av ködynamik i nätverk**.

Internet är ett komplext kommunikationssystem som började byggas i slutet på 1960-talet. Det är uppbyggt av lager med funktionalitet för att underlätta för ny design. Varje lager är avskärmat från över och underliggande lager och har hand om en specifik del av kommunikationen. Data på internet skickas paketvis och varje paket är individuellt, endast vid sändare och mottagare finns paketen med säkerhet i rätt ordning. Flera regler finns för att kommunikationen ska vara så bra, snabb och säker, som möjligt. Dessa regler är i många fall kvarlevor från Internets tidiga dagar och forskare världen över arbetar med förbättringar på olika plan.

Avhandlingen börjar med en översikt om hur reglerteknik kan användas på Internet både på hög och låg nivå. Detta synsätt används sedan för att förbättra prestanda i en kö på nätverket, genom att hitta samband för hur kön uppför sig och utifrån dessa ta beslut om ett paket som just anlänt ska kastas eller inte. Detta ger resultat på grund av att avsändaren till paketet kommer att sänka sin sändhastighet när sändningen inte lyckades, därför dämpas också inhastigheten till kön.

Köregleringen på Internet ger upphov till flera nya problem, bland annat på grund av att paket inte anländer i regelbunden takt utan tiden mellan deras ankomst varierar slumpmässigt. Avhandlingen behandlar matematik för ickelikformig sampling, som dessa slumpmässiga ankomster ger upphov till. Detta görs för att kunna använda den designade tekniken i en verklig Internetkö, samt för att utveckla den allmänna teorin på området.

Acknowledgments

A lot of work has been done behind the scenes for this thesis to be possible. With this short words I will try to express my gratitude for all this. First of all, the two Fredriks, who provide valuable advices. Professor Fredrik Gustafsson drafted me a few years back and Dr. Fredrik Gunnarsson have helped with inspiration and guidance.

I also thank Professor Lennart Ljung for allowing me to join the group and our secretary Ulla Salaneck for being invaluable and knowing almost everything. Numerous colleagues have had the (mis)fortune of answering my questions. With the risk of forgetting someone: Fredrik Tjärnström was the target of my identification questions. Rickard Karlsson and Måns Östring have, among other things, provided LaTeX help. Jonas Elbornsson has helped with tricky Matlab questions. Mikael Norrlöf and Ola Härkegård had offices nearby during my initial time at the group, and have helped with a lot of startup questions as well as more recent problems. Other specific issues have been solved by Ingela Lind and Martin Enqvist. Thank you every one!

I am also indebted to Ola Härkegård, Jonas Gillberg, David Törnqvist and Jonas Jansson for thorough proofreading and valuable comments, that significantly improved the work. The whole group is thanked for the open doors and fika-discussions, that make working here very much enjoyable.

This work was supported financially by the ECSEL (Excellence Center in Computer Science and Systems Engineering in Linköping) graduate school and VINNOVA's Competence Center ISIS (Information Systems for Industrial Control and Supervision), who are both gratefully acknowledged.

––

Jag vill tacka mina vänner, och särskilt $T\pi$, för viktiga perspektiv på livet. Mamma, pappa och lillstrumpa: era nyfikna och oroliga frågor är mycket uppskattade. Jag försöker svara så gott det går. Nu är iallafall boken klar, det är bara festen kvar.

Till slut, Anders, tack för ditt stöd och ditt tålamod. Det blir inte lika bra utan dig.

Frida Gunnarsson

Linköping, November 2003

Contents

Notation ix

1 Introduction 1
- 1.1 Network Preliminaries . . . 2
  - 1.1.1 Internet History . . . 2
  - 1.1.2 Layer Structure . . . 2
- 1.2 System Theory Preliminaries . . . 5
- 1.3 Contributions . . . 8
- 1.4 Thesis Outline . . . 9

2 System Overview 11
- 2.1 Streaming Data and Transferring Files . . . 12
- 2.2 Transportation and Routing . . . 12
  - 2.2.1 Flow Control . . . 14
  - 2.2.2 Active Queue Management . . . 16
- 2.3 Radio Links . . . 19
- 2.4 Control Structures and Interactions . . . 20
  - 2.4.1 Application Layer without Flow Control . . . 21
  - 2.4.2 A Streaming application . . . 22
  - 2.4.3 Transmission Control Protocol . . . 23
  - 2.4.4 Active Queue Management . . . 23
  - 2.4.5 Radio Link Layer . . . 24
- 2.5 Summary . . . 25
- 2.A Different Versions of TCP . . . 26

3 Queue Management 29
- 3.1 Performance Issues in Queue Management . . . 29
  - 3.1.1 Problems with TCP . . . 30
  - 3.1.2 Measurements Used in Current Research . . . 32

3.1.3 Using the Bottleneck Queue . . . 33
3.2 Modeling and Identification of the Queue Dynamics . . . 34
3.2.1 Frequency Analysis and Filtering . . . 36
3.2.2 AR Modeling of Nonzero-Mean Signals . . . 37
3.2.3 Recursive Identification . . . 39
3.3 Control of the Queue Length . . . 41
3.3.1 Estimation of the Derivative . . . 43
3.3.2 Simulation Results . . . 45
3.4 Summary . . . 49

4 Nonuniform Signal Processing **51**
4.1 Sampling . . . 51
4.2 Fourier Transform Approximation . . . 54
4.2.1 Extensions of the Riemann Approximation . . . 56
4.2.2 Resampling through Local Polynomial Models . . . 58
4.2.3 Basis Expansion . . . 62
4.2.4 Spline Interpolation . . . 65
4.2.5 Nyquist Criteria . . . 68
4.2.6 Zero-Mean Signals . . . 69
4.2.7 Collection of Transform Approximation Expressions . . . 69
4.3 Additive Random Sampling . . . 71
4.4 Evaluation . . . 75
4.4.1 Signal Model Evaluation . . . 76
4.4.2 A Large Example . . . 77
4.5 System Sampling . . . 82
4.5.1 Filtering . . . 84
4.6 Model Estimation . . . 85
4.6.1 Cost Function . . . 87
4.6.2 Output Estimation . . . 87
4.6.3 Calculation of Optimal K . . . 88
4.7 Summary . . . 89
4.A Calculation of the Mean for a Sum of sinc-Functions . . . 91
4.B Limited Distributions . . . 94

5 Conclusions **95**

Bibliography **97**
3GPP Specifications . . . 101
Request For Comments . . . 101

Notation

Abbreviations

cf.	confere, compare
e.g.	exempli gratia, for the sake of example
i.e.	id est, that is
i.i.d.	independent identically distributed, about stochastic variables
n.b.	nota bene, take notice

Acronyms

3G	Third Generation radio network
ACK	Acknowledgment (number), used in TCP as a receipt of received data.
AQM	Active Queue Management, mechanisms for intelligent control of packet flows through routers.
DiffServ	Differentiated Services, techniques to handle different traffic types according to their demands.
DFT	Discrete Fourier Transform.
ECN	Explicit Congestion Notification, an AQM mechanism.
FTP	File Transfer Protocol, handles regular file transfers at the application level.
HTTP	Hyper Text Transfer Protocol, is used to transfer web sites, at application layer.

IP, IPv6	Internet Protocol (version 6), handles routing of individual Internet packets at network layer.
MAC	Medium Access Control, unacknowledged data transfers over the radio link.
QoS	Quality of Service, different traffic types have different demands on the transmission.
RED	Random Early Detection, algorithm for flow control in routers.
RFC	Request For Comments, standards and proposals for network components.
RLC	Radio Link Control, protocol for retransmissions over the radio link.
RNC	Radio Network Controller, handles RRM in the UMTS network.
RRM	Radio Resource Management, handling, e.g., power limitations for individual base station.
RTP	Realtime Transfer Protocol, transport protocol designed specifically for stremaing transmissions.
SMTP	Simple Mail Transfer Protocol, transfer of electronic mail.
TCP	Transmission Control Protocol, protocol for end-to-end flow control from senders at transport level.
UDP	User Datagram Protocol, simple transport layer protocol used when TCP is overkill.
UMTS	Universal Mobile Telcommunication System, future standard for mobile telephony.
UTRAN	UMTS Terrestral Radio Access Network, part of the 3G system.

Math

$\forall$	for all, e.g., $x(t) = 0$, $\forall t$ describes a zero signal.
$x(t)$	A continuous signal measured at the time t.
x_k	A discrete measurements of x at time t_k, i.e., $x_k = x(t_k)$.
$\mathcal{F}(x)$	The Fourier transform of the continuous signal x.
$\mathcal{F}(x)_*$	The Fourier transform based on the discrete measurements of x, and the subscript, $*$, denotes the approximation method, e.g., ra for Riemann approximation or sp for spline interpolation.
$X_*(f)$	The same as $\mathcal{F}(x)_*$
$E[x]$	The expected value of the stochastic variable x.
$\mathbf{Z}$	The set of real integers
$\mathbf{N}$	The set of natural numbers, equal to the set $\{0, \mathbf{Z}^+\}$

$\mathbf{R}$	The set of real numbers
$\mathbf{C}$	The set of complex numbers
$\{x : A(x)\}$	The set of all x that makes $A(x)$ true.
$\|x\|$	The absolut value of x.
$\|x\|$	The norm of the vector x, $\|x\|^2 = \sum \|x_i\|^2$.
I	The identity matrix, $I^{-1} = I$, $Ix = x$. A subcript index is used to denote the size if there is risk for confusion, I_n.
$x \triangleq a$	x is defined as a.

Introduction

1

Communication has always been part of man's daily life. The techniques for communicating have therefore evolved together with the rest of the technical development. The postal services played a large part in building the infrastructures during the industrial revolution and it did not take long after the computers emerged until the thought of communication between computers started. Today the Internet is the largest man made system in the world, both geographically and in the number of connected parts. It is also one of todays most widely used tools for communication and information access, at least in the industrial world. The demands on the transmissions over this system is increasing with the number of users and the complexity of the applications. It is important that data is delivered exact, fast and uncorrupted. The size of the network makes it impossible to model every detail, level and packet and other methods are needed to analyze performance and design new algorithms for control of the transmissions.

The solutions need to be scalable and adaptive in order to be useful for different network sizes and traffic mixes. The tools in systems theory and automatic control gives possibilities to adaptively find models of the transmission and use these to tune control designs. An immense interest for this field is growing from the system theorists, and a large effort has already been carried out to find useful solutions to the upcoming performance problems.

This thesis will study the network problems from a control perspective and, in particular, the task of improving performance over Internet routers are discussed. The application at hand gives rise to a study of nonuniform sampling, that are independent of the particular problem.

This chapter will introduce some of the key concept that is necessary for the coming discussions. In Section 1.1, an introduction to the network system is given. A short history time line and the structure of the layered communication system is presented. In Section 1.2, systems theory is introduced with basic concept in system identification and automatic control. The previ-

ous publications that contributed to this work is listed in Section 1.3 and the outline of the thesis is presented in Section 1.4.

1.1 Network Preliminaries

To understand the structure of the network application, some background knowledge is necessary. First a short overview of some of the key events during the first years of computer network history is given. Then the building blocks of the Internet communication system, the layers and their functionality, are discussed.

1.1.1 Internet History

To give an overview of the time horizon for the Internet evolution some of the key events are here presented in chronological order. A continuously developing time line can be found in Zakon (2003). In **1961**, the first paper on packet switching was published and later on there was also a book on the subject both written by Leonard Kleinrock. In **1962**, a series of memos from J.C.R. Licklider was published, envisioning a globally interconnected set of computers where everyone could access data from any site. In **1965**, the first wide-area network was built, connecting two computers, by Lawrence Roberts. This experiment showed that packet switching was needed and circuit switching was totally inadequate. In **1966**, several papers on packet networks were presented at a conference and, in **1968**, the design of the first packet switches started at a small firm called BBN. Besides the team at BBN lead by Frank Heart, Robert Kahn was involved in the overall network architectural design, network topology and economics were designed and optimized by Roberts, and the network measurement system was prepared by Kleinrock's team at UCLA.

The first node in the network was installed at UCLA in **1969** and by the end of the year, four hosts were connected together, at the universities in Stanford, Santa Barbara and Utah. In **1970**, the first host-to-host protocol, Network Control Protocol (NCP), was designed and after the final implementation the users of the network could start with application implementations. In **1972**, the network was demonstrated to the public by Roberts and electronic mail was introduced. In **1973**, the design of what was to become the protocol suite TCP/IP started by Robert Kahn and Vinton Cerf. Much more information including references to the mentioned publications can be found in Leiner et al. (2000, Sec. Origins of the Internet).

1.1.2 Layer Structure

To manage the large system of interacting computers, the information flow on the Internet is structured into layers. Each layer is responsible for a certain

task and does not know anything about how layers above or below carry out their tasks. The actual data traverses the layer stack top down at the sender but the layer structure is transparent and information is seemed to be carried between the corresponding layers at the sender and the receiver using headers attached to the data. Every packet contains a header with information about the packet and data, which is delivered. Every layer adds a new header with information relevant for the corresponding layer at the receiver. The task of each layer is carried out using different *protocols*. Protocols are rules on how to handle files and packets at different layers. Different layers can use different protocols, depending on the type of transmission.

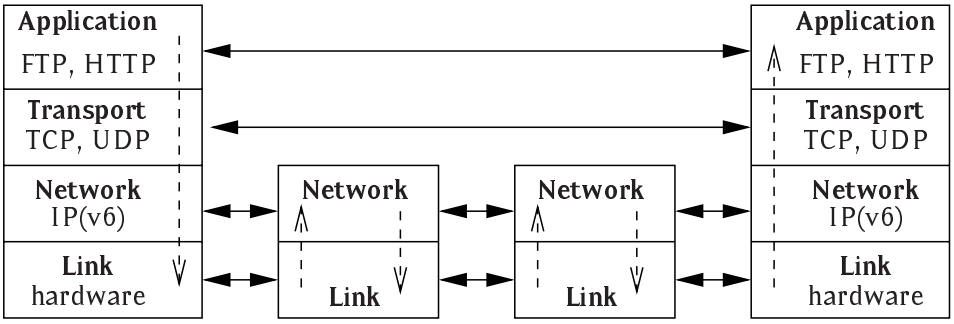


Figure 1.1 *The explicit data transfer (dashed) and the communication (solid) in the layered network structure. Each bar represents a physical position in the network, such as routers and servers.*

Example 1.1 (Data transfer) *When the goal is to deliver a file from sender A to receiver B different things happen at different layers. Both horizontal and vertical communication take place. The structure is described in Figure 1.1 together with names of some of the protocols for each layer..*

Application *A file is sent from A to B. When the whole file is successfully delivered a notification is sent to the sending application. The application delivers the IP address and port number to be used to the layer below.*

Transport *The file is separated into packets, each with a header stating the size of the packet and the port to which it should be delivered. After each packet is successfully delivered an acknowledgment is received, otherwise it is retransmitted. The IP address is forwarded downwards.*

Network *The packet is provided a header with the source and destination address, the IP addresses of A and B respectively, producing an IP frame. After each frame is sent the next one is prepared. The address of the next computer in the network is determined and delivered to the next layer.*

Link *The frame is given a header with the local address of the next computer and delivered on the physical link. The physical link can be both a wire and a radio link. In the latter case a number of internal layers are used to enhance the communication over the radio link.*

In this case the delivery of the file is always successful since the transport layer makes sure that lost packets are retransmitted. For other application types, e.g., real time transmissions, this might not be a desired behavior and the correction can instead be made in the application, e.g., by using redundant data, coding or retransmissions. □

In Figure 1.2, the principal components of a network are shown. The senders are at the ends and the packets traverses links and queues to the receiver. The receipts or acknowledgments, ACKs, go the other way. The setup can be compared to Figure 1.1 where the link and network layer exists in all nodes but the higher layers are only present in the sources and destinations.

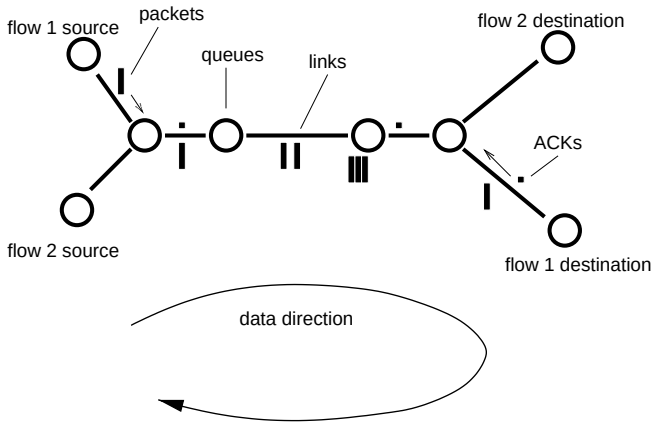


Figure 1.2 *A simple network with packets and receipts, queues and links. Each end node contains all the layers and the core nodes contains the two lower layers, see Figure 1.1.*

A generic layer provides a service to the upper layer and uses a service from the lower layer. Decisions are made on whether or not to forward additional data downward based on quality measures. The lower layer is thus fed with a varying data rate and the lower layer needs to try to provide its service fast enough.

This four layer model belongs to the TCP/IP protocol suite which is a collection of protocols for all the different layers. In Stevens (1994, Ch. 1) a

more formal description of the layers is given. The different parts including the radio link are described further in Chapter 2.

1.2 System Theory Preliminaries

What is a system? The connection between variable signals, inputs, and measured signals, outputs, is called a *system*. An example is a dam. The level can be measured and is also affected by the size of the hatch. Another example is a thermostat in a house. The indoor temperature can be measured and it is changed when the power to the radiators is changed. The system is a description of how the changes in one signal correspond to changes in the other. Usually, there are phenomena that cannot be related to changes in the inputs. These changes in the outputs are said to come from disturbances. In the dam example, the disturbances are rain and other weather phenomena and for the thermostat, the outdoor temperature is a disturbance. To describe the connection between inputs, outputs and disturbances *block diagram descriptions* are used in system theory. The system is described using mathematical tools. Figure 1.3 shows a generic block diagram for a system P, with inputs u and disturbances w. These signals affects P and the system produces the output y.

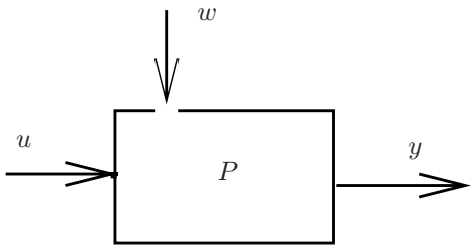


Figure 1.3 *Block diagram of a generic system, P, with inputs u, outputs y and disturbances w.*

To be able to describe the system in a mathematical way, *system identification* is used to find a description that best describes the connection between measurements of inputs, u, and outputs, y. The connection can be formulated as a function $y = f(u, \theta)$, where θ contains the parameters that will be used to make a good fit to observed measurements.

The best parameters are usually found by minimizing the difference between the model and the actual data. If we have a number of measurements pairs, y_i, u_i, it is logical that the difference between y_i and $f(u_i, \theta)$ is made as

small as possible for all i, i.e.,

$$\min_{\theta} \sum_{i} |y_i - f(u_i, \theta)|^2 \tag{1.1}$$

is used to decide what values of the parameters θ should be used. The system description in f is not normally static, but also dependent of derivatives or previous values of the signals.

Example 1.2 (Trend in input rate) *Consider a queue where the incoming rate is changes as a trend over time. A model can then be stated for the rate r based on t:*

$$r_s = at + b$$

An example of measurements of the rate is shown in Figure 1.4. The rate measurements of r_s are affected by noise w, since they deviate from the line,

$$r_m = r_s + w.$$

The task of system identification is to find the best values of the constants a and b based on this data, as was shown in Eq. (1.1). The best line in this case is shown as the thick line. It corresponds pretty good with the true one in the area of the measurements, but it is clear that the deviations will get larger if we move away from this region.
Based on this model the length of the queue, q, can be calculated using the send rate from the queue, r_s:

$$q(t) = \int_0^t (r_r(\tau) - r_s(\tau))d\tau.$$

□

The system is affected by the inputs and produces a certain output. This can be used to control the output by chosing good inputs. The task is called *automatic control* and the aim is to produce a wanted behavior of the output signals by controlling the inputs based on measurements of the outputs. The controller is a set of rules on how to calculate the input, u, based on the output and the wanted behavior, the reference signal, r. The aim is to keep $|r - y|$ small. A generic block diagram for a system controlled by automatic control is shown in Figure 1.5.

Example 1.3 (Proportional control) *A simple, but often good enough, controller is the proportional controller. P-control means that the output u is made proportional to the error in the output, y, i.e.,*

$$u = K_P(r - y).$$

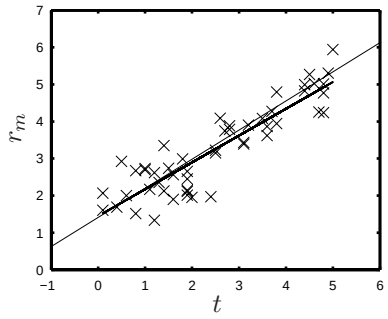


Figure 1.4 *Measurements, r_m ('x') of the queue input rate, $r_s = at + b$ (thin), affected by noise, w, varying according to a linear trend.*

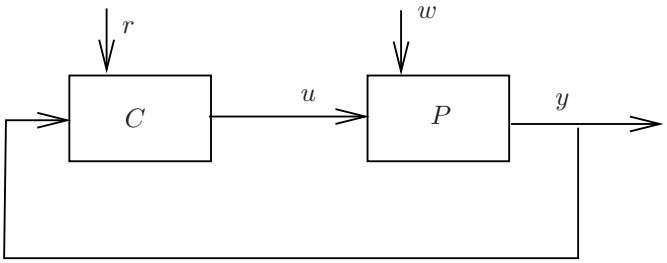


Figure 1.5 *A generic block diagram when automatic control is used to control the behavior of y. The input, u, to the system, P, is chosen by the controller, C, based on measurements of y and the reference signal, r.*

The logic is that often larger power or bigger movements or something like this is needed to correct for a large error than for a small one. The choice of the constant K_P is very much dependent on the application and on the demands on the system.
Studying the queue system in Example 1.2 and using the send rate as the control signal, we can control the queue length as

$$r_s = K_P(q_{ref} - q),$$

which will keep the true queue length close to q_{ref} by following the input rate. □

Example 1.4 (Derivative action) *In cases of unwanted high variations in the output signal derivative action can be used, together with proportional*

control, to suppress oscillations. The controller becomes

$$u = K_P(r - y) + K_D\dot{y}.$$

This means that when y is changing fast the control signal is large even if the error, $r - y$, is not large. □

System identification and automatic control will be used in Chapter 3 to perform better queue management.The calculations are on models of the queue length dynamics and on simple controllers, with derivative action to damp oscillations in the queue length.

1.3 Contributions

The main contributions of this thesis are the following:

- An overview of the network system, where descriptions and connections of the existing control problems are presented based on a system theoretic framework.
- The problem of queue management is thoroughly analyzed and controllers are designed based on simple, but accurate, models of the queue length dynamics.
- Nonuniform signal processing is discussed, with emphasize on frequency analysis for identification.

Part of the work in this thesis has been or will be published elsewhere. Discussions about performance issues have been presented at two Nordic conferences:

- Frida Gunnarsson, Fredrik Gunnarsson, and Fredrik Gustafsson. **TCP Performance based on Queue Occupation**. In *Towards 4G and Future Mobile Internet, Proceedings for Nordic Radio Symposium 2001*, March 2001.
- Frida Gunnarsson, Fredrik Gunnarsson, and Fredrik Gustafsson. **Issues on Performance Measurements of TCP**. In *Radiovetenskap och Kommunikation '02, RVK'02*, June 2002.

Results for AR-modeling of queue dynamics and improved performance is to be presented at the 42nd IEEE Conference on Decision and Control:

- Frida Gunnarsson, Fredrik Gunnarsson, and Fredrik Gustafsson. **Controlling Internet Queue Dynamics using Recursively Identified Models**. To appear in *42nd IEEE Conference on Decision and Control*, December 2003c.

Efforts on modeling of radio network components and performance analysis is published as a technical report at Linköpings universitet and presented at the First Swedish National Computer Networking Workshop:

- Frida Gunnarsson, Anders Björsson, Björn Knutsson, Fredrik Gunnarsson and Fredrik Gustafsson. **Radio Access Network (UTRAN) Modeling for Heterogeneous Network Simulations**. Technical Report LiTH-ISY-R-2533, Dept. of Electrical Engineering, Linköpings universitet, August 2003a.

- Frida Gunnarsson, Anders Björsson, Björn Knutsson, Fredrik Gunnarsson and Fredrik Gustafsson. **Radio Access Network (UTRAN) Modeling for Heterogeneous Network Simulations, a brief description**. In *First Swedish National Computer Networking Workshop*, September 2003b

A contribution about nonuniform frequency analysis is submitted to the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing:

- Frida Gunnarsson, Fredrik Gustafsson and Fredrik Gunnarsson. **Frequency Analysis unsing Non-Uniform Sampling with Application to Adaptive Queue Management**. Submitted to *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

The publications can be found at `www.control.isy.liu.se/publications`.

1.4 Thesis Outline

The structural connection between the different parts of the thesis is shown in Figure 1.6. An overview of the network system is presented in Chapter 2. Standards and common practice throughout the protocols are explained and the different control systems are identified and modeled. The connections between different control systems are also discussed. In Chapter 3, the focus is on queue management. The control system is further developed and several controllers are presented with theory and results. A discussion about nonuniform sampling for identification is done in Chapter 4, where the focus is on frequency analysis, but preliminary results on identification and filtering are also presented. Concluding remarks are presented in Chapter 5 together with a discussion about future work.

The thesis is written using LaTeX 2_ε and the layout partly inspired by Jane Austen and other girly writers.

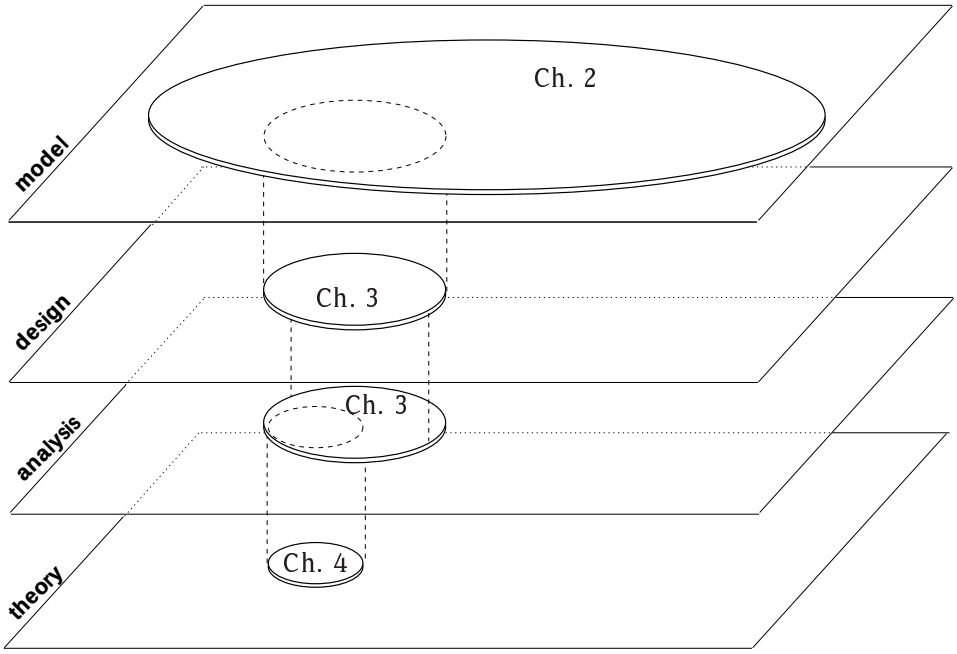


Figure 1.6 *Structural picture of thesis content.*

System Overview

Usually when studying Internet issues, the focus is on a certain level in the system or on a particular problem. To understand the complexity of controlling the traffic, an understanding of the whole picture is needed. Different levels interact and counteract in their demands on the communication.

Design of new algorithms require evaluation of the performance over the full network. Simulations can be done with different open source simulator such as *ns-2* (NS) and REAL (Keshav, 1997), or using proprietary environments. To make a good design, on the other hand, models are needed of different components and how they affect the performance of the specific task.

Due to the complexity of the network, simplifications of different levels are made to facilitate easier analysis. When designing for a certain layer, it is common to model lower layers as a delay, possibly varying, and a queue. This is in most cases true, but the behavior of the variation can be difficult to model and is strongly affected by the specific underlying protocols. The design parameter for the queue is the maximum length and the service rate which mostly is affected by the underlying topology.

This chapter describes the Internet standards for each layer, see Section 1.1.2, and the main functionalities they incorporate. The application layer is described in Section 2.1, where streaming and file transfers are emphasized. The main transport layer protocol is described in Section 2.2 together with existing and new link layer functionality. A short view of the functionality in a radio link is given in Section 2.3. The control view that was introduced in Section 1.2 is further expanded, using the control mechanisms described in the standards, in Section 2.4. The interaction between different parts of the network is also highlighted.

Standards and proposals in the network architecture are described in so called Requests For Comments, RFCs. Anyone can write an RFC and they have different categories such as standards track, informational, draft and ex-

perimental. As the name suggests, anyone can have opinions on the content, and the aim originally was to make Internet a product of consensus. The beginning of an RFC is always something like:

> This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

For the radio part, specifications are produced by the organization 3GPP (Third Generation Partnership Project), and consequently they are called 3GPP Technical Specifications, 3GPP TS. These are formed through "agreements" between the leading producers in the area.

2.1 Streaming Data and Transferring Files

The application layer, Figure 1.1, has one task, delivering a file from one user to another. The files can have different content and, therefore, put different demands on the transfer. Two extremes can be recognized, real-time/streaming and best effort. Best effort transfers can be email or file downloads, where the most important thing is to deliver the whole file without errors and there are no restrictions on the delay. Real-time transfers, on the other hand, can be video and radio transmissions, where the most important thing is to deliver a sequence at the same speed as it is intended to be viewed, e.g., for a TV transmission sound and video must be synchronized. Usually such an application includes some kind of error correction to be able to deliver even though some data is lost or corrupt. In between real time and best effort, we have interactive web surfing and conversations with varying delay restrictions and error tolerance.

The File Transfer Protocol, FTP, as described in RFC 0959, is used to copy data files between users, usually from FTP-servers which requires login. Actually, two connections are used for an FTP transmission, one for control commands and one for the data transfer. The Hypertext Transfer protocol, HTTP, is used to communicate web pages. It is specified in RFC 2616. To deliver all electronic mail, the Simple Mail Transfer Protocol, SMTP (RFC 2821), is used. All these and many more are different application protocols used for transferring information between client and server or from user to user.

2.2 Transportation and Routing

Transportation and routing were originally included in one protocol, NCP, see Section 1.1.1. This protocol was supposed to take care of flow control, retransmissions, routing and error control among other things. When the complexity of the network increased it was divided into two: The Internet Protocol, IP, and the Transmission Control Protocol, TCP.

The *Internet Protocol*, IP, concerns routing and is the only protocol used at the network layer. Routing is per-hop and per-packet based although routes through the network tend to be static over, e.g., one file transfer. IP has no control mechanism on the flow level, this is all left to TCP, the Transmission Control Protocol, at the transport layer. In each router the final destination is checked, the best way is found and IP adds the address of the next router to the packet and then also a checksum. The checksum is of course tested before the packet is further processed — if something is wrong the packet is thrown away. The specification for IP can be found in RFC 0791.

The *Transmission Control Protocol*, TCP, is the most common protocol at the transport layer and the basic specification can be found in RFC 0793. It performs flow control, by adding sequence numbers and using receipts for received data. The flow control mechanisms are described further in Section 2.2.1. TCP also adds port numbers and a checksum to each packet, and a note on how much free capacity it has to receive data, called the advertised window.[1] The data receipts are called acknowledgment numbers, ACKs, and they inform the sender of the packets what data the receiver is expecting. The ACKs are sent based on arriving packets, after every or every two packets.

Example 2.1 (ACKs) *A sequence of packets arrive at the receiver and we study the corresponding ACK numbers that will be used*

packet numbers	*1*	*2*	*3*	*5*	*6*	*4*
ACK numbers	*2*	*3*	*4*	*4*	*4*	*7*

The receiver is continuing to expect packet number 4 and then, as 5 and 6 have arrived previously, packet number 7 is expected next. □

There are many different version of TCP, and the focus here is on the structure rather than specific details. The details are taken from the TCP Reno version, other versions are described in Appendix 2.A.

A simpler transport protocol is the User Datagram Protocol, UDP (RFC 0768), where the control functions are left for the application. UDP adds the checksum and the port numbers but is not concerned with ordering or retransmissions. Other specialized protocols also exist, e.g., the Real-Time Transport Protocol, RTP, that is design especially for streaming transfers. The specifications can be found in RFC 1889. RTP adds sequence numbers for ordering, time stamps and also allows delivery monitoring. It does not retransmit packets. Usually UDP is used below RTP to get the checksum functionality as well.

[1] This window is a data window (the unit is bytes or packets) and has no connection to windows used in signal processing, apart from the name.

Many applications tend to use TCP anyway since it poses no limitations for the receiver. TCP is standard in all operating systems today.

On the network between the sender and the receiver the packets traverses links and router. In the routers queues are formed, since the incoming rate of packets might vary while the processing rate or sending rate of the routers are fixed.

Active Queue Management, AQM, is a set of techniques to control the queue length in the network routers by dropping or marking packets. The functionality is in the network layer but the design is made based on assumptions on the transport layer. The first proposal, and also most common, is Random Early Detection, RED which is described in Section 2.2.2.

2.2.1 Flow Control

TCP includes mechanisms for control of the send rate in order to interact with all traffic and prevent overloading or congestion of the network. As shown previously in Figure 1.2, the flow control performed by TCP resides in the end nodes, senders and receivers. The sender transmits data in packets and the receiver transmits ACKs in returning packets. Both the sender and the receiver use TCP mechanisms. At the packet arrival times, t_k, control variables are calculated. TCP keeps an internal window, the congestion window c_k, which is adapted as a measure of the capacity of the network, in bytes or packets. TCP also keeps track of the amount of data that is sent but not yet acknowledged, the outstanding data or flight size, $f(t)$. The received advertised window, a_k, which is delivered with the arriving packet, is the capacity of the receiver. These three, c_k, $f(t)$ and a_k, are used to limit the outgoing send rate, $r_{TCP}(t)$,

$$r_{TCP}(t) = \min(c_k - f(t), a_k), \text{ for } t \geq t_k \tag{2.1}$$

The congestion control algorithm specified in RFC 2581, states the updating mechanism for c_k based on acknowledged data. A threshold, s, should be used to separate different parts of the update, slow start and congestion avoidance. During slow start, $c_k < s$, c is increased with the size of one packet at the arrival of new ACKs. When $c_k \geq s$, the congestion avoidance phase is entered and the update is done with approximately one packet for each round trip time period, T_{RT}, RTT. The round trip time is a measure of the time it takes from a packet is sent until it is acknowledged. The flight size, f, was defined as the sent but not yet acknowledged data and can therefore be calculated as the sent packets during the last round trip time

$$f(t) = \int_{t-T_{RT}}^{t} r_{TCP}(\tau) d\tau. \tag{2.2}$$

Representing c and s in packets, rather than bytes, we get

$$c_k = \begin{cases} c_{k-1} + 1 & c_{k-1} \leq s \\ c_{k-1} + \frac{1}{c_{k-1}} & c_{k-1} > s \end{cases}. \tag{2.3}$$

There are two ways of noticing a failure of delivery for TCP, timeout and dupacks.

Time out Time out occurs when one packets individual RTT is higher than a constant times the mean RTT. This indicates a traffic jam and several lost packets and therefore c is reinitialized. How to calculate and measure the RTT is described in Appendix 2.A.

Dupacks Since the ACKs inform about expected deliveries, subsequent ACKs with the same number indicate a loss. TCP interprets three duplicate ACKs, dupacks, as a single packet loss and c is decreased, but more gently than after a timeout. In Example 2.1, one original ACK arrives for packet 4 and then two dupacks. Congestion is not detected based on dupacks in that case.

The threshold, s, is also updated when a data loss is detected form one of the ways described above. The updates, of c and s, after a congestion is detected are given by

$$\begin{aligned} c_k &= \begin{cases} 1 & \text{time out} \\ s & \text{3 dupacks} \end{cases} \\ s &= \max(f(t)/2, 2) \end{aligned}. \tag{2.4}$$

In summary, the TCP algorithm can be seen as in Algorithm 2.1.

Algorithm 2.1 (TCP flow control)

At each arrival time, t_k, the current congestion window, c_k is calculated as

$$c_k = \begin{cases} 1 & \textit{time out} \\ s & \textit{3 dupacks} \\ c_{k-1} + 1 & c_{k-1} \leq s \\ c_{k-1} + \frac{1}{c_{k-1}} & c_{k-1} > s \end{cases}.$$

The slow-start threshold, s, is updated on congestion according to

$$s = \max(f(t_k)/2, 2).$$

The flight size, $f(t)$, changes continuously as

$$f(t) = \int_{t-T_{RT}}^{t} r_{TCP}(\tau) d\tau.$$

The send rate, r_{TCP}, is given by

$$r_{TCP}(t) = \min(c_k - f(t), a_k), \text{ for } t \geq t_k,$$

where a_k denotes the received advertised window.

The specification of TCP has evolved over time and the most used version is the one called TCP Reno. Later version have more refined the mechanisms that trigger loss recoveries and also the updating of c_k during congestion avoidance. A short overview of these updates are given in Appendix 2.A.

2.2.2 Active Queue Management

The network layer protocols are present in all nodes in the network, e.g., Figure 1.2. Active Queue Management, AQM, is a set of suggested additions to the network layer. The protocols use dropping and marking to inform TCP (explicitly or implicitly) about congestion dangers. Since TCP lowers its send rate, r_{TCP}, when packets are lost, early dropping could be used to gently lower the send rate instead of causing a time out. This is supposed to damp oscillations in the queue length, prevent overflow and minimize empty queue time.

Random Early Detection, RED, is one suggestion of AQM, which uses a simple dropping profile based on the current queue length to decide which packets should be dropped. The longer the queue, the higher the probability of dropping an arriving packet. This technique was introduced in Floyd and Jacobson (1993) and has been widely adopted for further research. More specifically, when a packet arrives at the router at time, t_k, a filtered value, y_k, of the queue length, q_k, is calculated

$$y_k = (1 - \lambda)^m y_{k-1} + \lambda q_k. \tag{2.5}$$

Here $m = 1$ if $q_k > 0$ and otherwise is used to account for the time the queue was zero. If r is the service rate of the queue,

$$m = \begin{cases} 1, & q_k > 0, \\ (t_k - t_e)r, & q_k = 0. \end{cases}$$

The time t_e is used to denote the time when the queue became empty, i.e., the smallest t_e such that $q(t) = 0, t_e \leq t \leq t_k$. The probability, p, of dropping the new packet is then calculated as

$$p_k = \begin{cases} 0, & y_k \leq q_m, \\ \frac{p_M - 0}{q_M - q_m}(y_k - q_m), & q_m < y_k \leq q_M, \\ 1, & y_k > q_M, \end{cases} \tag{2.6}$$

i.e., according to a linear function between the constants q_m and q_M. The constant $0 < p_M < 1$ is suggested to be in the order of 0.1 for good performance. An example of p as a function of q is given in Figure 2.1 and Algorithm 2.2 summarizes the actions.

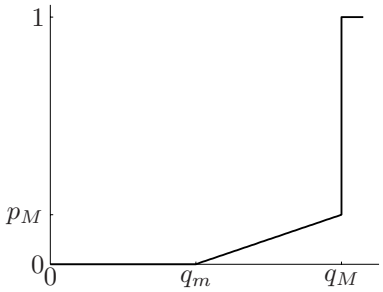


Figure 2.1 *The probability of drop as a function of the filtered queue length. The parameters q_m, q_M and p_M are design variables.*

Algorithm 2.2 (RED)

At each arrival time, t_k, the queue length q_k is measured. A filtered estimate, y_k, is calculated as

$$y_k = (1-\lambda)^m y_{k-1} + \lambda q_k,$$

with the scaling factor, m, being

$$m = \begin{cases} 1 & q_k > 0, \\ (t_k - t_e)r & q_k = 0, \end{cases}$$

where r is the service rate of the queue and

$$t_e = \min_{\tau}\{\tau : q(t) = 0, \tau \le t \le t_k\}.$$

The probability of drop, p, is then

$$p_k = \begin{cases} 0, & y_k \le q_m, \\ \frac{p_M}{q_M - q_m}(y_k - q_m), & q_m < y_k \le q_M, \\ 1, & y_k > q_M. \end{cases}$$

Explicit Congestion Notification, ECN, is described in RFC 3168. It is a technique which enables marking of packets instead of dropping, by using the TCP header. This requires changes in the TCP implementation as opposed to only dropping the packets, and the issue becomes how TCP should respond to these marks.

Several proposals for improvements on the dropping/marking decision mechanisms have been made. Most of them are based on the same structure as was proposed for RED and a few of them are described below.

BLUE The dropping probability is calculated based on packet losses and idle queue time, in a protocol called BLUE (Feng et al., 1999).

$$p_k = \begin{cases} p_{k-1} & t_k - t_s \leq t_F \\ p_{k-1} + d_1 & q_k > q_M \\ p_{k-1} - d_2 & q_k = 0 \end{cases}$$

where t_s is the time of the last change in p and t_F is the minimum time between changes. The constants d_1 and d_2 are the increments to be used.

AVQ, SVB Here a virtual queue is used to decide the drops in both the Adaptive Virtual Queue, AVQ (Kunniyur and Srikant, 2001), and the Stabilized Virtual buffer, SVB (Deng et al., 2002). In AVQ a virtual queue length $\tilde{q}_k$ is calculated, the virtual queue is served with a service rate $\tilde{r} \leq r$ and

$$\begin{aligned} \tilde{q}_k &= \max(\tilde{q}_{k-1} - \tilde{r}_{k-1}(t_k - t_{k-1}), 0) \\ p_k &= \begin{cases} 1 & \tilde{q}_k + 1 > q_M \\ 0 & \text{otherwise} \end{cases} \\ \tilde{q}_k &= \tilde{q}_k + p_k \\ \tilde{r}_k &= \max(\min[\tilde{r}_{k-1} + \alpha\gamma r(t_k - t_{k-1}), r] - \alpha, 0) \end{aligned}$$

For SVB the limit of the virtual buffer, cf. q_M, is adapted instead of the service rate, i.e., $\tilde{r} = r$ and $\tilde{q}_M \leq q_M$. The probability of drop is chosen as

$$p_k = \max(\frac{\tilde{q}_k - \tilde{q}_M}{\tilde{q}_k}, 0).$$

PI A proportional and integrating, PI, controller is tuned for the dropping probability based on a simplified physical model in Hollot et al. (2001b). The queue length is measured at times kT and the probability is updated as

$$p(nT) = a(q(nT) - q_{ref}) - b(q((n-1)T) - q_{ref}) + p((n-1)T)$$

q_{ref} is a value of the queue length that is desired and a, b and T are some constants. The drop probability for a packet arriving at time t_k is $p_k = p(nT), nT < t_k < (n+1)T$.

A nice comparison of a few algorithms can be found in Zhu et al. (2002). Basically, Eq. (2.6) is replaced in the different proposals, for some of them Eq. (2.5) as well.

There are also techniques that use RED functionality to achieve service differentiation. RFC 2474 describes differentiated services for IP traffic, **DiffServ**, where the RED parameters in Eq. (2.6) are tuned differently for each traffic class. DiffServ is also described in Kilkki (1999).

2.3 Radio Links

In the fixed Internet, the protocols below IP in the layer structure use no control mechanisms. Their task is solely to get an IP packet from one given point to another. When the radio medium is used for transmissions several new problems occur. One major problem is that packets can get damaged because of poor link quality, which has nothing to do with congestion. This dilemma is solved by introducing sub-layers in the radio link layer, each one with different new control tasks.

This description is based on the standard for UTRAN described in 3GPP TS 25.401. A schematic view of the UMTS architecture is shown in Figure 2.2. A Radio Network Controller, RNC, supervises several radio base stations, BS, and performs radio resource management for all the transmissions between these base stations and the User Equipments, UE. The transmissions over the wireless links use the Radio Link Control, RLC, protocol and the Medium Access Control, MAC, protocol to coordinate the transmissions.

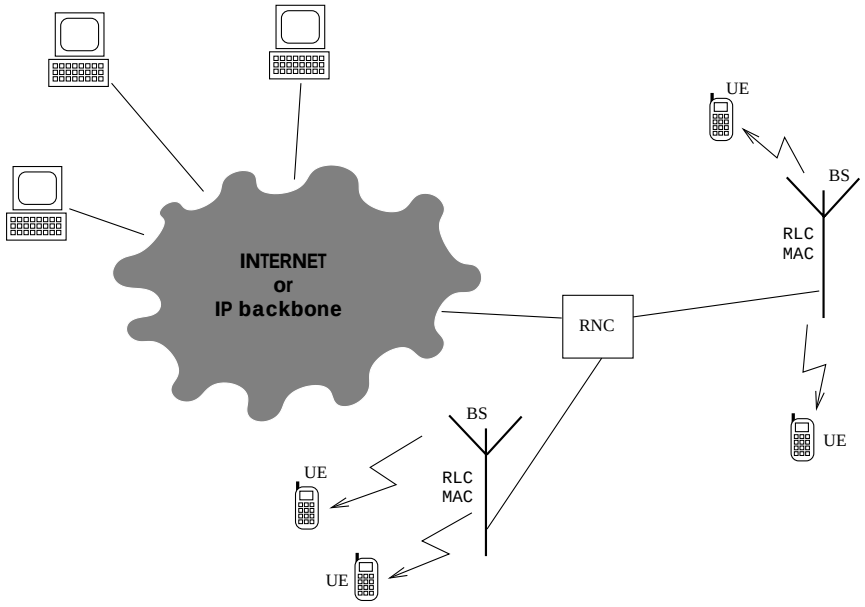


Figure 2.2 *A schematic view of the 3G architecture. The Radio Network Controller manages several base station and the transmissions over the wireless link uses the Radio Link Control and the Medium Access Control.*

The RNC is responsible for admission and congestion control and user mobility, which concerns adding, removing and moving connections to and from

the base station. In this section we focus on an existing connection going to the same base station during the whole transmission. Moving users can be modeled as one connection going down and another being established at a neighboring base station, which is why this assumption is no restriction here. The RNC functionality is specified in 3GPP TS 25.401, pp. 23—26. Apart from the services mentioned, the RNC handles transfer of user data and radio resource management, RRM. Each flow is assigned to a traffic class and each class is assigned a number of Quality of Service, QoS, attributes. These attributes are, e.g., maximum bit rate, guaranteed bit rate and transfer delay. The RNC controls these number to fulfill the class attributes for each flow. The QoS attributes are described in 3GPP TS 23.107, pp. 18—25.

The RLC protocol provides three modes for transmission: transparent, unacknowledged and acknowledged. Since the latter is the most interesting and the only one imposing control functionality, focus will be on the acknowledged mode. Between the base station and the UE, the RLC provides segmentation of IP data into smaller packets. Sequence numbering is used to enable retransmission of lost packets and in-sequence delivery to higher layers. Retransmissions are performed based on status reports from the receiver containing lists of received and missing packets. The status reports are sent upon a polling request from the sender (BS or UE). The 3GPP TS 25.322 contains the RLC specification.

MAC provides a few services listed in 3GPP TS 25.301, Sec. 5.3.1. It is responsible for unacknowledged transfer of the RLC packets and for reallocation of resources when requested from the RNC.

Below these new layers is the physical channel, the transmission of data using radio. Even this level includes control functions, the most important one being power control. Power control is used by the receiver (BS or UE) to control the transmission power of the transmitter (UE or BS) to make the perceived quality equal over time despite varying radio conditions. The power control is specified in 3GPP TS 25.214, pp. 13—30. The scope of this thesis does not include the power control mechanisms and the assumption is that they perform well enough to provide the allocated bit rate. This is an assumption the manufacturers probably won't argue with. A thorough description of the physical layer with the power control, and the other parts of the 3G system, can be found in Holma and Toskala (2000).

2.4 Control Structures and Interactions

The standards described previously imply some control structure for each layer. These descriptions are here translated into block diagrams and cross effects will be identified. As shown in Figure 1.2, packets on a network traverse queues and links. The queues add a queuing delay depending on their length and service rate and the links add a transmission delay. Therefore it is common to model the effects of a network on sent packets as a queue and

a delay.

Using block diagram representation (see Section 1.2), a pure delay,

$$y(t) = x(t - T),$$

is written using the Fourier transform e^{-sT}. The queue is fed with a certain arrival rate, r_a, and sends packets with a service rate, r_s. The difference between these two will be the increase in queue length. The total queue length at time t will be the accumulated difference from the initial time, i.e., the integral ($\frac{1}{s}$) of $r_a - r_s$,

$$q(t) = \int_0^t (r_a(\tau) - r_s(\tau))d\tau.$$

The block diagram representations of the delay and of the queue are shown in Figure 2.3.

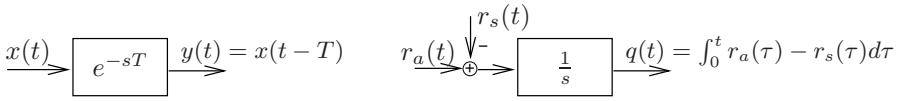


Figure 2.3 *A delay (left) and a queue (right) shown in block diagram representation. For the delay, x is the input and y is the output. For the queue, r_a is the arrival rate and r_s is the service rate.*

2.4.1 Application Layer without Flow Control

As stated previously, Section 2.1, the application layer can include flow control functionality but they are, in most cases, left for the transport layer. A simple mechanism observes the interface queue down to the TCP agent and performs some control based on the occupancy of it, see Figure 2.4. The difference in the rate of the application, r_{app}, and the rate of TCP, r_{TCP}, gives rise to the interface queue. The control of the application rate, C_{app}, is done based on the queue length and a reference value, q_{ref}. This type of control is performed during a simple file transfer, when FTP delivers data as fast as possible to the layer below. To deliver as fast as possible means that C_{app} in Figure 2.4 delivers r_{app} according to

$$r_{app} = \begin{cases} r_{max} & q_{ref} - q > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

and here $q_{ref} = q_{max}$ is the maximum buffer length between the layers. Whenever there is room, data is delivered downwards as fast as possible.

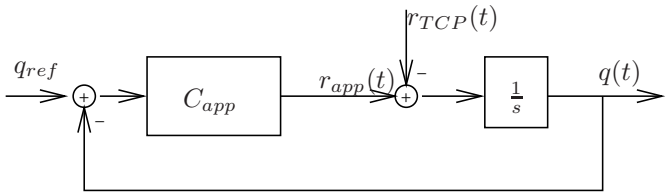


Figure 2.4 *The block diagram for an application, the control can be based on the interface queue down to TCP (or UDP). Notice that in this case the network is not visible to the application.*

2.4.2 A Streaming application

For a streaming application there are two scenarios: coding-decoding problem and following the critical link. For a video sequence, we can view the video stream as the reference signal that we want the receiver to follow (or play), perhaps with a certain fixed delay. The sender performs some controls, C_{send}, the data stream traverses the network or plant, P, and the receiver performs some controls, C_{rec}. This is shown in Figure 2.5 to result in open loop control. The structure is similar to the coding problem in radio transmission, where

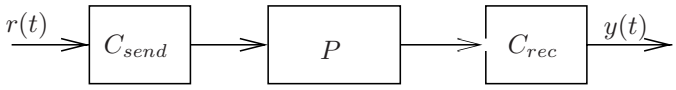


Figure 2.5 *When the video stream is the reference signal, $r(t)$, the situation is open loop control.*

a string of symbols is sent and should be decoded correctly at the receiver. However, the problems occurring in this system is different. In the network setup, data is delayed, reordered or lost. In the coding setup, data is not only delayed but every symbol can be corrupted without being lost. The problem can be solved by, e.g., training sequences. Perhaps a similar solution could be used for the streaming case.

At a more specific level the desired behavior might be to follow an unknown capacity, $r_{ref}(t)$ of the network, e.g., a varying radio link. The application sends with the same capacity as the network. To provide the same amount of information per time unit, different compression or coding have to be used. While following the capacity of the network, the transmission delay will remain constant. This scenario is shown in Figure 2.6. The send rate of the streamer is r_{str}. Decisions about the send rate, in the controller, C_{str}, are done based on when the queue length q differs from the wanted q_{ref}. A

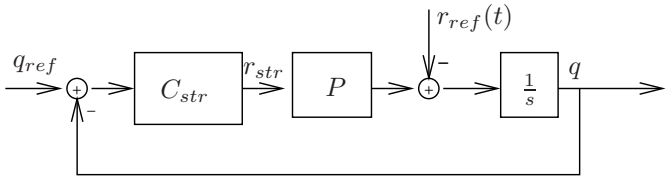


Figure 2.6 *The desired control situation for a streaming application. The send rate r_{str} should follow the rate on a critical link, r_{ref}. The plant, P, models the network effect on the output rate, such as delays and other effects, before it reaches the critical link. Decisions about the rate, C_{str}, are based on the queue length, q.*

simple situation would be $q_{ref} = q_{max}$ and the control acted when $q > q_{ref}$, i.e., on packet drops. If TCP is used below the application, the capacity of the network will be hidden behind TCP's send rate and the control will be difficult to perform.

2.4.3 Transmission Control Protocol

The control strategy in the transport layer is provided by TCP and was described in Section 2.2. The controller, C_{TCP}, acts based on notifications of deliveries and drops ($q > q_{max} = q_{ref}$) and it controls the send rate, r_{TCP}, into the network. Figure 2.7 shows the network as delays and a queue, which is how TCP sees it. The queue increases with the delivery rate, $r_{TCP}(t - T_1)$ and decreases with the service rate $r_q(t)$, i.e., $\dot{q}(t) = r_{TCP}(t - T_1) - r_q(t)$. The total round trip time of the network, T_{RT}, will depend on both the transmission delays, T_1 and T_2, and the queuing time, T_q. The queuing time can be found from solving the equation

$$\int_t^{t+T_q} r_q(\tau)d\tau = q(t),$$

i.e., the time it would take to empty a queue of length $q(t)$ with the specified send rate, r_q.

Since TCP Reno only acts based on explicit drops, the block diagram could be extended with a switch. The control signal back to C_{TCP} is 1 if $q > q_{ref}$ and 0 otherwise.

2.4.4 Active Queue Management

The additional functionality provided by the AQM mechanisms adds to the control loops. In Figure 2.8 a block diagram for an AQM scheme is shown.

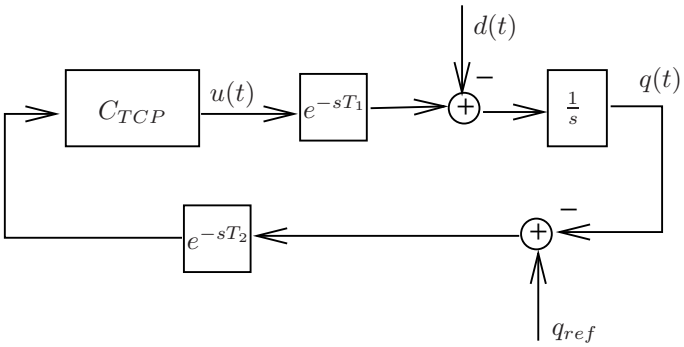


Figure 2.7 *The block diagram for a TCP connection.* $d(t)$ *is the available capacity for that flow in the queue and can be considered as a disturbance.* T_1 *and* T_2 *are transport delays for the connection and* T_q *is the queuing time, i.e.,* $T_{RT} = T_1 + T_2 + T_q$.

The queue length, q, is filtered through H. A control decision, u, is calculated in C_{AQM} based on q_{ref} and limited to produce the dropping probability, p. The dropping probability was before either 0 or 1, either there was room in the queue or there was not. Incorporating the AQM control with the TCP representation shows that AQM wants to change the reference for TCP, q_{ref}, from being the maximum queue length to something lower. The blocks should be included between the queue, q, and the return delay, T_2, in Figure 2.7.

The AQM block diagram shows that only drops are used for the control in TCP. This part is hidden in C_{TCP} in Figure 2.7. Several TCP connections will add to the queue length in the AQM queue and each drop affects one of them. If DiffServ, Section 2.2.2, is deployed, parallel AQM links with different controllers and different reference values will divide the TCP flows among them according to traffic class.

2.4.5 Radio Link Layer

In the link layer on the radio network a vast amount of control actions are performed. These are done by splitting an IP packet into smaller parts and using retransmission to successfully deliver the separate parts. This will cause a variable delay from the IP packet perspective. Radio Resource Management is performed over the radio link, which implies that the capacity can change. The IP packets are queued up at the RNC waiting to be delivered to the lower layers, RLC/MAC. A changing capacity will be seen as a varying service rate from the RNC queue. Therefore transmissions over a radio link can be mod-

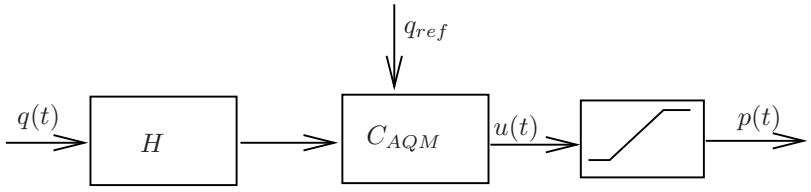


Figure 2.8 *The block diagram for a AQM queue. The blocks can be added to TCP's block diagram, Figure 2.7, between the queue, q, and the return delay, T_2.*

eled as a varying delay and a queue with varying service rate.

2.5 Summary

The block diagram representation is a way of compactly describing the main control functionality in a network. Together with the protocol specifications and standards, the specific controllers can be described for each case.

Some conclusions can be drawn from this block diagram interpretation of the network control components.

- For a streaming application that wants to follow the network capacity, TCP will hide the network variations behind its own control mechanisms.
- Adding AQM is a way of redefining the reference value for the controller in TCP.
- The layered approach when designing control algorithms is often adequate, but careful considerations have to be taken on the assumptions about the behavior of the underlying structure.

Appendices

2.A Different Versions of TCP

TCP has undergone some development since the first invention. The most commonly used version is TCP Reno, which was the foundation for the description in Section 2.2. There are a lot of extensions, special cases and even new versions present. This appendix contains a description of some of the existing standards and algorithms concerning the transmission control protocol.

TCP Tahoe was the first version of TCP and included only timeouts as detection of lost data. *TCP Reno* was improved with retransmission after three dupacks and later also with other improvements which was left out in the previous presentation. Retransmission after three dupacks is called fast recovery. Fast retransmit is often used together with fast recovery and it makes sure that the lost data is sent without waiting for permission by the congestion window.

Round trip time measurements It was obvious from the description in Section 2.2 that a measure of the round trip time is needed. Packet k is sent at time τ_k and the corresponding acknowledgment arrives at time t_k. A timer $T_k(t) = t - \tau_k$ is used for every packet. Upon each acknowledgment arrival a measurement

$$M(t_k) = T_k(t_k) = t_k - \tau_k$$

of the round trip time, RTT, for that packet is done. An average estimate, $R(t_k)$, of the RTT is calculated recursively based on the measurement $M(t_k)$,

$$R(t_k) = \alpha R(t_{k-1}) + (1-\alpha)M(t_k).$$

A timeout is defined as

$$T_{k+1}(t) > T_{TO}(t_k) = \beta R(t_k).$$

Recommended values for α and β is 0.9 and 2, respectively. An improved way of using the round trip time estimate is to also include the variations in R when deciding on the timeout timer, T_{TO}. This is done as follows:

$$\begin{aligned} R(t_k) &= \alpha R(t_{k-1}) + (1-\alpha)M(t_k) \\ D(t_k) &= \gamma D(t_{k-1}) + (1-\gamma)|M(t_k) - R(t_k)| \\ T_{TO}(t_k) &= R(t_k) + \beta D(t_k) \end{aligned}$$

where $\alpha = 3/4$, $\beta = 4$ and $\gamma = 7/8$ are recommended values. D is the average absolute deviation for the round trip time estimate. An extension to this is to distinguish between acknowledgments for single or retransmitted data. When the latter arrives R is not updated since it is impossible to know if the acknowledgment arrives from the first or the retransmitted packet.

Keeping overheads down Each packet produces an extra header and many small packets occupy more capacity than one large. Features have been added to TCP to prevent sending small amounts of data instead of waiting for more capacity or more data. E.g., acknowledgments should be collected in pairs if possible or used returning data packets to piggyback the acknowledgments. If there exists data, TCP should wait for increases in TCP send rate, r_{TCP}, instead of chunking the data into small pieces. This prevents the so called Silly Window Syndrome, which is observed when a small amount of data is sent as soon as the congestion window allows it.

TCP Vegas TCP Vegas (Brakmo and Peterson, 1995) tries to reduce the variations of the congestion window of the original TCP algorithm by using a new updating mechanism during congestion avoidance. First Vegas saves the shortest measured round trip time during the connection, $M_m = \min_k(M(t_k))$. It then calculates the expected throughput as

$$r_e = \frac{1}{M_m} \int_{t-M_m}^{t} r_{TCP}(\tau) d\tau$$

and for each packet k also the actual throughput as

$$r_a = \frac{1}{M(t_k)} \int_{t-M(t_k)}^{t} r_{TCP}(\tau) d\tau,$$

i.e., the total amount of sent data during the last round trip time divided by the round trip time. For each packet the difference in throughput is calculated, $r_d = r_e - r_a$. Then r_{TCP} is update based on this

$$r_{TCP}(t) = \begin{cases} r_{TCP}(t - M_m) - d_1, & r_d > b, \\ r_{TCP}(t - M_m) + d_2, & r_d < a, \\ r_{TCP}(t - M_m), & a \leq r_d \leq b, \end{cases}$$

where d_1 and d_2 are updating parameters and a and b are the accepted limits for the throughput difference, r_d. When $r_d 4$ is too low the send rate is increased and when it is too high the send rate is decreased. This means that R_{TCP} is used to try and keep the actual throughput, r_a, close to the expected one ,r_e. The update of r_{TCP} is done once every RTT, instead of for every new packet arrival.

Queue Management

3

Queue management is used to improve performance attributes, such as increasing throughput and damping oscillations. This is done by controlling network flows. Controlling flows means controlling queues. In this case it is queues on the Internet, typically core network routers or bridges between networks. It is routers that have a heavy load and are important for the overall performance of the network.

There are many areas and levels where queue management is on the agenda. This work focuses on the transmission, or flow, level, and it is based on current solutions, but the ideas easily carry over to independent settings. The approach is system theoretic, which is fairly new for network research. Several contributions have emerged recently that study network problems including queue management from a system theoretic viewpoint, e.g., Altman et al. (2000), Park et al. (2003), Hollot et al. (2001a), Low et al. (2002), Kunniyur and Srikant (2001).

This chapter starts by defining performance for a network and discuss performance measuring based on network queue lengths, in Section 3.1. In that section performance measures used by other researcher are described. Then modeling, identification and control of the queue length dynamics is discussed. In Section 3.2, models based on filtered measurements of the queue length is developed. The models are recursively identified to adopt to varying network settings. In Section 3.3, various controller are described and tested. The discussions are accompanied with explicit examples on a dataset from the Network Simulator (NS, 2.1b8a).

3.1 Performance Issues in Queue Management

Good network performance means different things depending on who is asked. The network provider might have a different measure than the user. To find

a good performance measure, the different issues that can be identified will be discussed.

For each user, transmission time and reliability are important. When TCP is used the reliability is assured and the only consideration is the transmission time. Thus, one issue in network performance is the time to transmit a certain number of bytes, measured on a per flow, per user, basis.

The network provider wants to fully utilize the available capacity and divide it among the users. This gives the requirement that there should always be packets processed by the slowest node. The issue of dividing the resources in an efficient way is not easy to resolve. Efficiency can be measured from both the overall network and from the single user, and optimizing the efficiency of the network might not be fair among the users and might cause users to change network provider. The two extreme cases are "total fairness", which means that every user gets exactly the same share of the resources, and "best effort", where no guarantees are given about delay or bandwidth share, and the matter has been discussed in various scheduling contexts (see, e.g., Kelly, 2003). We will not discuss the matter of resource sharing more here.

The two actors: the user and the network provider, here demand that can very well counteract each other. E.g., if one user gets access to a certain amount of the bandwidth all the time he will be very happy because he does not have to compete with anyone. On the other hand, the network will not be used very efficiently, since it is unlikely that the user is going to send at full speed day and night. The network provider would like keep the cost for the network low. If the network capacity is chosen to handle the average load, this could mean that users would experience large delays. Most traffic occurs at daytime and averaging the traffic would cause delays, even until the night.

Flow control is the main cause of varying the network load. As was described in the history review of Internet development in Section 1.1.1, the original flow control was performed at the end nodes, using a protocol now called TCP. After an immense increase in both size and number of available applications the Internet traffic does not behave as it did initially. In this section, the previously identified problems caused by TCP are described, together with the research effort that have been made based on these problems. Performance measurements in current research and based on the network queues are also discussed.

3.1.1 Problems with TCP

The transport protocol TCP has long been seen as the main trouble maker, when performance is concerned. A number of problems have been identified and corresponding solutions have been proposed. The research efforts have been grouped according to the problems they address. A more thorough description of each problem can be found in the corresponding references, where the effort is to solve them. The problems are also described in correspondence with the previous block diagram representation, see Figure 2.7.

1. TCP provokes packet losses to get an idea about the state of the network. We get implicit feedback only when something goes wrong.
   - This problem has been addressed with the inclusion of explicit feedback by Mascolo (1999).
   - This problem arises because of the signal that is measured by the controller, C_{TCP}. The controller only bases its decisions on packet drops.
2. The use of the available bandwidth oscillates. Thus the load of the network varies and the network becomes slower than it has to be.
   - This problem has been addressed using adaptive queue management, AQM, starting with RED and extensions such as ECN (Floyd and Jacobson, 1993, Floyd, 1994).
   - Many research efforts aim to improve performance of the initial AQM algorithms, e.g., Kunniyur and Srikant (2001), Deng et al. (2002)
   - In the block diagram, Figure 2.7, this is an effect of the design of the controller dynamics, i.e., the algorithm for calculating r_{TCP} based on measurements (drops).
3. There has been a shift in the Internet traffic mix. (Measurements come from Balakrishnan et al. (1998).)
   (a) Due to short transmissions, TCP is "always" (85%) in slow start.
   (b) Originally less than 1% of the packets were lost, now it is as much as $5-7\%$, which means a lot of retransmissions, (13% of the transmissions), most of which are due to timeouts.
      - The newer versions of TCP, such as Vegas, address this problem (Brakmo and Peterson, 1995)
      - The original design of C_{TCP} was done based on other assumptions about the rest of the system. A simple way is to say that the characteristics of the noise, or available capacity, r_s, have changed.
4. No difference is made between different traffic classes, such as FTP files, video streams or e-mail.
   - This falls under the area of quality of service, QoS, and DiffServ has been developed to solve this problem (Kilkki, 1999).
   - This is the responsibility of the network, and the problem lies in the construction of the controller C_{AQM}, Figure 2.8, and that the same settings are used for different flows or types of flows.

5. TCP always assumes that packet losses are due to congestion.
    - This is most important in wireless networks were packet losses can be caused by bad links or complete link failures. Some solutions have been proposed for so called Ad Hoc networks, where all nodes are both routers and potential receivers (Chandran et al., 2001, Liu and Singh, 2001).
    - Some of the proposed changes use this feature to improve performance, especially in adaptive queue management (Floyd and Jacobson, 1993, Kunniyur and Srikant, 2001).
    - This is due to the design of the controller, C_{TCP}, and how it interprets drops, once again.

Many solutions solve these problems by treating some of the other problems as features of TCP and include them in the knowledge of the system. In this way queue management is moved from the end nodes into the queues, routers. This family of techniques is called, as mentioned before, adaptive queue management, AQM. The idea is that it is easier to include changes in some of the routers and improve performance locally instead of having to include changes in most of the TCP implementations to improve anywhere. Since any TCP implementation can be used over any router it seems logical to place the improvements at the router of interest.

3.1.2 Measurements Used in Current Research

With all the research efforts put into the area of improving TCP, performance evaluations become more and more important. Some of the measures that are used in different contexts are briefly described below.

Sachs and Meyer (2001), Meyer (1999), Peisa and Meyer (2001)
use the time to transmit a certain number of bytes or the packet bit rate to evaluate mobile Internet access, to analyze TCP when used in GPRS and to analyze file transfers over UMTS, respectively.

Misra et al. (1999)
develop a model of the distribution of the queue length and the congestion window and use these to investigate TCP's performance.

Brakmo and Peterson (1995), Mascolo et al. (2001)
consider throughput (bytes/sec) and Jain's fairness index, Jain (1991), as performance measures to analyze the improvements of TCP when using TCP Vegas instead of TCP Reno and to evaluate the improvements achieved by a new version of TCP, respectively.

Hollot et al. (2001a)
models RED queue behavior using a probabilistic view and uses the

model to calculate RED parameters that fulfill stability and other control criteria.

Park et al. (2003)
studies the oscillations of the queue length and tries to damp them.

Altman et al. (2000)
uses the number of packet losses and the overall throughput as performance measures.

After the discussion above and previously in the section a personal view is presented for network performance measuring. The emphasize is put on the network queue since the earlier discussions showed that throughput, oscillations in the queue and other queue mechanisms are important in the performance discussion.

3.1.3 Using the Bottleneck Queue

We will now take the perspective of a network provider who wants satisfied customers. The focus will be on a specific part of the network, namely the bottleneck queue.

Definition 3.1 (The bottleneck queue)
The **bottleneck queue** *of a flow is the queue that is the slowest one for that flow and, consequently, the only queue where that flow has more than one packet at a time.*

There can of course be many bottlenecks in one network, but only one per flow and time instant. When investigating the network performance from the bottleneck queue perspective, we can observe, e.g.,

The time the queue is unused.
This indicates how efficiently the network is used. An idle bottleneck queue is a waste of capacity.

The fluctuations of the queue length.
This illustrates the ability to find a steady state of the network. The more stable the queue length is, the more stable is also the round trip time experienced by each connection. A stable queue will also make it easier for new connections if the total network is experienced as being in steady state, and thus scalability is improved.

The fraction of dropped packets.
This fraction represents how fast changes of the network are noticed. A larger number of dropped packets indicates a longer response time to overflows.

The throughput.
The throughout, also, signals if the queue is often empty. A higher throughput is of course better. It is also interesting for, e.g., the owner of this bottleneck queue if he is getting payed per delivered byte or packet.

The average queue length
A long average queue implies large round trip delays and a high probability to drop packets, while a short average queue means high probability for an unused queue.

Control objective: The conclusion is that we want to control the tails of the queue distribution to keep them small. If this is done drops and empty queues are avoided. We also want to make the distribution narrow around a not too high and not too low mean value. This assures small variations in the queue.

This, once again, points at the importance of having information about the actual queue.

3.2 Modeling and Identification of the Queue Dynamics

In order to achieve the control objective, the block diagram description, introduced inSection 2.4, will be used. Figure 3.1 is essentially the same as Figure 2.8 with P describing the dynamical relationship between the drop decision and the arrival rate to the queue, r_{nw}. This is the considered system and it is a connection of TCP links, seen in Figure 2.7. From the queue perspective this system is unknown: the number of flows, their respective round trip times and the type of TCP controller they obey. It is only known that, after some delay, the queue length decreases when packets are dropped and it decreases more rapidly if a larger number of packets are dropped at the same time. The queue send rate, r_s, is given and the AQM controller, C_{AQM}, bases the dropping decision on the filtered queue length, y.

The focus will be on a simple model of the queue length dynamics without any knowledge of the rest of the system P. The qualitative knowledge about P will be used in simple controllers in later sections. To exemplify and motivate the modeling choices throughout this section a typical data set is used.

Example 3.1 (Dataset) *Using the Network Simulator (NS, 2.1b8a) realistic network data can be produced. We measure the queue length, q, at the bottleneck node in a network. The network has between 0 and 100 sources sending TCP traffic over the bottleneck queue at the same time. The sending periods for the sources are randomly selected. Figure 3.2 shows the measured queue length over time. Inspection of the plots shows that the basic variation, for this network setup, is in the order of 0.6 Hz.* □

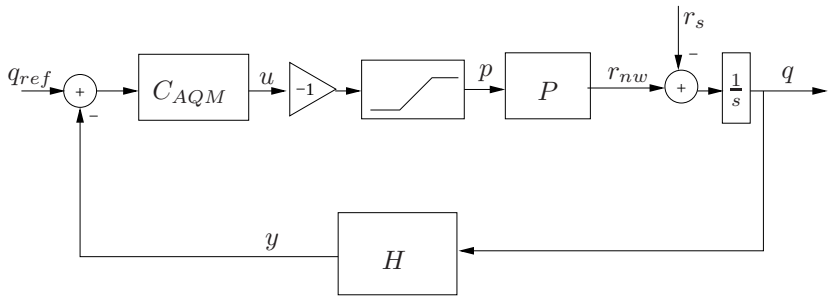


Figure 3.1 *Block diagram of the queue system. The AQM part is the same as in Figure 2.8. The filtered queue length is denoted y. The system P describes the network effects on the queue length after a certain drop is done.*

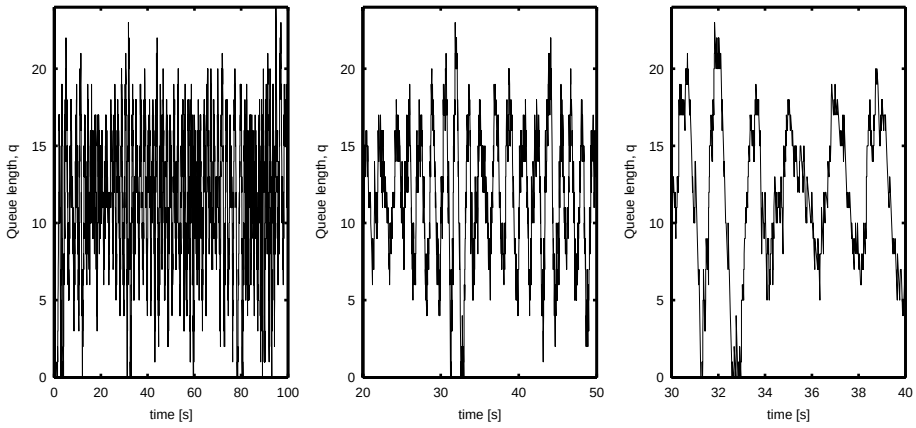


Figure 3.2 *The motivation data example from a bottleneck queue with the number of active transmissions varying from 0 to 100. Both the full time span and shorter segments are shown. The data is taken form the Network Simulator (NS).*

3.2.1 Frequency Analysis and Filtering

To describe the frequency content of a signal, $x(t)$, the Fourier transform, $X(f)$, is used,

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}dt. \tag{3.1}$$

This demands that $\int |x|^2 < \infty$. If the signal is sampled, $x(kT)\ k = 1, \ldots, N$, an unbiased estimate, X_s, is used

$$X_s(f) = \frac{1}{\sqrt{NT}} \sum_{k=1}^{N} x(kT)e^{-i2\pi fkT}. \tag{3.2}$$

$X_s(f)$ is what will be used in this work.

Example 3.2 (Frequency content) *Visual inspection of primarily the right part of Figure 3.2 shows that the signal is oscillating rapidly around the basic variation. Frequency analysis of the queue length shows, Figure 3.3, that a majority of the frequency content is around* 0.6 *Hz. The faster variations that were visible in the time plot are spread and not shown significantly here.* □

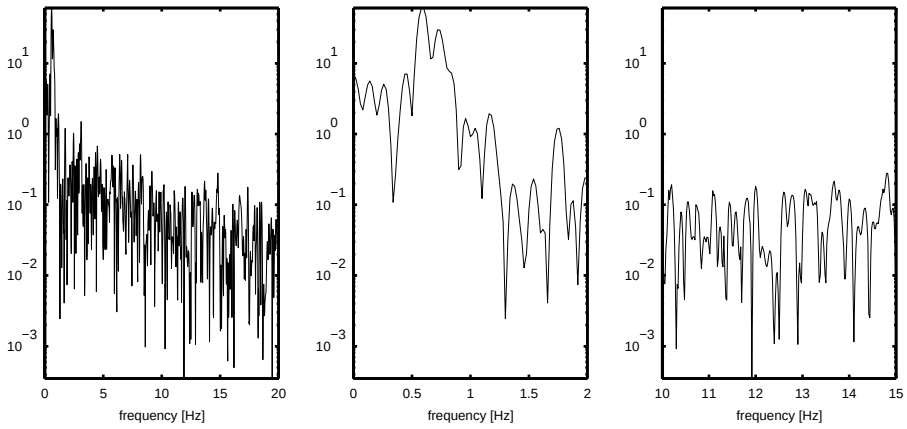


Figure 3.3 *The frequency content of the dataset described in Example 3.1.* $|X_s(f)|^2$ *is shown for the shortest time interval. Focusing on different frequency ranges shows the peak around* 0.6*. The high frequency variations are spread over several frequencies.*

Filtering of the signal is performed to reduce the fast oscillations and the noise.The control, both existing and new, can then be based on the long term variations (0.6 Hz). The noise will be reduced by a simple first order low pass filter,

$$H(s) = \frac{a}{s+a}, \tag{3.3}$$

which damps frequencies higher than $\frac{a}{2\pi}$ Hz more than 3 dB.

Example 3.3 (Filtering) *The main frequency content in the shorter time segment is below $f = 1$ Hz. With $a = 6.5$ the low pass filter effectively attenuates frequencies above 1.03 Hz more than 3 dB. The result is clearly visible in a comparison of the two signals, see Figure 3.4* □

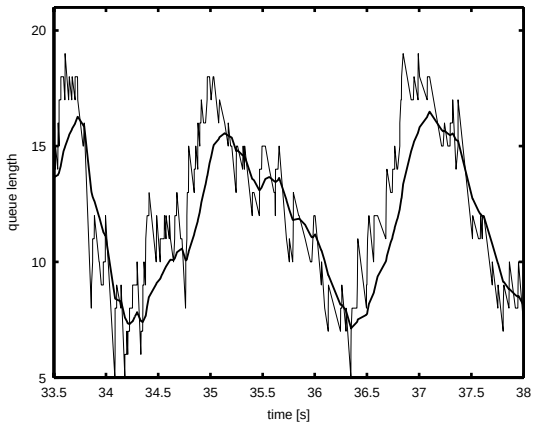


Figure 3.4 *Comparing the filtered signal (thick line) and the original one (thin line) for a short time.*

3.2.2 AR Modeling of Nonzero-Mean Signals

Auto Regressive, AR, models are models of a signal based only on the signal itself. The basic model equation contains the signal and its derivatives (or time shifts in the sampled case),

$$y^{(n)} + a_{n-1}y^{(n-1)} + \ldots a_1\dot{y} + a_0 y = 0.$$

Example 3.4 (AR model) *Variations with a certain frequency can be captured using AR modeling. A second order model*

$$\ddot{z}(t) + a_1\dot{z}(t) + a_2 z(t) = 0$$

can capture one frequency. If $z(t) = A\sin(\omega t)$ *it is easy to verify that the parameters have to be* $a_1 = 0$ *and* $a_2 = \omega^2$*. Studying a sampled model*

$$z(t) + \tilde{a}_1 z(t-T) + \tilde{a}_2 z(t-2T) = 0$$

calculations show that $\tilde{a}_1 = -2\cos(\omega T)$ *and* $\tilde{a}_2 = 1$ *for the same signal. This examples motivates why AR models are good at describing signal variations with correspondence to the frequency content. It also shows that there is a high correspondence between the structure of continuous and sampled models.* □

Because of the correspondence between continuous and discrete time indicated in Example 3.4, only time discrete models will be considered. For an AR model of order $n = 2m$, m frequencies can be captured. To include uncertainties the equation is not believed to hold exactly, a noise $v(t)$ is included,

$$z(t) + a_1 z(t-T) + \ldots + a_n z(t-nT) = v(t).$$

In order to model variations around a non-zero level $y(t) = z(t) + c$ can be used. This gives

$$\begin{aligned}(y(t) - c) + a_1(y(t-T) - c) + \ldots + a_n(y(t-nT) - c) &= v(t)\\ y(t) + a_1 y(t-T) + \ldots + a_n y(t-nT) &= b + v(t)\end{aligned}$$

with $b = (1 + a_1 + \ldots + a_n)c$. Gathering the unknown parameters in θ and the known signals in $\varphi(t)$ gives us

$$\begin{aligned} y(t) &= \varphi^T(t)\theta + v(t)\\ \varphi(t) &= \begin{pmatrix} -y(t-T) & \ldots & -y(t-nT) & 1 \end{pmatrix}^T\\ \theta &= \begin{pmatrix} a_1 & \ldots & a_n & b \end{pmatrix}^T \end{aligned} \tag{3.4}$$

and we can consider recursive estimation of the parameters.

Frequency estimation

As was indicated in Example 3.4, an AR model will attempt to describe the dominating frequencies of the signal. The frequencies can be found by studying the angles for the complex roots, x_0 of the characteristic polynomial,

$$x^n + a_1 x^{n-1} + \ldots + a_n = 0.$$

The frequencies are then

$$f = \frac{\text{angle}(x_0)}{\pi}\frac{1}{2T}, \tag{3.5}$$

where T is the sampling interval as before.

3.2.3 Recursive Identification

In on-line applications with changing characteristics adaptive models are needed. Recursive identification is then the only reasonable way to estimate the dynamics. Two possible algorithms for recursive identification of the model parameters are described, the recursive least squares, RLS, and the Kalman Filter, KF (see, e.g., Ljung and Söderström, 1983). Both algorithms use the model

$$y(t) = \varphi^T(t)\theta(t) + v(t),$$

with φ capturing all data, θ being the unknown parameters and $v(t)$ being additive i.i.d. noise. The assumption also includes a time varying system, which means that the parameters can change over time. This can be described as

$$\theta(t+T) = \theta(t) + w(t),$$

with $w(t)$ being i.i.d. noise describing the parameter changes. This model is called a random walk.

Let $\hat{\theta}(t-T)$ be the estimate of $\theta(t-T)$. The estimate of the signal value $y(t)$ is calculated

$$\hat{y}(t) = \varphi^T(t)\hat{\theta}(t-T)$$

and also the error of the estimate is used

$$\varepsilon(t) = y(t) - \hat{y}(t).$$

The updating algorithms for RLS and KF are similar but still significantly different. In RLS a forgetting factor, λ, is used to decide the importance of old measurements. For each updating time nT, RLS minimizes the cost function

$$V_n(\theta) = \sum_{k=1}^{n} \lambda^{n-k}(y(kT) - \varphi^T(kT)\theta(nT))^2$$

to find $\theta(nT)$.

Algorithm 3.1 (RLS)

Calculate a new estimate of θ at time t according to

$$\begin{aligned}
\varepsilon(t) &= y(t) - \varphi^T(t)\hat{\theta}(t-T) \\
P(t) &= \frac{1}{\lambda}\left(P(t-T) - \frac{P(t-T)\varphi(t)\varphi^T(t)P(t-T)}{\lambda + \varphi^T(t)P(t-T)\varphi(t)}\right) \\
K(t) &= P(t)\varphi(t) \\
\hat{\theta}(t) &= \hat{\theta}(t-T) + K(t)\varepsilon(t)
\end{aligned}$$

Here λ is the forgetting factor chosen as $\lambda < 1$.

In KF the model of the noises, w and v, influences the adaptation rate to new measurements.

Algorithm 3.2 (KF)

Calculate a new estimate of θ at time t according to

$$\begin{aligned}
\varepsilon(t) &= y(t) - \varphi^T(t)\hat{\theta}(t-T) \\
P(t) &= P(t-T) + Q(t) - \frac{P(t-T)\varphi(t)\varphi^T(t)P(t-T)}{\varphi^T(t)P(t-T)\varphi(t) + R(t)} \\
K(t) &= \frac{P(t)\varphi(t)}{\varphi^T(t)P(t)\varphi(t) + R(t)} \\
\hat{\theta}(t) &= \hat{\theta}(t-T) + K(t)\varepsilon(t)
\end{aligned}$$

$Q(t) = E[w(t)w^T(t)]$ models the covariance of the noise w, i.e., it expresses the variations of the parameters. Similarly, $R(t)$ models the noise variance in the measurements, i.e., $R(t) = E[v(t)^2]$.

After an estimate of the current model parameters is obtained, future values of the signal, y, can be predicted. For a model with sample time T, one step ahead predictions are calculated as

$$\hat{y}(t+T) = \varphi^T(t+T)\hat{\theta}(t).$$

Studying Eq. (3.4) we see that $\varphi(t+T)$, in the expression above, contains values of y from time t and older. The estimated parameters $\hat{\theta}(t)$ are the most recent estimate. Predictions further away into the future can also be calculated, but of course with larger uncertainty, both because of noise and of model changes. Looking k steps ahead can be done with

$$\hat{y}(t+kT) = \varphi^T(t+kT)\hat{\theta}(t).$$

In the case of an AR model, when future values of y are needed in φ the predicted values are used in stead. The accuracy of the prediction is thus effected by the accuracy of earlier predictions as well as of the accuracy of the model estimation.

Example 3.5 (AR modeling) *Modeling is done using the data set described in Example 3.1. Adapting a fifth order AR model to data yields a good fit. A segment of the filtered queue length, y, and the estimated model output, $\hat{y}$, is shown in Figure 3.5. For this example RLS with $\lambda = 0.999$ was used. We compare with estimating using the current measurement, i.e., $\hat{y}(t+T) = y(t)$. To measure the goodness the relative distance between the measurements and the estimates is calculated,*

$$R = 1 - \sqrt{\frac{\sum |y - \hat{y}|^2}{\sum |y|^2}}.$$

A value close to 1 *indicates a good fit. For the AR model,* $R = 0.83$ *and for the shifted model,* $R = 0.75$*. From this we see that the knowledge about the data is significantly increased by the identified AR model, compared to the shifted model.* □

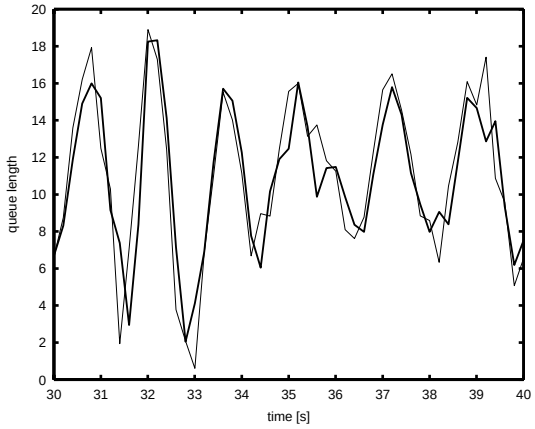


Figure 3.5 *Comparison of filtered queue length (thin) and model output (thick).*

3.3 Control of the Queue Length

In Section 3.1.1, a number of problems with TCP were discussed. The control of the queue dynamics that will be discussed here addresses problems 2 (bandwidth oscillation) and 3b (more packet losses). The control technique utilizes the identification procedure discussed earlier and considers problem 3b (packet losses always lowers the send rate) as system knowledge that can be used in the control design.

As discussed previously, the goals in queue management is to attenuate oscillations and to keep the queue from emptying and overflowing. The previous section indicated that the queue dynamics is well described by an AR model. To reduce the oscillations derivative action is needed and a few simple versions will be described here. The control is primarily based on the ideas in the existing RED algorithm.

In standard RED, the packet dropping probability first is calculated and then the packet is dropped or accepted based on this probability. A natural extension is to only drop packets when the queue is increasing, which a very

simple inclusion of derivative action into the controller. Derivative action changes the control decision based on the derivative, and a large derivative typically yields a higher control signal.

The idea with the adaptively estimated AR model is to predict the evolution of the near future queue length and use this prediction as a base for control decisions. This can work because of the round trip time. The TCP controller will not react immediately on a drop decision.

Combining prediction and this type of derivative action with the basic RED algorithm produces four alternatives, which are described below. For reference the saturation operator is first defined:

$$\mathrm{sat}(x) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x < 1 \\ 1, & x \geq 1 \end{cases} . \tag{3.6}$$

The saturation can also be seen in the block diagram Figure 3.1, when the control signal, u, is transformed into the dropping probability, p.

1. Proportional control

Ordinary RED essentially calculates the drop probability, p, as an affine transformation of the filtered queue length, y,

$$\begin{aligned} u(t) &= k(y(t) - r) \\ p(t) &= \mathrm{sat}(u(t)). \end{aligned}$$

2. Proportional control with prediction

The RED-functionality is applied to the predicted value, $\hat{y}$,

$$\begin{aligned} u(t) &= k(\hat{y}(t+T) - r) \\ p(t) &= \mathrm{sat}(u(t)). \end{aligned}$$

3. Proportional and derivative control, PD

A positive value of the dropping probability is used only if the derivative is positive, i.e., the queue length increases,

$$\begin{aligned} u(t) &= \begin{cases} k(y(t) - r), & \dot{y}(t) > 0 \\ 0 & \dot{y}(t) \leq 0 \end{cases} \\ p(t) &= \mathrm{sat}(u(t)). \end{aligned}$$

4. PD with prediction

As above but with based on the predicted value instead of the measured,

$$\begin{aligned} u(t) &= \begin{cases} k(\hat{y}(t+T) - r), & \dot{\hat{y}}(t+T) > 0 \\ 0 & \dot{\hat{y}}(t+T) \leq 0 \end{cases} \\ p(t) &= \mathrm{sat}(u(t)) \end{aligned}$$

For all cases, the constant r can be seen as the reference value towards which the queue length is controlled. The difference between the first two approaches can be seen as the difference between using the shifted model or the identified AR model in Example 3.5. To implement the derivative action an estimate of the derivative is needed. The problems with this is discussed in Section 3.3.1.

The derivative action can be included in other ways, the more typical setup gives a fifth alternative.

5. Explicit D action

An extension is to use D action more explicitly

$$u(t) = k_p(y(t) - r) + k_d\dot{y}(t)$$
$$p(t) = \text{sat}(u(t))$$

in which case it becomes important to have reliable derivative estimates, once again. This alternative can of course also be used with predictions.

3.3.1 Estimation of the Derivative

To be able to use the derivative control presented above, an estimate of the derivative $\dot{y}$ must be calculated. There are a few commonly used non-parametric estimates. We assume regular measurements of the signal $y(t)$, separated by T. The estimate of the derivative $\dot{y}(t)$ is denoted $\Delta y(t)$. When only sampled data is available some kind of difference approximation is needed. Three types are presented here:

$$\begin{array}{ll} \text{Backward} & \Delta y(t) = (y(t) - y(t-T))/T \\ \text{Forward} & \Delta y(t) = (y(t+T) - y(t))/T \\ \text{Central} & \Delta y(t) = (y(t+T) - y(t-T))/2T \end{array} \tag{3.7}$$

For recursively measured data only backward approximation is possible.

The more complex Tustin approximation of $\Delta y(t)$ is found by solving

$$\Delta y(t) + \Delta y(t-T) = \frac{2}{T}(y(t) - y(t-T)) \tag{3.8}$$

iteratively. Tustins approximation demands that $\Delta y(t)$ is stored to enable further estimates. The Tustin operator, Δ, is an unstable filter,

$$\Delta y = \frac{2(1-q^{-T})}{T(1+q^{-T})}y(t),$$

of the measurements, y. A stabilized version is

$$\Delta y(t) = \frac{2(1-q^{-T})}{T(1+aq^{-T})}y(t) \tag{3.9}$$

with $a < 1$.

Since we have a model of the signal, $y(t) = \varphi^T(t)\theta$, we can produce a parametric estimate of the derivative. Defining the operator Δ as any difference operator of the kind discussed above and using the model structure from Eq. (3.4) we get

$$\Delta y(t) = \Delta\varphi^T(t)\theta = \begin{pmatrix} -\Delta y(t-T) & -\Delta y(t-2T) & \dots & 0 \end{pmatrix}\theta. \tag{3.10}$$

It is possible to use the previous measurements and produce a model based estimate of the derivative.

Example 3.6 (Derivative approximation) *Using the signal and model parameters from Example 3.5, the result from the estimation methods can be illustrated. For the Tustin calculation, $a = 0.5$ was used and, in Eq. (3.10), a central approximation,*

$$\Delta y(t) = \frac{1}{2T}(y(t+T) - y(t-T)),$$

was used for Δ. Figure 3.6 shows the sign of the estimates together with the original signal. To the right is the derivative approximation. The parametric estimate is smoother than both the other but the Backward difference is very close. The sign is almost always correct for all the methods, but the result is improved with backward difference and even more with the parametric method. Since the central approximation is better than the backward difference, the parametric method gives two advantages, both a smoothed estimate because more measurements are used and it is possible to use the central approximation. □

The conclusion is that Backward difference seems to be fairly good but with a good parametric model the estimate can be even better.

To improve a derivative approximation the following can be noted. Assuming a function $F(y,T)$ is used to calculate the derivative of $y(t)$ using a difference approximation. The error is then proportional to T^p,

$$F(y,T) = F(y,0) + cT^p + O(T^r),$$

where $F(y,0)$ is the true value of the derivative. If $F(y,2T) = F(y,0) + 2^p cT^p + O(T^r)$ is calculated as well,

$$F_2(y,T) = F(y,T) + \frac{1}{2^p - 1}(F(y,T) - F(y,2T)) = F(y,0) + O(T^r)$$

can be used to approximate the derivative. $F_2(y,T)$ gives an error proportional to T^r with $r > p$. For the central approximation the error is proportional to T^2 and for backward difference to T. In basic numerical analysis this is called Richardson extrapolation (Dahlquist and Björck, 1974).

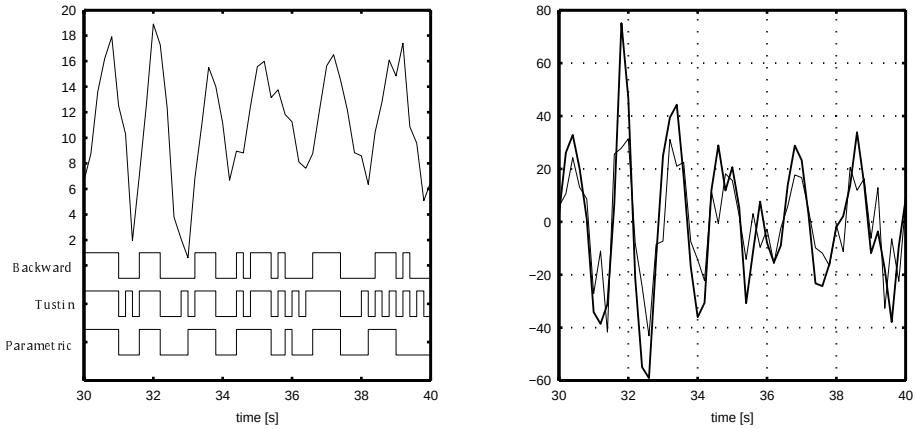


Figure 3.6 *The derivative estimation methods are compared. In the right plot the derivative estimates are shown, backward (thin) and the parametric method (thick). The estimate using Tustin is not shown since it is very noisy compared to the others. In the left plot the sign of the derivative is shown together with the original signal (top). The values are shifted vertically for separation.*

3.3.2 Simulation Results

The performance of the described control strategies are analyzed using simulations in the Network simulator, NS, 2.1b8a. A special version of the RED queue module has been implemented, which allows testing of both identification and control performance.

The setup is a network with varying number of flows and only one queue is studied. For comparative studies, identical realizations of the network are used, i.e., the start and stop times for the flows are fixed in each simulation to get comparable and reproducible conditions for all controllers.

Since it is not likely that one of the update times t_i matches $t-kT$, $k = 1, 2, \ldots, n$, simple linear interpolation between the two surrounding measurement points is used whenever an old value of y is needed.

Algorithm 3.3 (Linear Interpolation)

1. *From the sequence of measurements, y_i, at times t_i, find the values t_{lo} and t_{hi} surrounding the time $t - kT$, i.e.,*

$$t_{hi} = \arg\min_i (t_i > t - kT) \text{ and } t_{lo} = \arg\max_i (t_i < t - kT).$$

2. *Let* y_{lo} *and* y_{hi} *be the values* y_i *corresponding to the time points chosen.*
3. *Calculate the linear interpolation,* y_{me}*, for time* $t - kT$

$$y_{me} = \frac{(t_{hi} - (t - kT))\, y_{lo} + ((t - kT) - t_{lo})\, y_{hi}}{t_{hi} - t_{lo}}.$$

4. *Use* y_{me} *as an estimate of* $y(t - kT)$.

Improvements can be realized using, e.g., the techniques described in 4.2.2, but they have not been implemented.

Because of the uncertainty in the derivative calculation, controller number 5 has not been implemented. All the other control versions can be easily implemented using the generic Algorithm 3.4, where a simple Backward difference estimate is used for the derivative.

Algorithm 3.4 (Generic AQM implementation)

$$y_d = \frac{y_1 - y_2}{T_d},$$

$$u = \begin{cases} M_p \frac{y_1 - m_q}{M_q - mq}, & y_d > 0, \\ 0, & \textit{otherwise}, \end{cases}$$

and

$$p = \mathrm{sat}(u).$$

The parameters y_1*,* y_2 *and* T_d *are used to get the different versions from p. 42, e.g.,* $y_1 = y(t)$*,* $y_2 = 0$ *and* $T_d = y_1 - y_2$ *gives proportional control, (Type 1) but* $y_1 = \hat{y}(t + T)$*,* $y_2 = y(t)$ *and* $T_d = T$ *gives PD with prediction (Type 4).*

During the simulations $M_p = 0.2$, M_q is the total queue length and $m_q = M_q/2$.

The question arises what the suitable sampling time, T, is. Using the estimated model, the oscillating frequency, f, can be found and T should then, as a rule of thumb, be chosen as a tenth of the period, i.e., $T \approx \frac{1}{f}\frac{1}{10}$. Spectral analysis can also be used to find the main frequency, this is discussed in Section 4.2. Simulations indicate that it is the send rate of the queue that is the main cause of changing frequency, see Figure 3.7. Here, the scenarios have been adapted to work for $T = 0.2s \approx \frac{1}{10}\frac{1}{0.6}$, the results are valid no matter what value we choose but the choice is still a design question. To improve performance automatic detection of an accurate time constant should be considered.

A few performance measures are summarized in Table 3.1. These numbers are calculated for one single run with the exact same setup for all four controllers. In Figure 3.8 the resulting queue length is shown for the four controllers. The following observations can be made from Table 3.1 and Figure 3.8:

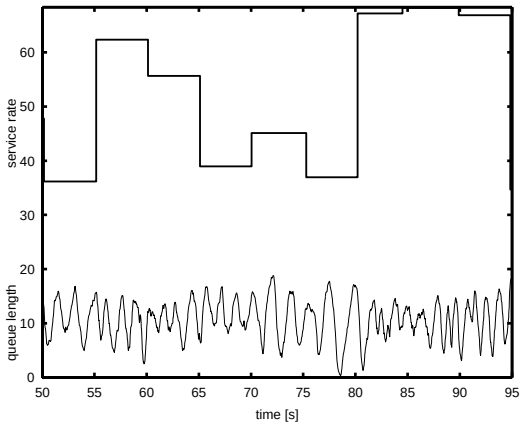


Figure 3.7 *Comparing different send rates (top thick line) with the corresponding queue length (lower thin line). A lower send rate in the queue seems to slow down the oscillations of the queue length.*

Type	Drops early+hard= tot	Sent unique packets
1	$55 + 250 = 305$	3260
2	$66 + 247 = 313$	3569
3	$82 + 207 = 289$	3126
4	$62 + 235 = 297$	3342

Table 3.1 *Comparison of the different control schemes, see p. 42 for a description. The number of drops and the total throughput is studied.*

- Performance (number of sent packets, throughput) is improved by increasing the total number of drops.
- Prediction can improve the throughput considerably, both in basic RED (cf. 1 to 2) and in RED with D action (cf. 3 to 4).
- The simple derivative action used does not improve the performance regarding throughput.
- In both cases the derivative action reduces the number of hard drops, cf. 1 to 3 and 2 to 4, which also was part of the stated control objective.

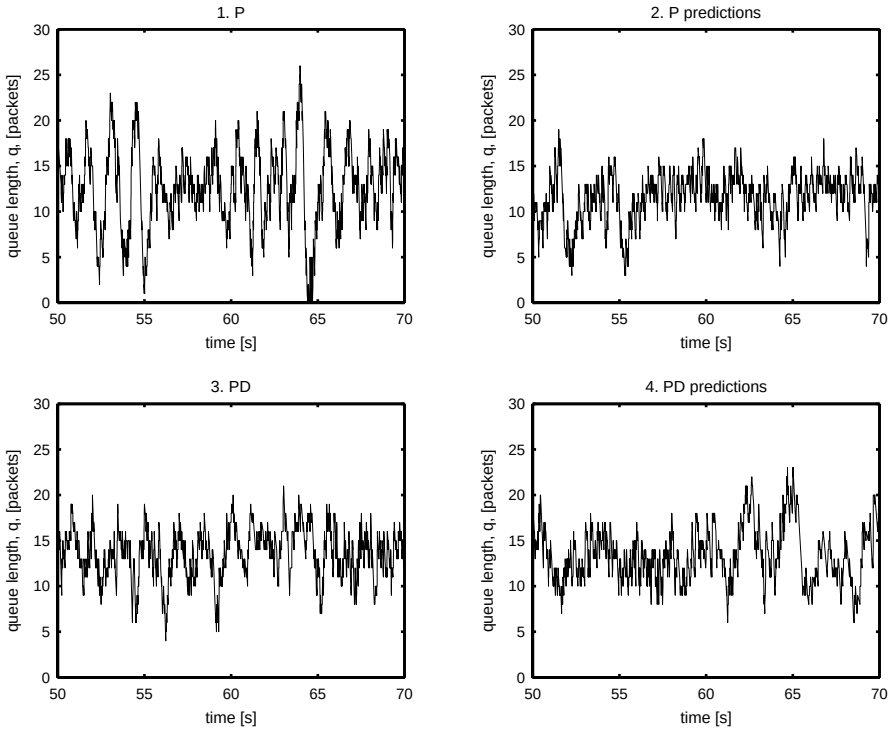


Figure 3.8 *A segment of the queue length, q, when using the four different controllers. Both differences in the height and the freqeuncy of the variations can be seen.*

- The derivative action lowers the span of the main variations in the queue length, which explains the reduced number of hard drops.
- Using predictions reduces the number of highest peaks in the variations and increases the frequency. Reducing the highest peaks should also indicate a lower number of hard drops, while an increased basic frequency indicates that using the predictions removes the frequcies captured in the model.

Closed-loop Identification

The system performs in closed loop and the output affects later inputs. This is cumbersome when system identification is used. The model identification is based on finding variations and basic frequencies and the control goal is to damp the oscillations. This means that a perfect controller will make it hard

to estimate the model and a bad model will make the controller worse. These properties will make it impossible to find a perfect controller for this system. However, the delay between a packet loss and a reaction in the queue length is so large, that it becomes impossible to control the system perfectly because of this. Therefore, it is worth the effort to use model based control for the network system.

3.4 Summary

Network performance can be studied from a queue perspective. The control objectives for the queue is to damp oscillations, prevent overflow and avoid an empty queue.

Queue management can improve network performance by intelligently dropping packets. The standard algorithm, RED, for AQM can be improved by using prediction. To damp queue length oscillations derivative action has been tried. The throughput was not improved using the simple derivative action, but ascillations were damped. Explicit derivative action was not included in the evaluation runs.

It was indicated that model based estimation of the derivative can improve the estimate. In the simulation run, only a difference approximation was used to find the derivative.

The improved controller was based on accurate modelling of the queue length dynamics. It was indicated that a biased AR model was well suited for this purpose.

Nonuniform Signal Processing

The application of interest, flow control in network queues, is based on events, namely the arrival of data packets. Upon such an arrival all calculations are triggered. This causes problems when design is done in the continuous time domain and the connections between the two frameworks need to be established. As discussed in Chapter 3, a number of calculations can be of interest based on the sequence of event based measurements. This chapter will discuss various approaches to identification based on nonuniform samples.

When sampling a signal a transition is made from the continuous time to the discrete time domain. A lot of calculations on sampled signals are based on assumptions about the original continuous signal and could not be performed without it. The modeling in the previous chapter, was based on the assumption of sinusoidal variations in the measured signal. This resulted in a dynamic model of the queue length variations. The calculations were performed using measurements collected at the packet arrivals, i.e., nonuniform sampling of the queue length. The results were presented as if the queue length was uniformly sampled. Everything regarding the transition from nonuniform to uniform sampling was suppressed. This chapter will highlight these hidden steps.

4.1 Sampling

For nonuniformly sampled signals many things differ from uniform sampling. There are a few strong connections though. Both the differences and the similarities will be discussed here. First we define the two terms:

Definition 4.1 (Sampling)
When a continuous signal $y(t)$ is measured at times t_k, the signal is **sampled**. *This gives corresponding signal samples $y_k = y(t_k)$. Different types of sampling can be identified*

Uniform sampling *when the sample points are equally spaced,* $t_k = kT$.

Nonuniform sampling *when the sampling points are unequally spaced.*

Jittered sampling *when* $t_k = kT + \nu_k$ *and* ν_k *are random variables.* Periodic jitter *is when the values of* ν_k *are unknown but are repeated periodically.*

Additive random sampling *when* $t_k = t_{k-1} + T_k$ *and* T_k *are random variables.*

Figure 4.1 shows additive random sampling of a signal y. *Uniform sampling can be seen as a special case of nonuniform sampling. In nonuniform sampling, the time points,* t_k, *can be chosen freely or, as in the network queue application, associated to events. Sometimes the term* **event sampling** *is used to denote the special case when events control the sampling times.*

Periodic jittered sampling occur, e.g., in AD converters (Elbornsson, 2003) and in cars (Persson, 2002).

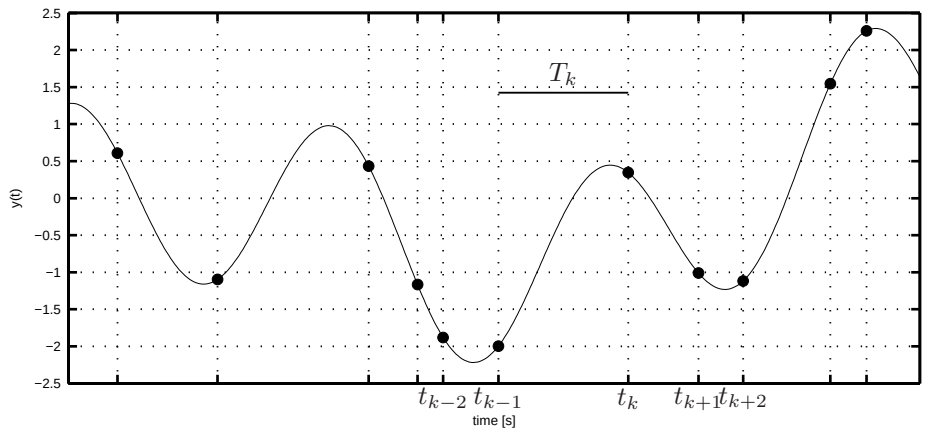


Figure 4.1 *A continuous signal, which is nonuniformly sampled.*

Earlier work on network queue modeling has defined the arrival process to the queue to be an additive random Poisson process. This means that the inter arrival times, T_k, are exponentially distributed. There also exist many studies where this is shown not to be true. (See, e.g. Paxson and Floyd, 1994, and the references therein.) The characteristics of network traffic is self-similar (same behavior on different time scales) and includes a long-range dependence (the autocorrelation decays hyperbolically). This is why assumptions about independent inter arrival times are not perfectly adequate for a

longer time scale. Several models have been proposed to describe the dependence over time for the inter arrival times, and superpositioned Poisson processes have been used successfully (Andersson, 2002, Andersen and Nielsen, 1998). Andersson (2002) also showed that network traffic can be locally modelled as a Poisson process.

Adopting the view with a probability distribution defining the arrival times, probability calculations can be performed for the measured signal. In this work, inter arrival times are assumed to be independent on a short time scale. This work will not focus on chosing the correct probability distribution, but the examples will be made using simple densities, such as rectangular or exponential distribution. It is possible to extend the results to any probability distribution.

Nonuniform sampling is described in Papoulis (1977), Bilinskis and Mikelsons (1992), Marvasti (1987). Research is focused on how to maximize alias frequency suppression by choice of sampling instants. In Bland and Tarczynski (1997), an empirical motivation to sample nonuniformly is given with some user guidelines. In Papenfuss et al. (2003), an algorithm for hardware implementation of jittered sampling ($t_k = kT + n_k$ and n_k is a random process) promises 40 times the bandwidth of the corresponding uniform sampling process. Other work is focused on optimal signal recovery from nonuniform samples. Algorithms for recovery of bandlimited signals, when $t_k = kT + \nu(t_k)$, with $\nu(t)$ being limited, are given in Marvasti (1996), Marvasti et al. (1991).

In Section 4.2, frequency analysis for nonuniformly sampled signals is discussed. Approximations of the Fourier transform are proposed based on different calculations on the nonuniform samples. Results based on stochastic sampling are discussed in Section 4.3. In Section 4.4, the different transform approximations are evaluated using the representation of a single frequency. If the frequency spectrum is found, a continuous time model can be adopted to data. A continuous time model can be preferred over a uniform discrete model, which requires a selection of the sampling interval. The ordinary identification techniques are discussed for nonuniform sampling in Sections 4.5 and 4.6. First a continuous system is nonuniformly sampled and the impact on the noise representation is shown. Then identification is discussed. The underlying goal of the chapter is to produce accurate dynamic models of nonuniformly sampled signals, e.g., the network queue length, that can be used to base control decisions on. The different routes are shown schematically in Figure 4.2.

An addition of simple dynamics between the input and the output can then be made, e.g., a constant with a time delay $G(p) = g_0 e^{-pT_{RT}}$ can be appropriate in the network case. The extended model, $y(t) = G(p)u(t) + H(p)e(t)$ can be used for control design. Two control goals can be seen: the servo goal, $y = y_{ref}$ and the noise supression goal, damp He. The control signal u becomes a function of measuremenst and refernece signal, $u = f(y, y_{ref})$. Simple strategies such as PD can be used to desifn the controller f, but also more advanced techniques, e.g., pole placement, LQG, descriptive functions,

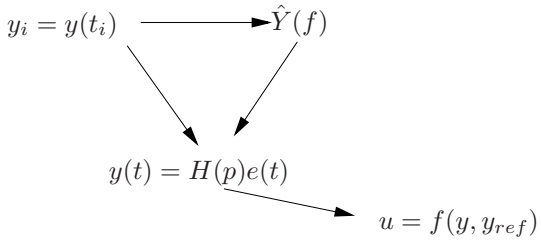


Figure 4.2 *The different routes of the chapter, towards the goal of model based control. The nonuniform samples, y_i can be used to adopt a continuous time model, H, via transform approximations, $\hat{Y}$, or via direct methods. The model are used for control $u = f(y, \hat{y})$.*

etc (Glad and Ljung, 2000).

4.2 Fourier Transform Approximation

To be able to analyze the frequency content of a measured signal, Fourier transforms are used in signal processing. An accurate Fourier transform can also be used for adapting continuous time models to data (see Ljung, 1999, Ch. 6). For a continuous signal, $y(t)$, the Fourier transform is defined as

$$\mathcal{F}(y) = Y(f) = \int_{-\infty}^{\infty} y(t)e^{-i2\pi ft}dt. \tag{4.1}$$

The signal $y(t)$ should fulfill

$$\int_{\mathbf{R}} |y(t)|^2 dt < \infty,$$

which, in this work, is fulfilled by studying only a finite interval of the signal and assuming it equal to zero outside the interval. If the signal $y(t)$ is sampled at time instants t_i, $i = 1, \ldots, N$, to get y_i, the corresponding discrete time Fourier transform depends on the assumptions about the continuous signal y.

For uniformly sampled signals, $y_k = y(t_k) = y(kT)$, the underlying signal, $y(t)$, is often assumed to be band limited, i.e., not containing infinite frequencies. This assumption implies that the signal can be represented as

$$y(t) = \sum_{k=1}^{N} y(kT)\,\mathrm{sinc}(\frac{t - kT}{T}). \tag{4.2}$$

The sinc is defined as

$$\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}.$$

Figure 4.3 shows the time plot of the sinc. It is 0 at all non-zero integer points and 1 at time 0. The frequency content of the sinc is found from the Fourier

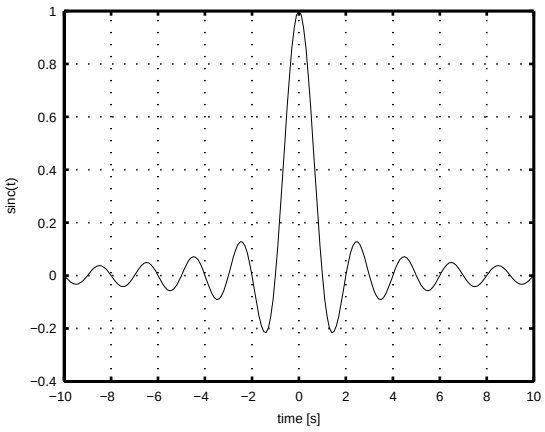


Figure 4.3 *The time plot of a* sinc*-function.*

transform

$$\mathcal{F}(\mathrm{sinc}(t)) = \int_{\mathbf{R}} \mathrm{sinc}(t) e^{-i2\pi ft} dt = \begin{cases} 1 & |f| \leq \frac{1}{2} \\ 0 & |f| > \frac{1}{2}. \end{cases} \tag{4.3}$$

This implies that the the frequency content of the signal, represented as Eq. (4.2), is limited by $f_N = \frac{1}{2T}$.

For a uniformly sampled signal, $y_k = y(kT)$, inserting Eq. (4.2) into Eq. (4.1) gives the resulting transform

$$\hat{Y}_{us}(f) = \int_{-\infty}^{\infty} y(t) e^{-i2\pi ft} dt = \int_{\mathbf{R}} \sum_{k=1}^{N} y_k \,\mathrm{sinc}(\frac{t - kT}{T}) e^{-i2\pi ft} dt \tag{4.4}$$

and the properties of the sinc-function, Eq. (4.3), give the final result

$$\hat{Y}_{us}(f) = \begin{cases} T \sum_{k=1}^{N} y_k e^{-i2\pi fkT}, & |f| < \frac{1}{2T} \\ 0, & \text{otherwise} \end{cases}. \tag{4.5}$$

When the underlying signal y is band limited, the representation in Eq. (4.2) is the true one and, consequently, $\hat{Y}_{us}$ is the correct transform. For other signals, it is an approximation. The commonly used Discrete Fourier Transform,

DFT, is the periodic continuation of the continuous approximation, $\hat{Y}_{us}(f)$, sampled also in the frequency domain, at $f = \frac{n}{NT}$,

$$\hat{Y}_{DFT}[n] = \sum_{k=1}^{N} y_k e^{-i2\pi nk/N}. \tag{4.6}$$

The *sampling theorem*, see e.g. Papoulis (1977), states that $\hat{Y}_{us}(f)$ is exactly equal to $Y(f)$ for all signals with $y(kT) = y_k$ that are band limited to $f < \frac{1}{2T}$, i.e., can be formulated as in Eq. (4.2).

When nonuniform sampling is used some considerations have to be made to generalize definitions and transforms for uniform sampling to be valid for nonuniform sampling as well. Nonuniform frequency analysis is not common in earlier work. Van Steenis and Tulen (1991) study the heart rate spectrum using the Riemann approximation of Eq. (4.1), based on event samples, as the transform

$$\hat{Y}_{ra}(f) = \sum_{k=1}^{N} y_k T_k e^{-i2\pi f t_k}. \tag{4.7}$$

Persson (2002) adopts the same transform and uses it for indirect tire pressure monitoring. No extensive evaluation of the correctness of the transform choice is done. For uniform sampling $\hat{Y}_{ra} = \hat{Y}_{us}$, which is the motivation for this choice.

The following sections will present different ways of approximating the Fourier transform, Eq. (4.1), for nonuniformly sampled signals. Three basic ideas will be discussed:

- Approximating the integrand in Eq. (4.1),
- resample the signal uniformly and use Eq. (4.5), and
- reconstructing the continuous signal to calculate Eq. (4.1).

In Section 4.2.1, the integrand is approximated using polynomials. Resampling with local polynomial modeling is discussed in Section 4.2.2. Reconstruction using basis functions, in particular the sinc, is studied in Section 4.2.3 and, in Section 4.2.4, polynomial splines is used for the reconstruction.

4.2.1 Extensions of the Riemann Approximation

Previous calculation of the Fourier transform for a nonuniformly sampled signal, Eq. (4.7), used a Riemann approximation of the integral, Eq. (4.1). This means that the integrand,

$$I(t) = y(t)e^{-i2\pi ft},$$

is assumed constant between the measurement points, t_k. Extending this with higher order splines is also possible.

Algorithm 4.1 (Spline interpolation)

Spline interpolation is done by connecting the sample points with polynomials, $p_i^n(t)$, of order n. For uniform sampling, splines are thoroughly discussed in Unser (1999). The theory is not straightforward to translate when nonuniform sampling is used. The building blocks are presented here from a different viewpoint. A function estimate $\hat{f}(t)$ is produced using measurements of $f(t)$ at the measurement points t_k. The continuous function is defined as

$$\hat{f}(t) = p_k^n(t), \quad t_{k-1} < t \leq t_k.$$

Introducing

$$p^n(t,\theta) = a_n t^n + \cdots + a_0 = \begin{pmatrix} t^n & \dots & 1 \end{pmatrix} \theta,$$

the spline polynomials are conveniently defined as $p_k^n(t) = p^n(t - t_k, \theta_k)$. The constants $\theta_k = \begin{pmatrix} a_{n,k} & \dots & a_{0,k} \end{pmatrix}^T$ are defined by continuity demands at the sample points:

$$\begin{aligned} p_i^n(t_k) &= f(t_k) \\ \frac{d^l}{dt^l} p_k^n(t_{k-1}) &= \frac{d^l}{dt^l} p_{k-1}^n(t_{k-1}), \quad l = 0, \dots, n-1 \end{aligned} \tag{4.8}$$

which gives $n+1$ equations for each spline. Defining, e.g., $p_0^n(t) = 0$ gives a unique solution for all θ_k. The choice of $p_0^n(t)$ has to be made to match the measurements reasonably.

Let $\hat{I}^n(t)$ be the continuous estimate of $I(t)$, based on measurements $I(t_k)$ and an nth order spline. From Algorithm 4.1, the first two splines become

$$\begin{aligned} \hat{I}^0(t) &= I(t_k), \quad t_{k-1} < t \leq t_k, \\ \hat{I}^1(t) &= \frac{I(t_k) - I(t_{k-1})}{T_k}(t - t_k) + I(t_k), \quad t_{k-1} < t \leq t_k. \end{aligned} \tag{4.9}$$

For higher orders Eq. (4.8) needs to be solved, with $f(t_k)$ being replaced with $I(t_k)$ to find $\hat{I}^n(t)$.

The Fourier transform estimate becomes

$$\hat{Y}_{ra}^n(f) = \sum_{k=1}^{N} \int_{t_{k-1}}^{t_k} \hat{I}^n(t) dt. \tag{4.10}$$

For the first orders of n, the explicit expressions become

$$\begin{aligned}\hat{Y}^0_{ra}(f) &= \sum_{k=1}^{N} I(t_k)T_k = \sum_{k=1}^{N} y(t_k)T_k e^{-i2\pi f t_k} \\ \hat{Y}^1_{ra}(f) &= \sum_{k=1}^{N} \frac{T_k}{2}(I(t_k) + I(t_{k-1})) \\ &= \frac{1}{2}\sum_{k=1}^{N}(T_k + T_{k+1})I(t_k).\end{aligned}$$

The last equality demands that $T_1 I(t_0) = 0$ and $T_{N+1} I(t_N) = 0$, but the relation shows that the increased polynomial order merely changed the scaling of the integrand $I(t_k)$. The mean value of two subsequent inter sample times are used instead of only T_k.

These two approximations are clearly linear in the measurements which will be used later.

4.2.2 Resampling through Local Polynomial Models

Because of the connection described by the sampling theorem between the continuous and discrete Fourier transform, the continuous transform can be approximated by finding estimates of the function values at uniform sample points, $\hat{y}(kT)$, and then calculate Y_{us} for these new samples. An example of uniform resampling is shown in Figure 4.4.

In this work, the estimation of the function samples, $\hat{y}(kT)$, are done using local polynomial modeling. There are, of course, a variety of techniques that can be used for this purpose, such as nearest neighbor and linear interpolation between surrounding measurements. This section focuses on a more advanced method were several measurements are used for each new sample. This methods also assures that the approximation of the transform is linear in the measurements y_i.

The underlying assumption is that y can be locally modeled as a polynomial,

$$p(t) = \theta(1)t^n + \cdots + \theta(n) = \begin{pmatrix} t^n & \cdots & 1 \end{pmatrix}\theta = \varphi^T(t)\theta.$$

For each new sample point kT, a weighted mean square error is minimized to find the best value for θ at this point.

Example 4.1 (A sinusoid) *Figure 4.5 shows how* $y(t) = \sin(t)$ *is well described by* $p(t) = t - \frac{t^3}{6}$ *around* $t = 0$*, i.e.,* $n = 3$ *and* $\theta = \begin{pmatrix} -1/6 & 0 & 1 & 0 \end{pmatrix}^T$*. In this case the polynomial comes from the Taylor expansion of a sinusoid around* $t = 0$*, but in general the nonuniform samples of* y *should be used to find the best polynomial. The example shows that it is critical to choose the span in time where the model is valid. The third order polynomial is not at all valid*

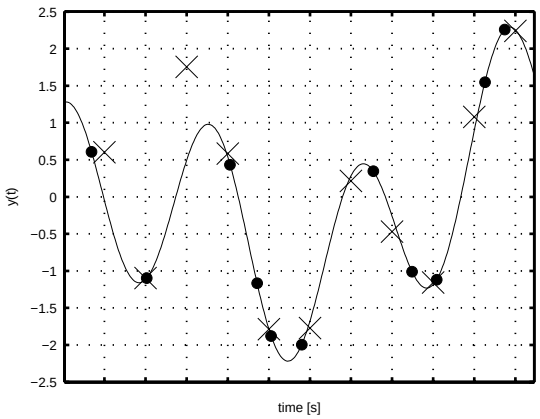


Figure 4.4 *A nonuniformly sampled signal is uniformly resampled ('x'), using local polynomial models of order* 3*, cf. Figure* 4.1*. The worst fit is achieved for the sample point,* kT*, during the largest* T_i*.*

for $|t| > 2$ *in this case. In this case it seems that samples between* -1 *and* 1 *could be used to estimate a third order local polynomial.* □

The value of $\hat{y}(kT)$ is found by minimizing the difference between the measurements, y_i, and the local polynomial $p(t_i - kT)$, giving the best choice of parameters, θ_k, around kT. Weighting is done based on the distance between the measurement point, t_i, and the time point of interest, kT, using w_i. The parameters are found from

$$\begin{aligned}\hat{\theta}_k &= \arg\min_{\theta} \sum_{i=1}^{N} w_i (y_i - p(t_i - kT))^2 \\ &= \arg\min_{\theta} (Y - \Phi(kT)\theta)^T W (Y - \Phi(kT)\theta) \\ &= (\Phi^T W \Phi)^{-1} \Phi^T W Y\end{aligned} \tag{4.11}$$

and $\hat{y}(kT) = p(0) = \hat{\theta}_k(n)$. Since $\hat{\theta}_k$ is linear in the measurements, so is $\hat{y}(kT)$.

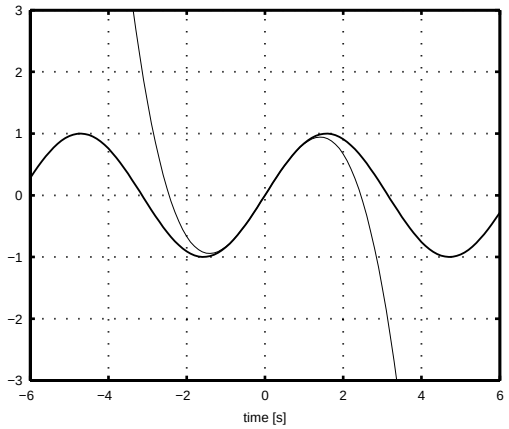


Figure 4.5 $y(t) = \sin(t)$ *(thick line) fits well to* $p(t) = t - t^3/6$ *(thin line) for* $|t| < 1$.

The matrices in Eq. (4.11) are formed as

$$Y = \begin{pmatrix} y_1 & y_2 & \dots & y_N \end{pmatrix}^T$$
$$\Phi(kT) = \begin{pmatrix} \varphi(t_1 - kT) & \varphi(t_2 - kT) & \dots & \varphi(t_N - kT) \end{pmatrix}^T$$
$$W = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & w_N \end{pmatrix}$$

The weights should be chosen to reflect the local modeling, measurements further away from kT should be less important.

A thorough analysis of local polynomial modeling and different choices of weights can be found in Roll (2003). One example on how to chose weights is based on the Epanechnikov function

$$K(u) = \max(1 - u^2, 0). \tag{4.12}$$

It can be scaled using a bandwidth parameter $K_h(u) = \frac{1}{h}K(u/h)$ and the weights can be chosen as

$$w_i = \frac{K_h(t_i - kT)}{\sum_j K_h(t_j - kT)}. \tag{4.13}$$

Figure 4.6 shows the scaled Epanechnikov function, $K_h(u)$. The bandwidth

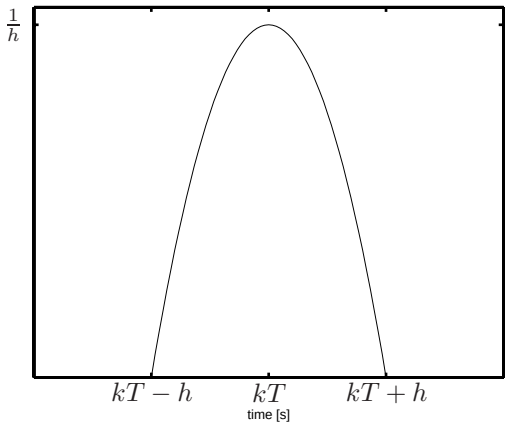


Figure 4.6 *The scaled Epanechnikov function,* $K_h(t) = \frac{1}{h}K((t-kT)/h)$.

parameter should be chosen to include enough measurement points in the interval $[kT-h, kT+h]$, and it can be varied with k.

Returning to the introductory example, Figure 4.4, the used resampling technique can be described. 3^{rd} order polynomials and a changing bandwidth, including approximately 5 nonuniform samples, were used for the calculation of each uniform sample. For this case, the samples match the original function very good in all cases except one.

If the local polynomials are accurate, the resampling procedure is, in principal, the same as direct uniform sampling of the signal. An important issue is how to choose the new sampling time, T. Both $T = \min_k T_k$ and $T = E[T_k]$ are possible and interesting choices. This work does not investigate the issue further.

In this work, the resampling of y_i with the Epanechnikov function with bandwidth h and an nth order polynomial yields new samples $\hat{y}^{h,n}(kT)$ and the transform

$$\hat{Y}_p^{h,n}(f) = T\sum_{k=1}^{N} \hat{y}^{h,n}(kT)e^{-i2\pi fkT}, \quad |f| < \frac{1}{2T}. \tag{4.14}$$

The bandwidth can be varied to include more measurements or to make the model more local. Choosing h can be done either to keep the time span fixed or to include the same number of measurement points for each calculation. Typically at least one measurement on each side of the new sample should be included and at least as many as the dimension of θ. The relation between T and the T_k's must also be considered when h is chosen. If $E[T_k] << T$, a fixed value of h is feasible, but, if $E[T_k]$ is close to or larger than T, the bandwidth should be varied to avoid empty sets of nonuniform samples.

4.2.3 Basis Expansion

Basis expansion can be used to reconstruct the continuous time signal, $\hat{y}(t)$, from measurements. The reconstruction, $\hat{y}$, is a scaled sum of basis functions, H,

$$\hat{y}(t) = \sum_k c_k H(a_k(t - b_k)) \tag{4.15}$$

and the basis function H is shifted and scaled. The parameters are calculated to match the measurements as well as possible. The optimal set of parameters is the solution to the equation system in Eq. (4.16):

$$\sum_{k=1}^{M} c_k H(a_k(t_i - b_k)) = y_i, \quad i = 1, \ldots, N \tag{4.16}$$

For uniformly sampled signals, the signal is often assumed band limited. The sinc function is then used because of its nice properties in the frequency domain, i.e.,

$$H(t) = \mathrm{sinc}(t).$$

The sinc was defined and discussed on p. 54, and, with this choice, $\hat{y}(t)$ in Eq. (4.15) is band limited.

Let us now study the assumption about the signal being a *sum of* sinc-*functions*. For uniform sampling, perfect fitting at the measurement points is obtained with $b_k = kT$, $a_k = 1/T$, $c_k = y_k$ and $M = N$, because of the properties of the sinc. This model for the signal,

$$y(t) = \sum_{k=1}^{N} y_k \,\mathrm{sinc}\left(\frac{t - kT}{T}\right) \tag{4.17}$$

is also what is used for most calculations when considering uniform sampling, yielding the optimal value 0 in Eq. (4.16).

After choosing a_k and b_k and M, the amplitudes, c_k, can be found by solving a linear equation system. The following system of equations is obtained from Eq. (4.16), when optimum 0 is wanted:

$$AX = B \tag{4.18}$$

where

$$A = \begin{pmatrix} \mathrm{sinc}(a_1(t_1 - b_1)) & \cdots & \mathrm{sinc}(a_M(t_1 - b_M)) \\ \vdots & & \vdots \\ \mathrm{sinc}(a_1(t_N - b_1)) & \cdots & \mathrm{sinc}(a_M(t_N - b_M)) \end{pmatrix} \tag{4.19a}$$

$$X = \begin{pmatrix} c_1 & \cdots & c_M \end{pmatrix}^T \tag{4.19b}$$

$$B = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^T \tag{4.19c}$$

Fixing a_k and b_k assures that the c_k's are linear in the measurements. Note that A equals the identity matrix, I_N, for uniformly sampled signals, i.e., $t_i = iT$, $b_k = kT$ and $a_k = 1/T$, $M = N$. For nonuniformly sampled signals a higher M will give give a better fit to the original signal, but it is not possible to use $M > N$ to solve Eq. (4.18). Therefore, there are two possible choices of a_k and b_k related to the choices made for uniform sampling. Place the basis functions at the sample points t_k or at uniformly spaced points kT and scale with the inter sample time; these two placement choices coincide for uniform sampling.

Case 4.1 (Nonequidistant grid of basis functions, $\hat{Y}^{neq}_{sinc}(f)$)
For measurements at times t_k with $T_k = t_k - t_{k-1}$: One sinc *is placed at each measurement point, i.e., $b_k = t_k$ and $a_k = \frac{1}{T_k}$. This gives $M = N$. The transform for this choice is $\hat{Y}^{neq}_{sinc}(f)$.*

Case 4.2 (Equidistant grid of basis functions, $\hat{Y}^{eq}_{sinc}(f)$)
One sinc *is placed at equidistant predetermined points, i.e., $b_k = kT$ and $a_k = \frac{1}{T}$. Since $\mathrm{sinc}(n) = 0$ when $0 \neq n \in \mathbf{Z}$ this implies that $c_k = y(kT)$ for a perfect fit at times kT. T becomes the new design variable and c_k has to be calculated as estimates of the function values $y(kT)$ based on the measurements, y_i. A uniquely determined equations system and a span over the whole time gives $M = N$ and $T = t_N/N$. Larger values of T and a smaller M is also possible. The approximation of the transform is called $\hat{Y}^{eq}_{sinc}(f)$.*

Figure 4.7 show the reconstructed signal when the nonequidistant grid is used and Figure 4.8 shows the correspsonding result when the equidistant grid is used.

Applying the assumption of an underlying sum of sinc's

$$\hat{y}(t) = \sum_k c_k \,\mathrm{sinc}(a_k(t - b_k))$$

gives a straightforward calculation of the Fourier transform.

$$\begin{aligned} \hat{Y}_{sinc}(f) &= \sum_k c_k \int_{-\infty}^{\infty} \mathrm{sinc}(a_k(t-b_k)) e^{-i2\pi f t} dt \\ &= \sum_k c_k e^{-i2\pi f b_k} H_f(\frac{1}{a_k}) \\ &= \sum_{k: f < \frac{a_k}{2}} \frac{1}{a_k} c_k e^{-i2\pi f b_k} \end{aligned} \tag{4.20}$$

with

$$H_f(T) = \begin{cases} T, & |f| < \frac{1}{2T} \\ 0, & \text{otherwise} \end{cases} . \tag{4.21}$$

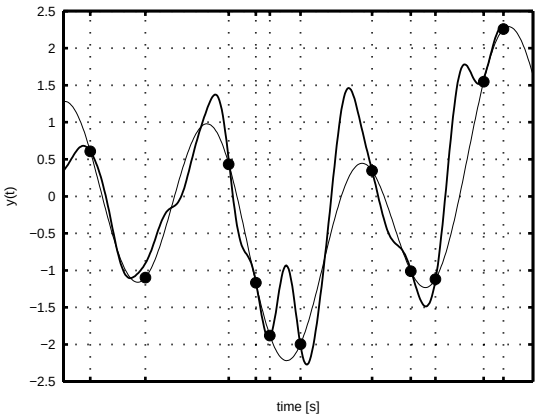


Figure 4.7 *The reconstruction $\hat{y}$ (thick) when a nonequidistant grid of basis functions is used to reconstruct the signal.*

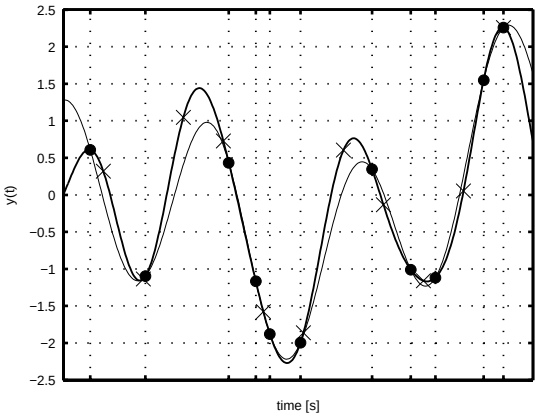


Figure 4.8 *The reconstructed signal, $\hat{y}$, (thick) when an equidistant grid ('x') is used for the basis functions.*

The first equality in Eq. (4.20) is given by Eq. (4.3) with a change of parameters. Note that with a non equidistant grid of basis functions, Case 4.1, and when $f < \frac{1}{2T_k}\,\forall k$, $\hat{Y}_{sinc}(f)$ is simply the Riemann approximation of the integral

$$\int g(t)e^{-i2\pi ft}dt$$

for any function g with $g(t_k) = c_k$. For uniformly sampled signals, $g(t_k) = g(kT) = y_k$ yields the discrete time Fourier transform in Eq. (4.5). The connection between the Riemann approximation and the transform using sinc basis functions is clear for uniformly sampled signals.

Continuing with the matrix notation introduced in Eq. (4.18), the transform, Eq. (4.20), becomes

$$\hat{Y}_{sinc}(f) = \begin{pmatrix} \cdots & e^{-i2\pi f b_k} H_f(\frac{1}{a_k}) & \cdots \end{pmatrix} X. \tag{4.22}$$

The difference between this approximation, hY^{neq}_{sinc}, and the transform when assuming $X = B$, which corresponds to the Riemann approximation, $\hat{Y}^0_{ra}$, is related to the distance between A and the identity matrix, since $AX = B$.

It is interesting to note a difference between the result using uniform or nonuniform sampling. For uniform sampling, $\hat{Y}^0_{ra}(f)$ corresponds exactly to $\hat{Y}_{sinc}(f)$ for $f < \frac{1}{2T}$, which is not true for nonuniform sampling.

4.2.4 Spline Interpolation

Spline interpolation can be done by connecting the sample points with polynomials, $p^n_k(t)$, of order n. Spline interpolation was introduced in Algorithm 4.1.

The continuous function is defined as

$$\hat{y}^n(t) = p^n_k(t), \quad t_{k-1} < t < t_k, \tag{4.23}$$

where the connecting polynomials p_k are found from Algorithm 4.1 using $f(t_k) = y_k$. Figure 4.9 shows the reconstruction using $n = 0$ and Figure 4.10 shows the reconstruction for $n = 1$.

The approximation of the Fourier transform becomes

$$\hat{Y}^n_{sp}(f) = \sum_k \int_{t_{k-1}}^{t_k} p^n_k(t)e^{-i2\pi ft}dt = \sum_{k=1}^{N}\sum_{m=0}^{n} a_{m,k}\int_{t_{k-1}}^{t_k}(t-t_k)^m e^{-i2\pi ft}dt \tag{4.24}$$

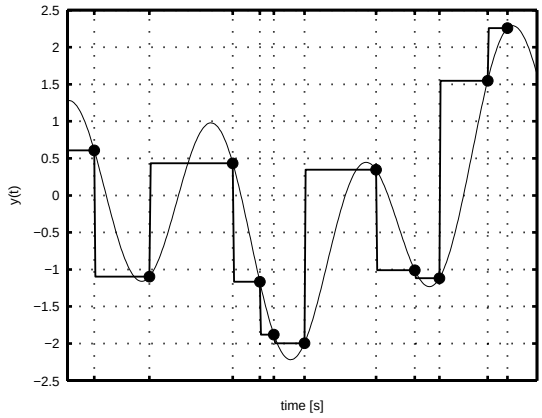


Figure 4.9 *The reconstruction $\hat{y}$ when splines of order $n = 0$ is used.*

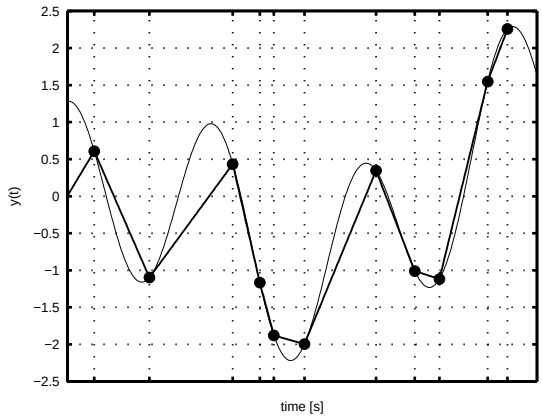


Figure 4.10 *The reconstruction $\hat{y}$ when splines of order $n = 1$ is used.*

and expanding the integrals gives

$$
\begin{aligned}
I_k^m &= \int_{t_{k-1}}^{t_k} (t - t_k)^m e^{-i2\pi ft} dt \\
&= \frac{i}{2\pi f} \left(-(-T_k)^m e^{-i2\pi f t_{k-1}} \right) + \frac{m}{i2\pi f} I_k^{m-1} \\
&= J_k^m + \frac{m}{i2\pi f} I_k^{m-1} \\
&= \sum_{l=0}^{m} \frac{m!}{(m-l)!} \frac{1}{(i2\pi f)^l} J_k^{m-l}, \\
J_k^m &= \frac{i}{2\pi f} \left(-(-T_k)^m e^{-i2\pi f t_{k-1}} \right), \quad m = 1, \ldots, n, \\
J_k^0 &= \frac{i}{2\pi f} \left(e^{-i2\pi f t_k} - e^{-i2\pi f t_{k-1}} \right).
\end{aligned}
$$

Note that $I_k^0 = J_k^0$. This gives the Fourier transform estimate when the signal is piece-wise polynomial of order n.

When $n = 0$, $\hat{y}^n(t)$ is piece-wise constant. The explicit transform in this case becomes

$$\hat{Y}_{sp}^0(f) = \sum_k y_k \int_{t_{k-1}}^{t_k} e^{-i2\pi ft} dt = \frac{i}{2\pi f} \sum_{k=1}^{N} y_k e^{-i2\pi f t_k} \left(1 - e^{i2\pi f T_k}\right). \tag{4.25}$$

Comparing with using Riemann approximation of the integral, $\hat{Y}_{ra}^0$, Eq. (4.10), shows that the difference lies in scaling every term in the sum with

$$\frac{i}{2\pi f T_k}(1 - e^{-i2\pi f T_k}) = \frac{1}{i2\pi f T_k}(e^{i\pi f T_k} - e^{i\pi f T_k})e^{i\pi f T_k} = e^{i\pi f T_k} \operatorname{sinc}(f T_k).$$

Using a Taylor expansion for $e^{i2\pi f T_k}$ shows that Y_{sp}^0 scales with

$$\frac{i}{2\pi f}(1 - e^{-i2\pi f T_k}) = T_k + O(T_k^2),$$

while $\hat{Y}_{ra}^0$ scales with T_k. The connection, between the Riemann approximation and reconstruction using piecewise constant splines, is multiplication of the transform with a sinc. This corresponds to convolution of the time signal with a square window. Straightforward calculations for uniform signals are probably not as straightforward for nonuniform signals. The other view shows a connection with the Taylor expansion, where the Riemann approximation uses an approximation of the scaling term used for piecewise constant signal, i.e.,

$$\hat{Y}_{sp}^0 \approx \hat{Y}_{ra}^0, \quad T_k << 1 \, \forall k.$$

These connections are worth investigating further in future work.

For $n = 1$, $\hat{y}^1(t)$ is piece-wise linear between the measurement points. The spline polynomials become

$$p_k^1(t) = \frac{y_k - y_{k-1}}{T_k}(t - t_k) + y_k = \alpha_k(t - t_k) + y_k, \quad k = 1, \ldots, N.$$

which gives the transform approximation

$$\begin{aligned}
\hat{Y}_{sp}^1(f) &= \sum_{k=1}^{N} \int_{t_{k-1}}^{t_k} \left[\alpha_k(t - t_k) + y_k\right] e^{-i2\pi ft} dt \\
&= \sum_{k=1}^{N} y_k I_k^0 + \sum_k \alpha_k I_k^1 \\
&= \sum_{k=1}^{N} y_k I_k^0 + \sum_k \alpha_k \left(J_k^1 + \frac{1}{i2\pi f} I_k^0\right) \\
&= \hat{Y}_{sp}^0(f) + \frac{1}{i2\pi f} \sum_{k=1}^{N} \alpha_k I_k^0 + \frac{i}{2\pi f} \sum_{k=1}^{N} \alpha_k T_k e^{-i2\pi f t_k} e^{i2\pi f T_k}
\end{aligned} \tag{4.26}$$

The transform of y_k for $n = 1$ contains the transform of y_k for $n = 0$, the transform of α_k for $n = 0$ and a third term based on α_k. The structure of the third term is very similar to the structure of a Riemann approximation. The smoothing when introducing a higher order spline representation is shown in the last two terms.

For higher order splines the expressions become messier.

4.2.5 Nyquist Criteria

For time sampled signals the Nyquist criteria is used to determined the frequency content that is possible to represent. If the sample time is T, only frequencies below $f_N = \frac{1}{2T}$ can be identified and if the signal contains higher frequencies these will distort the true spectrum, i.e, frequency aliasing will occur.

For nonuniformly sampled signals there is no unique Nyquist frequency. Assuming that T_k comes from a known distribution with known maximum and minimum values, τ_h and τ_l respectively, a few notes can be made. The minimum Nyquist frequency is

$$f_N^m = \frac{1}{2\tau_h}$$

and if the signal has all its frequency content below f_N^m it should be possible to reconstruct it properly, as is the case in Section 4.2.3 when the sinc is used. The frequencies that lie between the maximum and minimum is defined as the Nyquist band.

Definition 4.2 (Nyquist band)
For a signal measured with inter sample times, T_i, between τ_l and τ_h the **Nyquist band***, F_N, is defined as the set*

$$F_N \triangleq \left\{ f : f_N^m = \frac{1}{2\tau_h} \leq f \leq \frac{1}{2\tau_l} = f_N^M \right\}.$$

When splines are used to reconstruct the signal, there is no assumption about frequency limitations. The similarities between the Riemann approximation and the 0-order spline reconstruction indicated that the resulting transform, when splines are used, is an extension of the Riemann approximation with different higher order scaling of frequencies.

The assumption about sinc-functions says that the signal has only frequencies up to $\max(a_k/2)$. With a non equidistant grid of basis functions, Case 4.1, this gives the same Nyquist band as from the sampling itself. For an equidistant grid of basis functions, Case 4.2, the Nyquist frequency is chosen with T, $f_N = 1/2T$.

Using resampling with local polynomial models, it might be possible to investigate frequencies above the Nyquist band by up-sampling the signal.

The correctness of the transform will in this case depend on the calculation of $\hat{y}_p^{h,n}(kT)$ and, intuitively, changing bandwidth h and $n = 3$, at least, is necessary. It is easy to come up with cases where it will not work, and the demand on the sampling is probably crucial and worth to examine more closely.

4.2.6 Zero-Mean Signals

In many cases a signal with zero mean is wanted for processing. To be able to remove the mean value, implications of the different reconstruction techniques are here evaluated to produce the corresponding mean value.

If the signal is measured from time 0 to $t_N = \sum_{k=1}^{N} T_k$, the mean is

$$\mu = \frac{1}{t_N} \int_0^{t_N} y(t)dt. \tag{4.27}$$

For polynomial splines between the measurements this gives

$$\mu_{sp}^0 = \frac{\sum_k y_k T_k}{\sum_k T_k}, \tag{4.28}$$

$$\mu_{sp}^1 = \frac{\sum_k T_k (y_k + y_{k-1})/2}{\sum_k T_k}. \tag{4.29}$$

When $\hat{Y}_{us}$ or a sum of sinc's are used, the mean is calculated based on $\int \mathrm{sinc}(x)dx$,

$$\mu_{sinc}^{neq} = \frac{1}{t_N} \sum_{k=1}^{N} c_k \sum_{n=0}^{\infty} (-\pi^2)^n \frac{\left((t_N - t_k)^{2n+1} + t_k^{2n+1}\right)}{T_k^{2n}(2n+1)(2n+1)!}, \tag{4.30}$$

$$\mu_{sinc}^{eq} = \frac{1}{N} \sum_{k=1}^{N} c_k \sum_{n=0}^{\infty} (-\pi^2)^n \frac{\left((N-k)^{2n+1} + k^{2n+1}\right)}{(2n+1)(2n+1)!}, \tag{4.31}$$

$$\mu_p^{h,n} = \frac{1}{M} \sum_{k=1}^{M} \hat{y}^{h,n}(kT) \sum_{n=0}^{\infty} (-\pi^2)^n \frac{\left((M-k)^{2n+1} + k^{2n+1}\right)}{(2n+1)(2n+1)!}, \tag{4.32}$$

where M is the new number of uniform samples after the resampling has been done.

The calculation of $\int \mathrm{sinc}(x)dx$ was done using Taylor expansion of a sinusoid and is shown in Appendix 4.A. The sum can be hard to calculate numerically, why a table with numerical values is given as well.

4.2.7 Collection of Transform Approximation Expressions

The different expressions for the Fourier transform approximation are recapitulated here. The measurements, y_i, are used to approximate the integrand $I(t)$, for resampling, or to approximate original continuous signal, $y(t)$.

These calculations are then used together with the definition of the continuous Fourier transform, Eq. (4.1), to produce an estimate, $\hat{Y}(f)$, of the transform for the measured signal.

Extension of Riemann approximations The integrand

$$I(t) = y(t)e^{-i2\pi ft}$$

is approximated with a polynomial spline of order n, $\hat{I}^n(t)$, between the measurement points. For $n = 0$, the transform is the ordinary Riemann approximation,

$$\hat{Y}^0_{ra}(f) = \sum_{k=1}^{N} y(t_k)T_k e^{-i2\pi f t_k}.$$

When $n = 1$, giving a piecewise linear approximation of the integrand, the transform becomes

$$\hat{Y}^1_{ra}(f) = \sum_{k=1}^{N} \frac{T_k}{2}(I(t_k) + I(t_{k-1})).$$

Resampling through local polynomial models New samples of y are produced at predetermined points using local polynomial modeling. A polynomial of order n is adapted to the measurements with weights that are nonzero a bandwidth, h, away from the point. The producing of new samples are described from Eq. (4.11) and forward. The new samples, $\hat{y}^{h,n}(kT)$ are used to produce the transform approximation

$$\hat{Y}^{h,n}_p(f) = T\sum_{k=1}^{M} \hat{y}^{h,n}(kT)e^{-i2\pi fkT}, \quad f < \frac{1}{2T}$$

Basis expansion The estimate of y is

$$\hat{y}(t) = \sum_k c_k \operatorname{sinc}(a_k(t - b_k))$$

where the choice of constants a, b and c are discussed in the cases on page 63. When a non equidistant grid for the basis functions, Case 4.1, is used, the transform approximation becomes

$$\hat{Y}^{neq}_{sinc}(f) = \sum_{k:f<\frac{1}{sT_k}} T_k c_k e^{-i2\pi f t_k}$$

and with an equidistant grid, Case 4.2, the result is

$$\hat{Y}^{eq}_{sinc}(f) = T\sum_{k=1}^{M} c_k e^{-i2\pi fkT}, \quad f < \frac{1}{2T}.$$

Spline interpolation The estimate of y is produced using polynomial splines of order n between the measurements points, Eq. (4.24). For $n = 0$, the transform estimate is

$$\hat{Y}^0_{sp}(f) = \frac{i}{2\pi f}\sum_{k=1}^{N} y_k e^{-i2\pi f t_k}\left(1 - e^{i2\pi f T_k}\right)$$

and for $n = 1$ the approximation is

$$\hat{Y}^1_{sp}(f) = \hat{Y}^0_{sp}(f) + \frac{1}{(2\pi f)^2}\sum_{k=1}^{N}\alpha_k e^{-i2\pi f t_k}\left(1 - e^{i2\pi f T_k}\right) + \frac{i}{2\pi f}\sum_{k=1}^{N}\alpha_k T_k e^{-i2\pi f t_k} e^{i2\pi f T_k}$$

with $\alpha_k = \frac{y_k - y_{k-1}}{T_k}$.

4.3 Additive Random Sampling

For a network queue the signal is piecewise constant and a few more thorough calculations will be done for the case when a 0-order spline is used to reconstruct the signal. In Chapter 3 the dynamics of the queue were successfully modeled using an AR-model. The AR-model captures a number of frequencies and the result is that the queue length is seen as a sum of sinusoids. Therefore, this work studies the effects of the transform of a single sinusoid. The transform approximation when a spline of order 0 is used to reconstruct the signal, $\hat{Y}^0_{sp}$ Eq. (4.25), is linear in the measurements, and further frequencies in the signal will only add transforms.

Let $y(t) = \sin(2\pi f_0 t)$ and assume that the signal is sampled at times t_k where $T_k = t_k - t_{k-1}$ are independent identically distributed, i.i.d., stochastic variables with the probability density function $f_T(\tau)$. Then $\hat{Y}^0_{sp}(f)$ is a stochastic variable. Since $t_k = \sum_{j=1}^{k} T_j$,

$$\hat{Y}^0_{sp}(f) = g(T_1, \ldots, T_N)$$

with $g(\cdot)$ given from Eq. (4.25). The expected value is

$$E[\hat{Y}^0_{sp}(f)] = \int_{\mathbf{R}^N}\cdots\int g(\tau_1, ..., \tau_N)\prod_{k=1}^{N} f_T(\tau_k)d\tau_1 \ldots d\tau_N. \tag{4.33}$$

To show how single frequencies show up in the transform, we will calculate the expected value of $\hat{Y}^0_{sp}(f)$ when

$$y(t) = \sin(2\pi f_0 t) = \frac{1}{2i}(e^{i2\pi f_0 t} - e^{-i2\pi f_0 t})$$

is sampled nonuniformly and the inter event times, T_k, have the probability density function $f_T(\tau)$. The samples, $y_k = y(t_k) = y(\sum_{n=1}^{k} T_n)$[1], become

$$\begin{aligned} y_k &= \frac{1}{2i}\left(e^{i2\pi f_0 \sum_{n=1}^{k} T_n} - e^{-i2\pi f_0 \sum_{n=1}^{k} T_n}\right) \\ &= \frac{1}{2i}\left(\prod_{n=1}^{k} \eta(f_0, T_n) - \prod_{n=1}^{k} \eta(-f_0, T_n)\right). \qquad (4.34) \\ \eta(f,t) &= e^{i2\pi f t} \qquad (4.35) \end{aligned}$$

Combining Eq. (4.34) with Eq. (4.25) and Eq. (4.33) gives

$$\begin{aligned} E[\hat{Y}_{sp}^{0}(f)] &= \frac{i}{2\pi f}\int_{\mathbf{R}^N} \sum_{k=1}^{N}\left[y_k\left(1 - e^{i2\pi f \tau_k}\right)\prod_{n=1}^{k} e^{-i2\pi f \tau_n}\right]\prod_{k=1}^{N} f_T(\tau_k)d\tau_k \\ &= \frac{1}{4\pi f}(\chi(f, f_0) - \chi(f, -f_0)) \qquad (4.36) \end{aligned}$$

where $\chi(f, f_0)$ evaluates according to

$$\begin{aligned} \chi(f, f_0) &= \int_{\mathbf{R}^N} \sum_{k=1}^{N}\left[\left(1 - e^{i2\pi f \tau_k}\right)\prod_{n=1}^{k} \eta(f_0, \tau_n) e^{-i2\pi f \tau_n}\right]\prod_{k=1}^{N} f_T(\tau_k)d\tau_k \\ &= \sum_{k=1}^{N}\left[\left(\int_{\mathbf{R}}\left(1 - e^{i2\pi f \tau_k}\right)e^{-i2\pi(f-f_0)\tau_k} f_T(\tau_k)d\tau_k\right)\times \right. \\ &\quad \left.\left(\prod_{n=1}^{k-1}\int_{\mathbf{R}} e^{-i2\pi(f-f_0)\tau_n} f_T(\tau_n)d\tau_n\right)\right] \\ &= \sum_{k=1}^{N}\gamma(f, f_0)^k - \gamma(0, f_0)\sum_{k=1}^{N}\gamma(f, f_0)^{k-1} \\ &= \begin{cases} (\gamma(f, f_0) - \gamma(0, f_0))\frac{1-\gamma(f,f_0)^N}{1-\gamma(f,f_0)} & \gamma(f, f_0) \neq 1 \\ (1 - \gamma(0, f_0))N & \text{otherwise} \end{cases} \qquad (4.37a) \\ \gamma(f, f_0) &= \int_{\mathbf{R}} e^{-i2\pi(f-f_0)\tau} f_T(\tau)d\tau \qquad (4.37b) \end{aligned}$$

Figures 4.11, 4.12 and 4.13 show the expected value of $\hat{Y}_{sp}^{0}(f)$ when the T_i's come from a rectangular distribution between τ_l and τ_h, i.e., Eq. (4.36) together with Eq. (4.37) and

$$f_T(\tau) = \begin{cases} \frac{1}{\tau_h - \tau_l} & \tau_l \leq \tau \leq \tau_h \\ 0 & \text{otherwise} \end{cases}$$

[1] When jitter sampling is used $t_k = kT + \nu_k$. The expression will become different and simpler since each measurement only depend on one random variable.

The input signal y is a single sinusoid with frequency f_0. The width of the inter sample distribution is shown with dashed vertical lines at $1/(2\tau_l)$ and $1/(2\tau_l)$. The frequency, f_0 and the support and position f_T clearly affect the height and the tails of the transform approximation, $\hat{Y}^0_{sp}$.

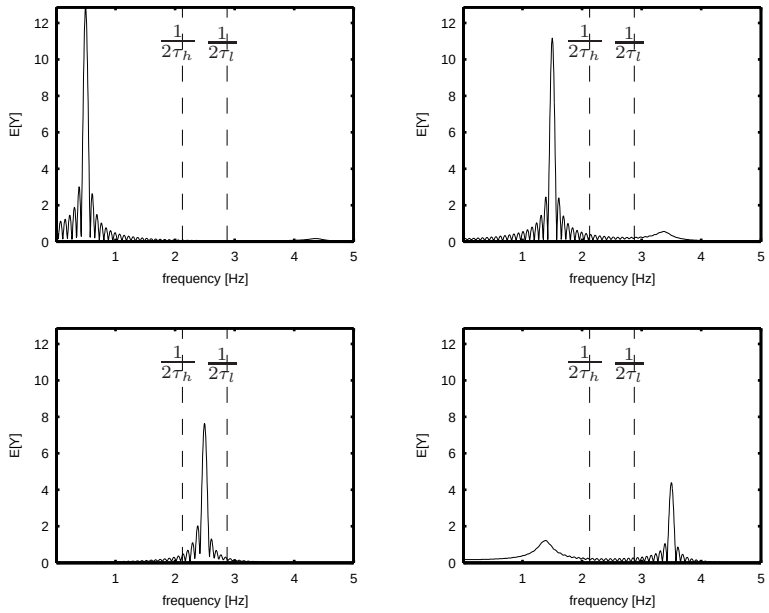


Figure 4.11 *The expected value, $E[\hat{Y}^0_{sp}(f)]$, Eq. (4.25), when $y(t) = \sin(2\pi f_0 t)$ and f_0 is varied $f_0 = \{0.5, 1.5, 2.5, 3.5\}$. The impact on the height of the response for different frequencies, f_0, is evident. The two vertical lines show the limits for the distribution of $\frac{1}{2T}$.*

The conclusions from the studies above are valid even if $f_T(\tau)$ is an exponential function as has been suggested for network queues, see Section 4.1.

$$f_T(\tau) = \begin{cases} g_0 e^{-\lambda(\tau-\tau_l)} & \tau_l \leq \tau \leq \tau_h \\ 0 & \text{otherwise} \end{cases}$$

gives an exponential distribution between τ_l and τ_h and g_0 is chosen to make $\int f_T(\tau)d\tau = 1$. See Appendix 4.B for further discussions about limited distributions. Calculating $\gamma(f, f_0)$ for the two choices of f_T shows that the difference

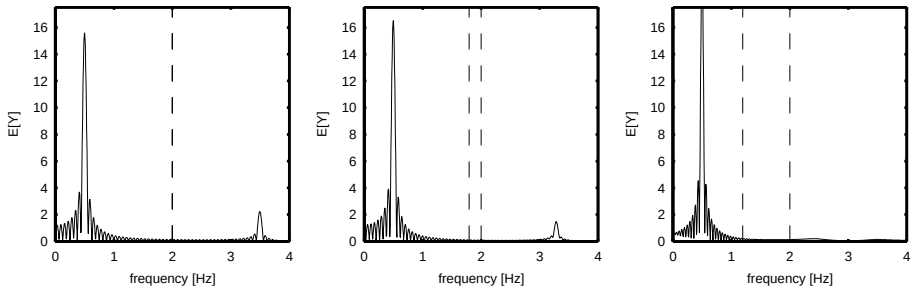


Figure 4.12 *The expected value, $E[\hat{Y}^0_{sp}(f)]$, Eq. (4.25), when $y(t) = \sin(2\pi f_0 t)$ ($f_0 = 0.5$ Hz) and the support for $f_T(\tau)$ (dashed lines show the distribution limits for $\frac{1}{2T}$) is varied. Due to the piecewise constant assumption, new peaks arise, whose width, height and position depend on the T_i-distribution. The left plot corresponds to uniform sampling.*

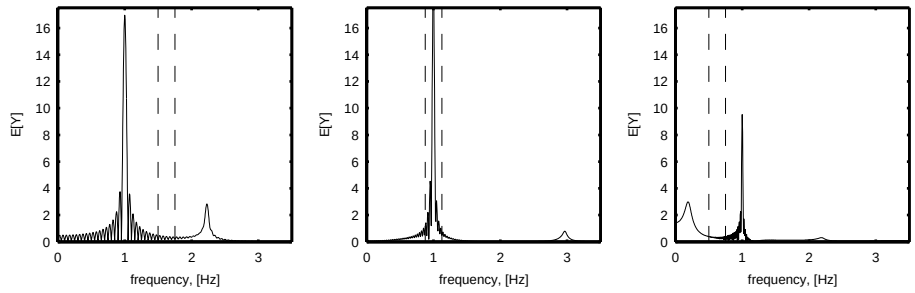


Figure 4.13 *The expected value, $E[\hat{Y}^0_{sp}(f)]$, Eq. (4.25), when $y(t) = \sin(2\pi f_0 t)$ ($f_0 = 1$ Hz) when the limits for $f_T(\tau)$ (dashed lines show the limits for $\frac{1}{2T}$-distribution) is moved. The tails of the transform are also affected by the position of the T_i-distribution.*

is mainly in constants. For $f \neq f_0$ Eq. (4.37b) becomes

$$\begin{aligned}\gamma_{rect}(f, f_0) &= \frac{1}{(\tau_h - \tau_l)(i2\pi(f - f_0))}\left(e^{-i2\pi(f-f_0)\tau_l} - e^{-i2\pi(f-f_0)\tau_h}\right) \\ \gamma_{exp}(f, f_0) &= \frac{g_0}{i2\pi(f - f_0) + \lambda}\left(e^{-i2\pi(f-f_0)\tau_l} - e^{-\lambda(\tau_h-\tau_l)}e^{-i2\pi(f-f_0)\tau_h}\right).\end{aligned} \tag{4.38}$$

The principal form, e.g., the number of side lobes, of $E[\hat{Y}^0_{sp}(f)]$ will only be affected by the support for the distribution f_T, i.e., τ_h and τ_l. The choice of distribution affects the heights of the center and side lobes, via g_0 and λ. Figure 4.14 shows $E[\hat{Y}^0_{sp}(f)]$ when γ_{exp} is used with $\lambda = 2$ and $\tau_l = 1, \tau_h = 5$ to the left and $\tau_l = 0, \tau_h = \infty$ to the right. In Figure 4.15, the same limits for the distribution of T is used, but different distributions.

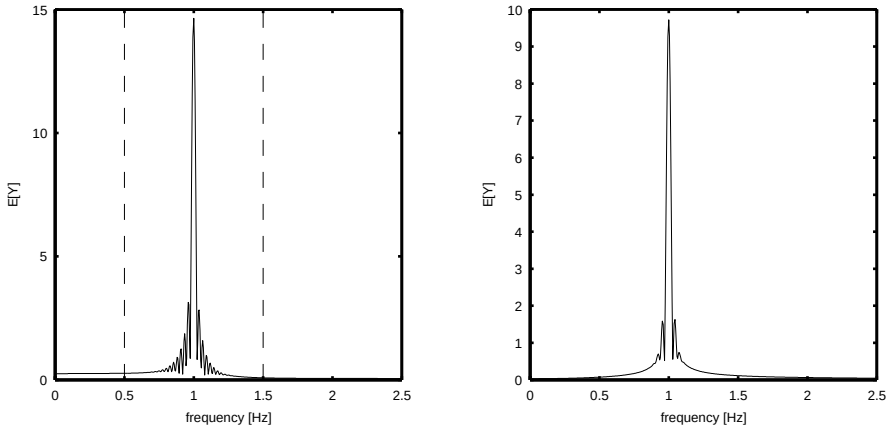


Figure 4.14 *A comparison of $E[\hat{Y}^0_{sp}(f)]$ when an exponential distribution is chosen for f_T, Eq. (4.38) and the limits for $f_T(\tau)$ are varied. To the left $\lambda = 2$ and $\tau_l = 1/3, \tau_h = 1$ (The limits for the distribution of $1/2T$ are shown as vertical lines) and $\tau_l = 0, \tau_h = \infty$ to the right.*

4.4 Evaluation

After presenting a number of methods to produce an estimate of the Fourier transform, a discussion about performance for the different approaches is needed. The different transforms have been implemented using Matlab. First the different methods to approximate the continuous function is discussed and then a large example is used to compare the different transform approximations.

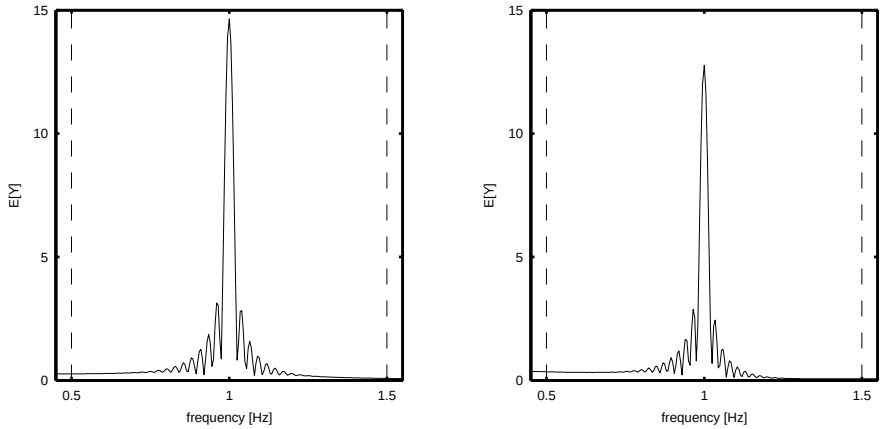


Figure 4.15 *A comparison of $E[\hat{Y}_{sp}^0(f)]$ when different distributions are used. To the left an exponential distribution and to the right a rectangular distribution is used.*

4.4.1 Signal Model Evaluation

Based on what we know about the model of the continuous signal, a few conclusions about the positive and negative aspects of them can be drawn.

- For reconstruction of $y(t)$ with a piecewise constant $\hat{y}$, $\hat{Y}_{sp}^0$, the calculations are quite straightforward, which is not the case for any of the other assumptions. Studying the expected value, Section 4.3, showed that a wider Nyquist band will average out the mirror frequencies.

- Real signals are seldom piecewise constant or linear, but when that is the case these two models of course are highly relevant, e.g., for network queues. In other cases it is a trade off between exactness and amount of calculations. A higher order spline polynomial will require more calculations but probably be closer to the real signal.

- Spline interpolation introduces no assumption about frequency band limitation on the reconstructed signal.

- Using a sinc as basis function is a commonly used signal model for uniformly sampled signals, and it introduces band limitation in the frequency domain also for the nonuniform case. The calculations can be more easily compared to known results for uniform sampling.

- The Riemann approximation and its extensions provide low complexity.

Name	τ_l	τ_h	T_s	N	MC
Value	0.23	0.28	0.25	2^9	20

Table 4.1 *The parameter settings for the evaluation run, notation as in Section 4.2.5.*

- Resampling with local polynomial models requires more design choices than the other setups.

4.4.2 A Large Example

To evaluate the different transform approximations we will study a particular example. The different transform approximation techniques will be used on a signal with a single frequency. Some of the techniques are clearly linear, but no investigation about this fact is done. The example will be used to draw conclusions about the performance of the approximations. The setup of the example was chosen because of the modeling of the queue length dynamics in the previous chapter, with a few basic frequencies.

Example 4.2 (Transform comparison) *Let us study the case when*

$$y(t) = \sin(2\pi f_0 t) \tag{4.39}$$

is measured at points $t_i = \sum_{j=1}^{i} T_j$ *and* T_i *comes from an exponential distribution with mean* T_s *truncated to* $[\tau_l, \tau_h]$. *Limited distributions as they are used here are described in Appendix 4.B. Figure 4.16 provides a normalized histogram of the inter sample intervals,* T_k. *Table 4.1 shows the parameter settings for this run. The Nyquist band becomes*

$$F_N = \{f : 1.79 \leq f \leq 2.17\}$$

and, if nothing else is stated, this is what is used for the different approximations. Each run is performed MC *times to avoid effects from the realization of the* T_i*'s.*

Four different frequencies, f_0, *are used,* 0.1, 0.55, 1.9 *and* 2.6 *Hz.* 12 *different transform approximation for nonuniform sampling are compared to the transform when the signal is uniformly sampled. The different setups are described and named. A summary of the definitions were given in Section 4.2.7.*

Uniform sampling, Eq. (4.5) $\hat{Y}_{us}(f)$ *is calculated for reference, using the fixed sampling interval* $T = \frac{1}{5}\frac{1}{2f_0}$, *which gives* $f_N = 5f_0$.

Riemann approximation, Section 4.2.1 *Only the piecewise constant approximation of the integrand is tested,* $\hat{Y}^0_{ra}(f)$.

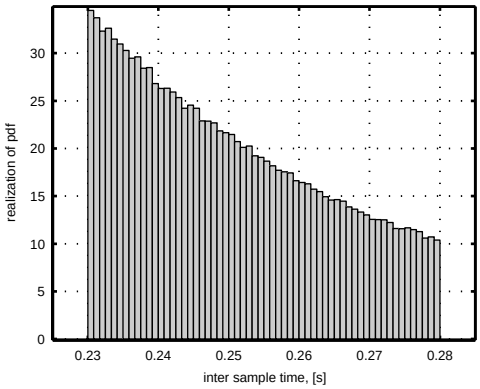


Figure 4.16 *The realization of the probability density function is shown as the histogram of the T_i's for the simulation example. The exact form of the distribution can be found from the exponential function.*

Resampling through local polynomial models, Section 4.2.2 *Polynomials of order $n = 3$, two strategies for choosing the bandwidth, h:*

Fixed bandwidth, h_f**:** *The bandwidth is fixed to $h_f = T/2$,*

$\hat{Y}_{p,1}^{h_f,3}(f)$ $T = 1.1T_s$ *giving* $f_N = 1.82$,

$\hat{Y}_{p,2}^{h_f,3}(f)$ $T = 3T_s$ *and* $f_N = 0.67$.

Varying bandwidth, h_v**:** *The bandwidth is varied to include approximately 10 points, i.e., $kT - h_v < t_i < kT + h_v$, $i \approx k - 5, \ldots, k + 5$:*

$\hat{Y}_{p,1}^{h_v,3}(f)$ $T = T_s/3$ *and* $f_N = 6$,

$\hat{Y}_{p,2}^{h_v,3}(f)$ $T = T_s$ *and* $f_N = 2$,

$\hat{Y}_{p,3}^{h_v,3}(f)$ $T = 3T_s$ *and* $f_N = 0.67$.

Basis expansion, Section 4.2.3 *The* sinc *is used as basis function:*

$\hat{Y}_{sinc}^{neq}(f)$ *Non equidistant grid, Case 4.1.*

$\hat{Y}_{sinc}^{eq}(f)$ *Equidistant grid, Case 4.2 with $T = T_s$ giving $f_N = 2$.*

$\hat{Y}_{sinc,d}^{eq}(f)$ *Downsampling with an equidistant grid, using $T = 3T_s$ and $M = N/3$, giving $f_N = 0.67$.*

Spline interpolation, Section 4.2.4 *Two cases of spline interpolation are evaluated,*

$\hat{Y}_{sp}^{0}(f)$ *piecewise constant, and,*

$\hat{Y}_{sp}^{1}(f)$ *piecewise linear.*

Each transform is calculated over 256 *frequency points spaced with* 0.02 *Hz, i.e., the comparison ranges from* 0 *to* 5.1 *Hz. The relative position of the different* f_0, f_N *and* F_N *is shown in Figure 4.17.*

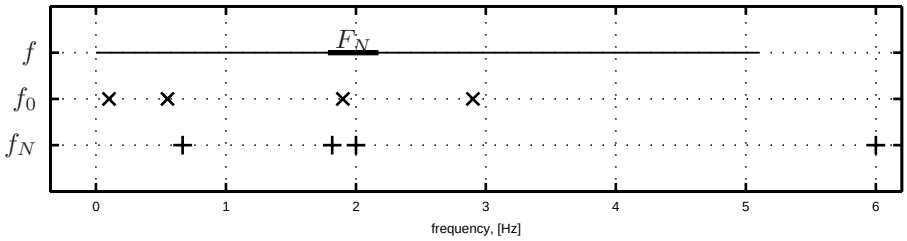


Figure 4.17 *The relative position of the different values of the signal frequency,* f_0 *Eq. (4.39), the nyquist frequencies of the resmapled signals,* f_N*, and the Nyquist band,* F_N*, during the simulation runs.*

The absolute and relative difference between the tested transform approximation, $\hat{Y}_*(f)$*, and the uniform one,* $\hat{Y}_{us}(f)$*, is used as performance measures. Figure 4.18 shows differences between* $\hat{Y}_{us}$ *and six of the tested transform approximations;* $\hat{Y}^0_{ra}$*,* $\hat{Y}^{neq}_{sinc}$ *and four* $\hat{Y}^{h,3}_p$*, when* $f_0 = 0.55$ *Hz. Table 4.2 shows a comparison of all the instances calculated. To measure the correctness of the approximations the absolute error,*

$$dP_a = \| |\hat{Y}_*(f)|^2/t_N - |\hat{Y}(f)|^2/(NT)\|, \tag{4.40}$$

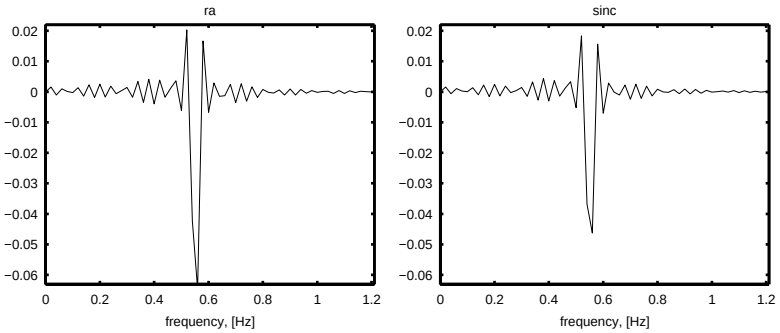
and the relative error

$$dP_r = \left\| \frac{|\hat{Y}_*(f)|^2/t_N - |\hat{Y}(f)|^2/(NT)}{|\hat{Y}(f)|^2/(NT)} \right\|, \tag{4.41}$$

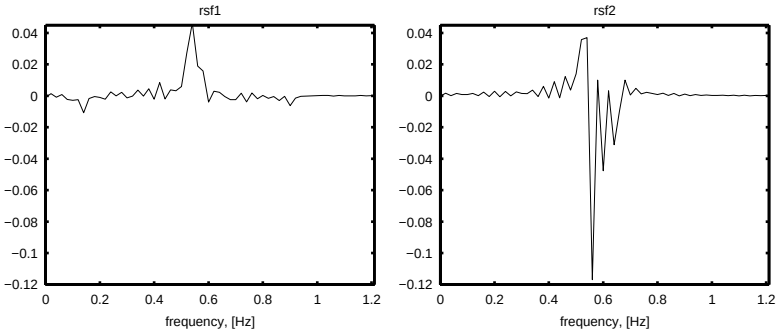
are used, where $\hat{Y}_*$ *is the tested transform approximation. A small value of* dP_a *shows a good fit overall and a small value of* dP_r *shows a good fit for small* $|\hat{Y}|^2/(NT)$*, i.e., on the tails of the transform. Rank numbering is used to show the best and some of the runner ups for each column.* □

The numeric results in Example 4.2 show that nothing is obvious when the Fourier transform of an nonuniformly sampled signal is sought. Some notes can be made:

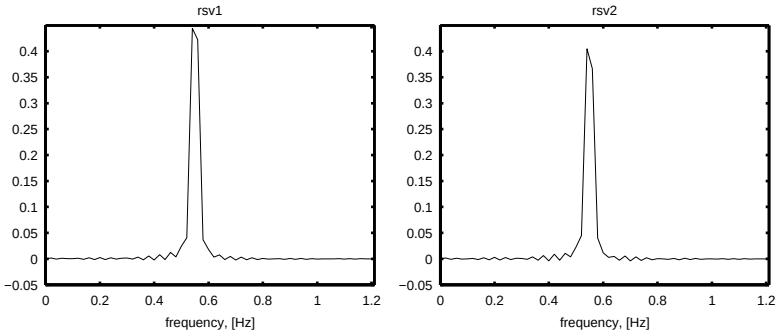
1. It is easier to get good results for frequencies significantly below the Nyquist band. The lower f_0 the better the fit of the tails (cf. columns from left to right).



a. $\hat{Y}_* = \hat{Y}^0_{ra}$ **b.** $\hat{Y}_* = \hat{Y}^{neq}_{sinc}$



c. $\hat{Y}_* = \hat{Y}^{h_f,3}_{p,1}$ **d.** $\hat{Y}_* = \hat{Y}^{h_f,3}_{p,2}$



e. $\hat{Y}_* = \hat{Y}^{h_v,3}_{p,1}$ **f.** $\hat{Y}_* = \hat{Y}^{h_v,3}_{p,2}$

Figure 4.18 *The transform of a nonuniformly sampled sinusoid with* $f_0 = 0.55$ *Hz, is compared to the transform when uniform sampling is used. The difference between the power spectras,* $|\hat{Y}_*(f)|^2/t_N - |\hat{Y}_{us}(f)|^2/t_N$*, for different frequencies,* f*, are shown. Notice the varying y-scales.*

	$f_0 = 0.10$ Hz		$f_0 = 0.55$ Hz		$f_0 = 1.9$ Hz		$f_0 = 2.6$ Hz		
FT	$\mathbf{dP_a}$	$\mathbf{dP_r}$	$\mathbf{dP_a}$	$\mathbf{dP_r}$	$\mathbf{dP_a}$	$\mathbf{dP_r}$	$\mathbf{dP_a}$	$\mathbf{dP_r}$	**time**
$\hat{Y}_{us}$	0		0		0		0		2.3 s
$\hat{Y}^0_{ra}$	$_4$0.27	$_4$0.91	$_4$0.08	111.9	$_1$13.0	$_2$ 63	$> F_N$		0.7 s
$\hat{Y}^{h_f,3}_{p,1}$	0.78	1.04	$_2$0.06	28.1	$> f_N$		$> f_N$		19.3 s
$\hat{Y}^{h_f,3}_{p,2}$	0.39	1.44	0.14	$_2$ 8.2	$> f_N$		$> f_N$		7.4 s
$\hat{Y}^{h_v,3}_{p,1}$	0.46	1.87	0.62	17.6	31.7	$_1$ 61	31.8	$_2$ 12	64.1 s
$\hat{Y}^{h_v,3}_{p,2}$	$_3$0.20	$_1$0.60	0.55	27.9	31.7	$5e3$	$> f_N$		20.9 s
$\hat{Y}^{h_v,3}_{p,3}$	$_2$0.08	1.63	4.14	31.2	$> f_N$		$> f_N$		7.5 s
$\hat{Y}^{neq}_{sinc}$	0.29	1.09	$_3$0.07	53.8	$_2$13.2	$_3$100	$> F_N$		1.9 s
$\hat{Y}^{eq}_{sinc}$	$_4$0.27	$_3$0.90	$_5$0.09	21.8	51.3	$2e6$	$> f_N$		4.2 s
$\hat{Y}^{eq}_{sinc,d}$	$_1$0.06	1.72	0.35	$_1$ 8.1	$> f_N$		$> f_N$		1.4 s
$\hat{Y}^0_{sp}$	0.28	$_2$0.85	$_1$0.05	$_4$13.3	$_3$17.9	273	$_1$26.0	40	0.9 s
$\hat{Y}^1_{sp}$	0.56	1.45	0.14	$_3$12.7	$_4$25.7	121	30.7	$_1$ 9	2.5 s

Table 4.2 *Performance evaluation for different signal assumptions and different setups. The absolute and relative difference to $\hat{Y}_{us}$ and the evaluation time is shown. The smaller numbers indicate a ranking of the top values for each column. The Fourier transforms, FT, were described on p. 77. The evaluation times show the difference for these particular implementations performed on a SunBlade 100 using Matlab 6.1. The notation $2e6 = 2 \cdot 10^6$ is used.*

2. It is extremely hard to get good results for frequencies inside the Nyquist band, and the tails seem to cause the most trouble. Moving the sampling points seems to make it worse (cf. $\hat{Y}^0_{ra}$ and $\hat{Y}^{neq}_{sinc}$ with $\hat{Y}^{eq}_{sinc}$ and $\hat{Y}^{h_v,3}_p$).

3. $\hat{Y}^{neq}_{sinc}$ and $\hat{Y}^0_{ra}$ perform relatively similar, which indicates that A in Eq. (4.19) is not too far from the identity matrix in this case. This can be due to the fact that the Nyquist band is fairly small. The difference in evaluation time shows the difference the solving of $AX = B$ makes for this implementation.

4. It is interesting to note that estimation of the frequency, f_0, using

$$\hat{f}_0 = \arg\max_f |\hat{Y}_*(f)|,$$

gives the correct frequency for all the calculations, except $\hat{Y}_{sinc}^{neq}$ when $f_0 = 1.9$ Hz when $\hat{f}_0 = 0.02$ Hz.

5. Who is the winner for each frequency, f_0?

- When $f_0 = 0.1$ Hz, $\hat{Y}_{sinc}^{eq}$ and $\hat{Y}_{p,2}^{h_v,3}$ is best.
- When $f_0 = 0.55$ Hz, $\hat{Y}_{p,2}^{h_f,3}$ and $\hat{Y}_{sp}^{0}$ come out on top.
- Furthermore, when $f_0 = 2.6$ Hz, either $\hat{Y}_{sp}^{1}$ or $\hat{Y}_{p,1}^{h_v,3}$ performs better and
- $f_0 = 1.9$ Hz is favourable to $\hat{Y}_{ra}^{0}$. $\hat{Y}_{p,1}^{h_v,3}$ or $\hat{Y}_{sinc}^{neq}$ could also be used.

It seems favourable, for most cases, to use equidistant placement of new samples or of the basis grid. Simple spline interpolation is also a reasonable option. For $f_0 > F_N$, nothing is really a good choice and the three examined methods are equally bad. Previous investigations of the resulting continuous approximation, when a nonequidistnat grid is used, indicated that there might be troubles with this choice (Figure 4.7).

6. If frequencies above the Nyquist band is sought, there is a risk of high distortion, especially in the tails. Single peaks can be found if the tails are ignored, e.g., $\hat{Y}_{sp}^{0}$ seems to be most accurate around $f_0 = 2.6$ Hz since dP_a is relatively low and dP_r is quite high.

7. For $f_0 = 0.1$ Hz, $\hat{Y}_{sinc,d}^{eq}$ and $\hat{Y}_{p,3}^{h_f,3}$ seem to have problems with the tails but overall a good fit. In this case $\hat{Y}_{p,2}^{h_f,3}$ is relatively bad overall. This means that the different techniques for downsampling does not help in approximating the transform.

8. The Nyquist band in the example was chosen to make it possible to test for frequencies both below, inside and above it. The calculations in 4.2.4 indicate that a wider Nyquist band might be favourable.

4.5 System Sampling

After producing the transform, the continous model can be adapted to the spectrum. The other route is to use direct methods for identification. This section concerns sampling of continous systems, mainly to show the effect on the noise. The results are straighforward from simple equation solving.

Let us assume that we start with a continuous time model and want to investigate the effects of sampling nonuniformly,

$$\begin{array}{rcl} \dot{x} &=& Ax + Bu + Dv \\ y &=& Cx + e \end{array} \tag{4.42}$$

v and e are independent white Gaussian noise sources with $E[vv^T] = R_v$ and $E[ee^T] = R_e$. We leave the second equation and assume that we have measurements at times t_i with $i = 1, 2, \ldots, N$. The solution to Eq. (4.42) then becomes

$$\begin{aligned} x(t_i) &= e^{A(t_i - t_{i-1})}x(t_{i-1}) + \int_{t_{i-1}}^{t_i} e^{A(\tau - t_{i-1})}\left(Bu(\tau) + Dv(\tau)\right)d\tau \\ &= e^{AT_i}x(t_{i-1}) + A^{-1}(e^{AT_i} - I)Bu(t_{i-1}) + \\ &\quad + \int_0^{T_i} e^{A\tau}Dv(t_{i-1} + \tau)d\tau \end{aligned} \tag{4.43}$$

The second equality demands that the input, u, is constant between the measurements to be exactly true. It would also be possible to assume other structures of u based on the measurements, e.g., piecewise constant or an expansion of basis functions. The assumption about u being constant between the sample points is used to simplify calculations. It can also be argued for, when the control signal u is calculated by a digital controller. Other representations of u will not be investigated here.

We will now study the noise term further. Simple calculations lead to

$$w(t_i) \triangleq \int_0^{T_i} e^{A\tau}Dv(t_{i-1} + \tau)d\tau \tag{4.44a}$$

$$E[w] = 0 \tag{4.44b}$$

$$\begin{aligned} E[ww^T] &= \int_0^{T_i}\int_0^{T_i} e^{A\tau}DE[v(t_{i-1} + \tau)v(t_{i-1} + s)^T]D^T e^{A^T s}d\tau ds \\ &= \int_0^{T_i} e^{A\tau}DR_vD^T e^{A^T\tau}d\tau \triangleq R_w \end{aligned} \tag{4.44c}$$

The only constraint for this to hold is that $R_w < \infty$, which is assumed to be true for all systems of interest in this work. Since $v(t)$ is uncorrelated with $x(s)$ and $y(s)$ for $s \leq t$, we see that $w(t_i)$ is uncorrelated with $x(s)$ and $y(s)$ for $s \leq t_{i-1}$.

We finally get the model equations

$$\left\{ \begin{array}{rcl} x(t_i) &=& \tilde{A}_i x(t_{i-1}) + \tilde{B}_i u(t_{i-1}) + w(t_i) \\ y(t_i) &=& Cx(t_i) + e(t_i) \end{array} \right. \tag{4.45}$$

with

$$\tilde{A}_i = e^{AT_i} \approx (I + AT_i) \tag{4.46a}$$

$$\tilde{B}_i = A^{-1}(e^{AT_i} - I)B \approx BT_i \tag{4.46b}$$

and the approximations come from a first order Taylor expansion of the exponential, which means that AT_i needs to be small in some sense. The Taylor expansion can be important, when calculation time is crucial. One important fact is that R_w changes with T_i, which can be crucial in algorithms.

Example 4.3 (Random walk) *A random walk was used in Chapter 3 to describe variations of parameters. In continuous time this can be written*

$$\begin{cases} \dot{\theta}(t) &= v(t) \\ y(t) &= \varphi^T(t)\theta(t) \end{cases},$$

i.e., $A = B = 0$, $C = \varphi^T(t)$ and $D = I$ in Eq. (4.42). This means that $w(t_i) = \int_0^{T_i} v(t_{i-1} + \tau)d\tau$ and

$$R_w = R_v T_i$$

□

4.5.1 Filtering

Sampling a continuous filter, described on state-space form, nonuniformly, is straight-forward from the above discussion, using $y(t_i) = Cx(y_i) + e(t_i)$. For a simple first order filter with scalar signals $\dot{x} = ax + bu$, we get

$$x(t_i) = e^{aT_i}x(t_{i-1}) + \frac{b}{a}(e^{aT_i} - 1)u(t_{i-1}), \tag{4.47}$$

which is of course still valid when $T_i = T$, $\forall i$, i.e., uniform sampling. If aT_i is small

$$x(t_i) = (1 + aT_i)x(t_{i-1}) + bT_i u(t_{i-1}) \tag{4.48}$$

is a good approximation.

Example 4.4 (Filtering of the queue length) *In Example 3.3 the queue length was filtered with*

$$H(s) = \frac{6.5}{s + 6.5},$$

which can be written as

$$\begin{aligned} \dot{x}(t) &= -6.5x(t) + 6.5q(t) \\ y(t) &= x(t). \end{aligned}$$

The filter actually used was the nonuniform filter calculated as Eq. (4.47):

$$\begin{aligned} x(t_i) &= e^{-6.5T_i}x(t_{i-1}) - (e^{-6.5T_i} - 1)q(t_{i-1}) \\ y(t_i) &= x(t_i). \end{aligned}$$

□

Example 4.5 (RED's filter) *For a busy queue the filter in RED can be described as*

$$y(t_i) = (1 - \lambda)y(t_{i-1}) + \lambda q(t_i),$$

see Section 2.2.2. A busy queue is assumed never empty, examples are core nodes in the network or the connection between a fixed and a wireless network. Since the time difference $T_i = t_i - t_{i-1}$ is not taken into account for the nonuniformly sampled filter, the resulting continuous filter is time-varying, cf. Eq. (4.48). RED's filter can be seen as a low pass filter with a time-varying cut off frequency.
When the queue is idle, the situation is somewhat better since the time effect is considered in some sense via the parameter m, the exact impact is not clear however. Though, it is clear that it is not the correct time invariant filter for this case either. □

That is, it would be natural to replace the low-pass filter in RED with a time-invariant filter implemented for nonuniform sampling. This is an open field for further improvement.

4.6 Model Estimation

When dealing with nonuniformly sampled data there are several issues to consider. Modeling the dynamics of a system based on measurements at times t_i, can be approached in a few different ways. One objective of the model is to predict future outputs well. We will discuss three different approaches, argue for and against each of them and finally perform some of the necessary calculations for a full identification experiment.

Model 1: Use a continuous time model as a base and estimate the continuous parameters based on the measurements and then sample the model at a useful rate to predict the behavior of the system.

$$\begin{aligned} \dot{x}(t) &= A_1 x(t) + B_1 u(t) \\ y(t) &= C_1 x(t) + D_1 u(t) \end{aligned}$$

For this we need $\hat{x}(t|t_1, \ldots, t_N, \theta)$, where θ contains the parameters in the model and we have done N measurements until time t.

Model 2: A discrete time sampled model requires the knowledge of a suitable sample interval, T, for the model.

$$\begin{aligned} x(t+T) &= A_2 x(t) + B_2 u(t) \\ y(t) &= C_2 x(t) + D_2 u(t) \end{aligned}$$

The time constant can be

- chosen beforehand as a design parameter,
- chosen during calculations by comparing performance of several parallel model estimations using different time constants, or
- chosen from frequency analysis of the signal.

The preferred way is to use frequency analysis, which gives the best accuracy in theory. The parameters will depend on the choice of T.

Model 3: A nonuniformly sampled model requires an unknown number of parameters to be able to capture the basic dynamics.

$$\begin{aligned} x(t_{i+1}) &= A_3(t_i)x(t_i) + B_3(t_i)u(t_i) \\ y(t_i) &= C_3(t_i)x(t_i) + D_3(t_i)u(t_i) \end{aligned}$$

The size of the model is dependent on the basic frequency and the inter arrival times, T_i. The value of the parameters will also depend on T_i.

Every approach has some advantages over the others but they all cause problems as well. Some of the considerations that have been done are listed here. The primary aim is to find a continous model based on the data:

1. There exists no theory for estimation of a continuous time model using nonuniformly sampled data. Previous work has been done on model structures like
$$\dot{y} + ay + b = u,$$
where clever integration of the equation above is used to eliminate the derivative, see the toolbox Contsid described in Garnier and Mensler (1999). During the integration, the signals y and u are considered piecewise constant, and the authors argue that the sampling is done fast enough for this to be a reasonably good approximation. In Larsson and Söderström (2002), continuous time AR processes are identified by numerical approximation of the derivative.

2. A continuous time model will describe the basic frequency without having to consider a good choice of the sampling time, T.

3. A nonuniformly sampled model does not need a sample time but still needs the same information about the basic frequency as the discrete time model. For small inter sample times this can produce an extremely large number of parameters to make sure that the main variations are captured.

4. A discrete time model necessitates interpolation between nonuniform samples to find, $y(t \pm kT)$. The techniques for local polynomial modeling discussed in Section 4.2.2 can be used, but they still require noncausality and is not useful for recursive actions.

5. Perhaps different parts could be modeled in different ways, e.g., parameter variation and signal variation can have different model types.

In summary, the nonuniformly sampled model is of large complexity and the required information does not differ much from that of the discrete time model. The continuous time model is more suitable for recursive calculations while the discrete time model is simpler to handle. The following discussions will include both these options in parallel.

4.6.1 Cost Function

When a model is to be fitted to measured data a cost function, $V(\theta)$, is chosen and then minimized with respect to the parameterization of the model, here denoted θ. For *Model 1* the natural way is to minimize the distance between the model output, $\hat{y}$, in the measurement points, t_i, and the actual measurements, y_i, i.e.,

$$V(\theta) = \sum |y(t_i) - \hat{y}(t_i|\theta)|^2 . \tag{4.49}$$

If *Model 2* is used it is more natural to look at the correspondence of the model with the signal at the model discretization times, kT, i.e.,

$$V(\theta) = \sum |y(kT) - \hat{y}(kT|\theta)|^2 . \tag{4.50}$$

The first difference is the weighting of measurements. Using Eq. (4.49) it will be more important with a close fit when the measurements lie close (small T_i). This can be solved by using weights in Eq. (4.49) in a clever way. Depending on the method used for resampling, Eq. (4.50) can even miss samples if the sampling interval, T, is much larger than the measurement intervals, T_i. Secondly, in the second approach the model is compared to values calculated from measurement data, not to real measurements. Depending on the calculation of $y(kT)$, Eq. (4.50) can yield different results.

4.6.2 Output Estimation

In Chapter 3, the queue length dynamics was modeled with a biased AR-model. Such a model is described in state space form with $D = 0$ and $u(t) = 1$. The rest of the description focuses on calculations for this model type. Most parts are straightforward to extend for other choices of $u(t)$.

For the two cases, the states, x, have to be estimated in order to find the output estimates, $\hat{y}$. The state estimates $\hat{x}$, are found by using the model and correcting with the error between the measurement and the model output. Starting again with *Model 1*, somewhere along the way it becomes necessary to sample the model to find the estimates at times t_i. Two versions are possible:

- Start by designing the state estimate update equation and then sample it to get

$$\begin{aligned}\dot{\hat{x}}(t) &= A_1\hat{x}(t) + B_1 + K(y(t) - C_1\hat{x}(t)) \\ \hat{x}(t_i) &= e^{(A_1 - KC_1)T_i}\hat{x}(t_{i-1}) + S_iB_1 + S_iKy_{i-1}. \\ S_i &= (A_1 - KC_1)^{-1}(e^{(A_1-KC_1)T_i} - I)\end{aligned} \tag{4.51}$$

  The sampling phase means that y is assumed to be piecewise constant between the measurements, the same holds for u but since we only consider the case when $u(t) = 1$ this is of less importance.

- Sample the model equation first and then design the state update equation to get

$$\begin{aligned}x(t_i) &= e^{A_1T_i}x(t_{i-1}) + A_1^{-1}(e^{A_1T_i} - I)B_1 \\ \hat{x}(t_i) &= e^{A_1T_i}\hat{x}(t_{i-1}) + A_1^{-1}(e^{A_1T_i} - I)B_1 + K_i(y_i - C_1\hat{x}(t_{i-1}))\end{aligned}. \tag{4.52}$$

  The scaling of the estimation error, K, depends on time since the system description changes over time, $\tilde{A}_i = E^{A_1T_i}$.

For *Model 2* it is straightforward to design the state estimate update equation.

$$\hat{x}((k+1)T) = A_2\hat{x}(kT) + B_2 + K(kT)(y(kT) - C_2\hat{x}(kT)) \tag{4.53}$$

For all cases

$$\hat{y} = C\hat{x}$$

is used to find the output estimate at the wanted time.

4.6.3 Calculation of Optimal K

Algorithm 3.2 describes the Kalman Filter for parameter estimation but the same algorithm can be used as a state estimator.

Algorithm 4.2 (KF for nonuniformly sampled data)

Given the state space model

$$\begin{aligned}x(t_i) &= A_ix(t_{i-1}) + B_i + w(t_i) \\ y(t_i) &= C_ix(t_i) + e(t_i)\end{aligned}$$

the state estimation is done as follows

$$\begin{aligned}\hat{x}(t_i) &= A_i\hat{x}(t_{i-1}) + B_i + K_i(y_i - C_i\hat{x}(t_{i-1})) \\ S_i &= C_iP_{i-1}C_i^T + R_i \\ K_i &= A_iP_{i-1}C_i^TS_i^{-1} \\ P_i &= A_iP_{i-1}A_i + Q_i - A_iP_{i-1}C_i^TS_i^{-1}C_iP_{i-1}A_i^T\end{aligned}$$

$R_i = E[e(t_i)Te(t_i)]$ and $Q_i = E[w(t_i)^T w(t_i)]$ describe the properties of the measurement and process noises. The estimation of the output is then

$$\hat{y}(t_i) = C_i \hat{x}(t_i).$$

The Kalman Filter is insensitive to the chosen time scale, as long as the noise representations reflect it, i.e., t_i can be the nonuniform sample points or the uniform resampling points depending on the chosen model structure. How to deal with the noise representation was discussed in Section 4.5.

Example 4.6 (AR modeling of queue length dynamics) *In Example 3.5 a fifth order AR model is adapted to the queue length data. The random walk described in Example 4.3 was used for parameter changes, where the changing noise description was plugged into the Kalman Filter in Algorithm 4.2 with $A_i = I$, $B_1 = 0$,*

$$C_i = \varphi^T(t_i) = \begin{pmatrix} y(t_i - T) & \dots & y(t_i - 5T) & 1 \end{pmatrix},$$

$R_i = 0$ and $Q_i = R_v T_i$. The calculation on θ was done at the measurement points but uniform sampling was used for the model description. This means that the cost function Eq. (4.49) was used together with Model 2. □

4.7 Summary

Modeling and identification based on nonuniform sampling can be done in a variety of ways. This chapter investigated two alternatives: adopt a continuous model to the Fourier transform, or use the samples directly. Several approximations of the Fourier transform based on nonuniform sampling were presented and evaluated. These transform approximations can be used to adapt to a continuous time model. The approximations were done using approximations of the integrand, resampling and reconstruction of the continuous signal. The chapter also discussed identification directly in the continuous domain.

In the previous chapter, the queue length dynamics wer modeled as a sum of sinusoids. Therefore, the evaluation of the transform approximations was based on a single frequency. The transform approximations are linear in the measurements and therefore superposition of different frequencies apply. The performance evaluation of the approximations showed no single answer on how to approximate the transform based on nonuniform sampling. Resampling or simple spline interpolation seemed to be favorable. Calculations of expected value for the approximation when splines of order 0 were used to reconstruct the signal showed that the width of the support for the distribution had great impact on the appearance of the transform. This was done for the case when the inter sample times were rectangularly distributed.

It was shown that for state-space models, standard identification methods can be used after sampling the system, if only consideration is taken to varying noise characteristics. System identification using state-space models is described in Ljung (1999).

Appendices

4.A Calculation of the Mean for a Sum of sinc-Functions

Eq. (4.27) should be calculated using Eq. (4.15) with $H(t) = \mathrm{sinc}(t)$.

$$\mu_{sinc} = \frac{1}{t_N}\int_0^{t_N} \sum_k c_k \,\mathrm{sinc}(a_k(t-b_k))dt = \frac{1}{t_N}\sum_k \frac{c_k}{a_k}\left(g(a_k(t_n-b_k)) + g(a_k b_k)\right)$$

Here

$$g(x) = \int_0^x \mathrm{sinc}(t)dt \tag{4.54}$$

is calculated using the Taylor expansion for a sinusoid and then dividing by πx.

$$\begin{aligned}
\sin(x) &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots \\
&= \sum_{n=1}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!} \\
\mathrm{sinc}(x) &= \frac{\sin(\pi x)}{\pi x} \\
&= \frac{\sum_n (-1)^n \frac{(\pi x)^{2n+1}}{(2n+1)!}}{\pi x} \\
&= \sum_{n=0}^{\infty}(-1)^n \frac{(\pi x)^{2n}}{(2n+1)!} \\
\int \mathrm{sinc}(x)dx &= \sum_{n=0}^{\infty}(-1)^n \frac{\pi^{2n}x^{2n+1}}{(2n+1)(2n+1)!}
\end{aligned} \tag{4.55}$$

From this expression, it is easy to see that

$$g(-x) = -g(x). \tag{4.56}$$

Finally,

$$\int_{x_0}^{x_1} \mathrm{sinc}(a(x-b))dx = \frac{1}{a}\left(g(a(x_1-b)) - g(a(x_0-b))\right) \tag{4.57}$$

gives the resulting mean value:

$$\mu_{sinc} = \frac{1}{t_N}\sum_k \sum_{n=0}^{\infty}(-\pi^2)^n c_k \frac{a_k^{2n}\left((t_N-b_k)^{2n+1} - b_k^{2n+1}\right)}{(2n+1)(2n+1)!}$$

A large amount of memory is needed to calculate $g(x)$ from Eq. (4.55) for high values of x and n. In Matlab the sum is possible to calculate for n up to 200 and the mean value is then correct for x up to approximately 10. A tabular of values of $g(x)$, Eq. (4.54), is given in Table 4.3. Together with Eqs. (4.56) and (4.57) the values in the tabular can be used to find all mean values. For

x	$g(x)$	x	$g(x)$	x	$g(x)$	x	$g(x)$
0.01	0.0100	0.31	0.2941	0.61	0.4984	0.91	0.5852
0.02	0.0200	0.32	0.3026	0.62	0.5033	0.92	0.5861
0.03	0.0300	0.33	0.3109	0.63	0.5080	0.93	0.5869
0.04	0.0400	0.34	0.3192	0.64	0.5125	0.94	0.5876
0.05	0.0499	0.35	0.3273	0.65	0.5170	0.95	0.5882
0.06	0.0599	0.36	0.3354	0.66	0.5213	0.96	0.5887
0.07	0.0698	0.37	0.3433	0.67	0.5254	0.97	0.5890
0.08	0.0797	0.38	0.3512	0.68	0.5295	0.98	0.5893
0.09	0.0896	0.39	0.3589	0.69	0.5333	0.99	0.5894
0.10	0.0995	0.40	0.3665	0.70	0.5371	1.0	0.5895
0.11	0.1093	0.41	0.3740	0.71	0.5407	1.1	0.5848
0.12	0.1191	0.42	0.3814	0.72	0.5442	1.2	0.5724
0.13	0.1288	0.43	0.3887	0.73	0.5475	1.3	0.5545
0.14	0.1385	0.44	0.3959	0.74	0.5507	1.4	0.5336
0.15	0.1482	0.45	0.4029	0.75	0.5538	1.5	0.5120
0.16	0.1578	0.46	0.4099	0.76	0.5567	1.6	0.4917
0.17	0.1673	0.47	0.4167	0.77	0.5595	1.7	0.4746
0.18	0.1768	0.48	0.4233	0.78	0.5622	1.8	0.4618
0.19	0.1863	0.49	0.4299	0.79	0.5647	1.9	0.4540
0.20	0.1957	0.50	0.4363	0.80	0.5671	2.0	0.4514
0.21	0.2050	0.51	0.4426	0.81	0.5694	2.1	0.4538
0.22	0.2142	0.52	0.4488	0.82	0.5715	2.2	0.4605
0.23	0.2234	0.53	0.4548	0.83	0.5736	2.3	0.4704
0.24	0.2325	0.54	0.4608	0.84	0.5754	2.4	0.4825
0.25	0.2416	0.55	0.4665	0.85	0.5772	2.5	0.4952
0.26	0.2506	0.56	0.4722	0.86	0.5788	2.6	0.5075
0.27	0.2594	0.57	0.4777	0.87	0.5804	2.7	0.5182
0.28	0.2682	0.58	0.4831	0.88	0.5818	2.8	0.5263
0.29	0.2770	0.59	0.4883	0.89	0.5830	2.9	0.5314
0.30	0.2856	0.60	0.4935	0.90	0.5842	3.0	0.5331

Table 4.3 *Values of $g(x)$ for different x. Eqs.* (4.56) *and* (4.57) *can be used to get values of the integral.*

higher values

$$\hat{g}(x) = 0.5\left(1 - \frac{0.199}{x^{0.99}}\cos(\pi x)\right)$$

gives very good accuracy ($|g(x) - \hat{g}(x)| < 3 \cdot 10^{-3}$). The approximation of g was

found by comparing a step response of a first order system to the function plot of $g(x)$. Then parameter fitting was done for a few measurement points.

4.B Limited Distributions

Let $f(t)$ be a probability density function with infinite support, i.e., $f(t) > 0$, $\forall t$. To construct a limited distribution, $g(t)$, between t_0 and t_1, let

$$g(t) = \begin{cases} \frac{1}{g_0} f(t) & t_0 \le t \le t_1 \\ 0 & \text{otherwise} \end{cases} .$$

The scaling factor g_0 makes sure that $\int g(t)dt = 1$, i.e., $g_0 = \int_{t_0}^{t_1} f(t)dt$.

If $f(t)$ is the exponential distribution $g(t)$ becomes:

$$\begin{aligned} \Delta T &= t_1 - t_0 \\ f(t) &= \lambda e^{-\lambda t} \\ \int_{t_0}^{t_1} f(t)dt &= e^{-\lambda t_0}(1 - e^{-\lambda \Delta T}) \\ g(t) &= \lambda \frac{e^{-\lambda(t-t_0)}}{1 - e^{-\lambda \Delta T}}, \quad t_0 \le t \le t_1. \end{aligned}$$

To find the mean for the distribution note that

$$\int_{t_0}^{t_1} \lambda(t - t_0)e^{-\lambda t}dt = \left[-(t - t_0)e^{-\lambda t} + \frac{e^{-\lambda t}}{-\lambda}\right]_{t=t_0}^{t_1} =$$
$$= e^{-\lambda t_0}/\lambda - e^{-\lambda t_1}(\Delta T + 1/\lambda) = \frac{e^{-\lambda t_0}}{\lambda}(1 - (1 + \lambda \Delta T)e^{-\lambda \Delta T})$$

then let $m = \int tg(t)dt$ and the resulting equation

$$\lambda(1 - e^{-\lambda \Delta T})(m - t_0) = e^{-\lambda t_0}(1 - (1 + \lambda \Delta T)e^{-\lambda \Delta T})$$

gives the mean, m. Inspection shows that $m = 1/\lambda$ when $t_0 = 0$ and $t_1 = \infty$ as expected.

Conclusions

5

This thesis presents three parts of the network control problem: An overview, a design case and background theory. It is a part of the efforts that are needed to manage future networks. The work shows promising results in using adaptive queue management to improve transmission performance over wired or wireless links. It also adds to the knowledge about interacting control structures on the network.

The different parts can be completely separated. It is not necessary to have any understanding of the underlying application to be able to understand the improvements made in queue management or the usefulness of the theory for nonuniform signal processing.

By using systems theory several control structures at different levels on the Internet are highlighted. Important connections between these controllers are also discovered as well as cases where simplifications are appropriate.

This work concerns the task of improving performance over a network system by intelligent dropping of packets at the nodes. This is done by modeling of the queue length dynamics and controlling packet drops. It is shown that biased autoregressive models of the queue length dynamics capture the main dynamics. The control technique is built on the existing structure and showed promising results. It is shown that predictions of the queue length can be used successfully to increase the throughput and that control based on the derivative damps queue length oscillations.

A major part of the network control is triggered by packet arrivals and departures. The control decisions in Adaptive Queue Management, e.g., are calculated for each new packet arrival. A discussion about nonuniform signal processing presents two routes toward continuous modeling of a nonuniformly sampled signal; using transform approximations or direct methods. Several methods to approximate the Fourier transform from nonuniform samples are presented and evaluated. Resampling uniformly or spline interpolation of nonuniform samples produce the most accurate transform.

Filtering and ordinary system identification based on nonuniform samples are presented and also illustrated with examples. The commonly used filter in todays adaptive queue management algorithms are interpreted based on this discussion. It is also shown how the noise representation should be done in identification of continuous time models at nonuniform samples.

This work leaves room for several improvements and further investigations. The overall view of network control can be further developed and used for analysis. Control algorithms for other layers can be designed and evaluated. When studying network queue management, modeling and control were kept simple as well as basic calculations. It is believed that the major improvements can be done by improved implementations; using more sophisticated control structures; and automatic tuning of the sampling time for the models. There are a lot of connections between different approximations of the Fourier transform that need to be further investigated. Evaluations of control based on models from transform approximations should be done, and investigations of implementation aspects for the different methods.

Further work that would be very interesting, is implementation and testing of the proposed methods against other simulators and also real systems, especially for radio applications.

Bibliography

E. Altman, T. Başar, and N. Hovakimian. Worst-case rate-based flow control with an ARMA model of the available bandwidth. *Annals of Dynamic Games*, 2000.

A. Andersen and B. Nielsen. A markovian approach for modeling packet traffic with long-range dependence. *IEEE Journal on Selected Areas in Communications*, 1998.

S. Andersson. *Hidden Markov Models — Traffic Modeling and Subspace Methods*. PhD thesis, Lund Institute of Technology, 2002.

H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP behavior of a busy internet server: analysis and improvements. In *IEEE INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.*, volume 1, pages 256–262, 1998.

I. Bilinskis and A. Mikelsons. *Randomized Signal Processing*. Prentice Hall, London, 1992.

D. M. Bland and A. Tarczynski. Optimum nonuniform sampling sequence for alias frequency suppression. In *IEEE International Symposium on Circuits and Systems*, Jun 1997.

L. S. Brakmo and L. L. Peterson. TCP Vegas: end to end congestion avoidance on a global internet. *Selected Areas in Communications, IEEE Journal on*, 13:1465–1480, 1995.

K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad hoc wireless networks. *IEEE Personal Communications*, Feb 2001.

G. Dahlquist and A. Björck. *Numerical Methods*. Prentice-Hall, 1974.

X. Deng, S. Yi, G. Kesidis, and C. R. Das. Stabilized virtual buffer (SVB) - an active queue management scheme for internet quality-of-service. In *Proc. IEEE Globecom*, 2002.

J. Elbornsson. *Analysis, Estimation and Compensation of Mismatch Effects in A/D Converters*. PhD thesis, Linköpins universitet, 2003. Dissertation No. 811.

W.-C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. BLUE: A new class of active queue management algorithms. Technical Report CSE-TR-387-99, 1999. URL `citeseer.nj.nec.com/feng99blue.html`.

S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, Oct. 1994.

S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

H. Garnier and M. Mensler. CONTSID — A continous-time system identification toolbox for matlab. In *5th European Control Conference*, Aug 1999.

T. Glad and L. Ljung. *Control Theory - Multivariable and Nonlinear Methods*. Taylor and Francis, London and New York, 2000.

F. Gunnarsson. Problems and improvements of internet traffic congestion control. Master's thesis, Linköpings Universitet, Oct. 2000. LiTH-ISY-EX-3098.

F. Gunnarsson. Frequency analysis using nonuniform sampling with applications to adaptive queue management. Submitted to *Acoustics, Speech, and Signal Processing., IEEE International Conference on*, May 2004.

F. Gunnarsson, A. Björsson, B. Knutsson, F. Gunnarsson, and F. Gustafsson. Radio access network (UTRAN) modeling for heterogenous network simulations. Technical Report LiTH-ISY-R-2533, Dept. of Electrical Engineering, Linköpings universitet, Aug. 2003a. URL `www.control.isy.liu.se/publications`.

F. Gunnarsson, A. Björsson, B. Knutsson, F. Gunnarsson, and F. Gustafsson. Radio access network (UTRAN) modeling for heterogenous network simulations, a brief description. In *First Swedish National Computer Networking Workshop*, Sept. 2003b.

F. Gunnarsson, F. Gunnarsson, and F. Gustafsson. TCP performance based on queue occupation. In *Towards 4G and Future Mobile Internet, Proceedings for Nordic Radio Symposium 2001*, Mar. 2001.

F. Gunnarsson, F. Gunnarsson, and F. Gustafsson. Issues on performance measurements of TCP. In *Radiovetenskap och Kommunikation '02, RVK'02*, June 2002.

F. Gunnarsson, F. Gunnarsson, and F. Gustafsson. Controlling internet queue dynamics using recursively identified models. To appear in *42nd IEEE Conference on Decision and Control*, Dec. 2003c.

C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. A control theoretic analysis of RED. In *INFOCOM 2001. Proceedings. IEEE*, 2001a.

C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *INFOCOM 2001. Proceedings. IEEE*, volume 3, 2001b.

H. Holma and A. Toskala, editors. *WCDMA for UMTS*. John Wiley & Sons, Ltd, 2000.

R. Jain. The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation and modeling. *SIGMETRICS Performance Evaluation Review*, 19(2):5–11, 1991.

F. Kelly. Fairness and stability of end-to-end congestion control. *European Journal of Control*, 9, 2003.

S. Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley, 1997. URL `http://www.cs.cornell.edu/skeshav/real/`. Description of the REAL simulator.

K. Kilkki. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999. ISBN 1-57870-132-5.

S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue algorithm for active queue management. In *Proc. ACM Sigcomm*, 2001. URL `citeseer.nj.nec.com/kunniyur01analysis.html`.

E. K. Larsson and T. Söderström. Identification of continous-time AR processes from unevenly sampled data. *Automatica*, Apr 2002.

B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. A brief history of the internet, 2000. URL `www.isoc.org/internet/history/brief.shtml`.

J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, Jul 2001.

L. Ljung. *System Identification – Theory for the User, 2nd edition*. Prentice Hall, 1999.

L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. The MIT Press, 1983.

S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, Feb. 2002.

F. Marvasti. *Zero-Crossings and Nonuniform Sampling of Single and Multi-dimensional Signals and Systems*. Nonuniform, 1987.

F. Marvasti. Nonuniform sampling theorem for bandpass signals at or below the nyquist density. *IEEE Transactions on Signal Processing*, Mar 1996.

F. Marvasti, M. Analoui, and M. Gamshadzahi. Recovery of signals from nonuniform samples using iterative methods. *IEEE Transactions on Signal Processing*, Apr 1991.

S. Mascolo. Congestion control in high-speed communication networks using the smith principle. *Automatica*, 35(12), Dec. 1999.

S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of MobiCom 2001*, 2001.

Matlab. URL `http://www.mathworks.com`.

M. Meyer. TCP performance over GPRS. In *Wireless Communications and Networking Conference, 1999. IEEE*, 1999.

A. Misra, J. S. Baras, and T. Ott. Window distribution of multiple TCP's with random loss queues. In *Proceedings of Globecom'99*, 1999.

NS. UCB/LBNL/VINT Network Simulator, *ns-2*.

F. Papenfuss, Y. Artyukh, E. Boole, and D. Timmermann. Optimal sampling functions in nonuniform sampling driver designs to overcome the nyquist limit. In *Acoustics, Speech, and Signal Processing, 2003., IEEE International Conference on*, Apr 2003.

A. Papoulis. *Signal Analysis*. McGraw-Hill, 1977.

K.-J. Park, H. Lim, T. Başar, and C. ho Choi. Anti-windup compensator for active queue management in TCP networks. *Control Engineering Practice*, Oct. 2003.

V. Paxson and S. Floyd. Wide-area traffic: the failure of poisson modeling. In *ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Communications architectures, protocols and applications*, Oct 1994.

J. Peisa and M. Meyer. Analytical model for TCP file transfers over UMTS. In *3G Wireless 2001*, 2001.

N. Persson. Event based sampling with application to spectral estimation. Licenciate Thesis, No. 981, Dep. of Electrical Engineering, Linköpings Universitet, 2002.

J. Roll. *Local and Piecewise Affine Approaches to System Identification*. PhD thesis, Linköping universitet, 2003. Thesis No. 802.

J. Sachs and M. Meyer. Mobile internet - performance issues beyond the radio interface. In *Aachen Symposium on Signal Theory 2001*, 2001.

W. R. Stevens. *TCP/IP illustrated vol 1:The protocols*. Addison Wesley, 1994.

M. Unser. Splines. *IEEE Signal Processing Magazine*, Nov 1999.

H. van Steenis and J. Tulen. A comparison of two methods to calculate the heart rate spectrum based on non-equidistant sampling. In *Proceedings of the Annual Int. Conf. of the IEEE*. Engineering in Medicine and Biology Society, 1991.

R. H. Zakon. Hobbes' internet timeline v6.0, Feb. 2003. URL `http://www.zakon.org/robert/internet/timeline/`.

C. Zhu, O. W. W. Yang, L. Aweya, M. Ouellette, and D. Y. Montuno. A comparison of active queue management algorithms using the OPNET modeler. *IEEE Communications Magazine*, 2002.

3rd Generation Partnership Project specifications

`http://www.3gpp.org/`

3GPP TS 23.107. Quality of Service (QoS) concept and architecture, version 5.5, 2002.

3GPP TS 25.214. Physical layer procedures (FDD), version 5.5, 2003.

3GPP TS 25.301. Radio Interface Protocol Architecture, version 5.2, 2002.

3GPP TS 25.322. Radio Link Control (RLC) protocol specification, version 5.2, 2002.

3GPP TS 25.401. UTRAN overall description, version 5.5, 2002.

Request For Comments

`http://www.cis.ohio-state.edu/cs/Services/rfc/`

RFC 0768. User Datagram Protocol. Full standard, 1980. J. Postel (ed.).

RFC 0791. Internet Protocol. Full standard, 1981.

RFC 0793. Transmission Control Protocol. Full standard, 1981.

RFC 0959. File Transfer Protocol (FTP). Full standard, 1985. J. Postel and J. Reynolds.

RFC 1889. RTP: A Transport Protocol for Real-Time Applications. Proposed standard, Jan. 1996. H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson.

RFC 2474. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Proposed standard, Dec 1998. K. Nichols, S. Blake, F. Baker and D. Black.

RFC 2581. TCP Congestion Control. Proposed standard, Apr. 1999. M. Allman, V. Paxson and W. Stevens.

RFC 2616. Hypertext Transfer Protocol – HTTP/1.1. Draft standard, June 1999. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee.

RFC 2821. Simple Mail transfer Protocol. Proposed standard, Apr. 2001. J. Postel (ed).

RFC 3168. The Addition of Explicit Congestion Notification (ECN) to IP. Proposed standard, 2001. K. Ramakrishnan, S. Floyd and D. Black.

Tekn. lic. Dissertations
Division of Automatic Control and Communication Systems
Linköping University

P. Andersson: Adaptive Forgetting through Multiple Models and Adaptive Control of Car Dynamics. Thesis No. 15, 1983.

B. Wahlberg: On Model Simplification in System Identification. Thesis No. 47, 1985.

A. Isaksson: Identification of Time Varying Systems and Applications of System Identification to Signal Processing. Thesis No 75, 1986.

G. Malmberg: A Study of Adaptive Control Missiles. Thesis No 76, 1986.

S. Gunnarsson: On the Mean Square Error of Transfer Function Estimates with Applications to Control. Thesis No. 90, 1986.

M. Viberg: On the Adaptive Array Problem. Thesis No. 117, 1987.

K. Ståhl: On the Frequency Domain Analysis of Nonlinear Systems. Thesis No. 137, 1988.

A. Skeppstedt: Construction of Composite Models from Large Data-Sets. Thesis No. 149, 1988.

P. A. J. Nagy: MaMiS: A Programming Environment for Numeric/Symbolic Data Processing. Thesis No. 153, 1988.

K. Forsman: Applications of Constructive Algebra to Control Problems. Thesis No. 231, 1990.

I. Klein: Planning for a Class of Sequential Control Problems. Thesis No. 234, 1990.

F. Gustafsson: Optimal Segmentation of Linear Regression Parameters. Thesis No. 246, 1990.

H. Hjalmarsson: On Estimation of Model Quality in System Identification. Thesis No. 251, 1990.

S. Andersson: Sensor Array Processing; Application to Mobile Communication Systems and Dimension Reduction. Thesis No. 255, 1990.

K. Wang Chen: Observability and Invertibility of Nonlinear Systems: A Differential Algebraic Approach. Thesis No. 282, 1991.

J. Sjöberg: Regularization Issues in Neural Network Models of Dynamical Systems. Thesis No. 366, 1993.

P. Pucar: Segmentation of Laser Range Radar Images Using Hidden Markov Field Models. Thesis No. 403, 1993.

H. Fortell: Volterra and Algebraic Approaches to the Zero Dynamics. Thesis No. 438, 1994.

T. McKelvey: On State-Space Models in System Identification. Thesis No. 447, 1994.

T. Andersson: Concepts and Algorithms for Non-Linear System Identifiability. Thesis No. 448, 1994.

P. Lindskog: Algorithms and Tools for System Identification Using Prior Knowledge. Thesis No. 456, 1994.

J. Plantin: Algebraic Methods for Verification and Control of Discrete Event Dynamic Systems. Thesis No. 501, 1995.

J. Gunnarsson: On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods. Thesis No. 502, 1995.

A. Ericsson: Fast Power Control to Counteract Rayleigh Fading in Cellular Radio Systems. Thesis No. 527, 1995.

M. Jirstrand: Algebraic Methods for Modeling and Design in Control. Thesis No. 540, 1996.

K. Edström: Simulation of Mode Switching Systems Using Switched Bond Graphs. Thesis No. 586, 1996.

J. Palmqvist: On Integrity Monitoring of Integrated Navigation Systems. Thesis No. 600, 1997.

A. Stenman: Just-in-Time Models with Applications to Dynamical Systems. Thesis No. 601, 1997.

M. Andersson: Experimental Design and Updating of Finite Element Models. Thesis No. 611, 1997.

U. Forssell: Properties and Usage of Closed-Loop Identification Methods. Thesis No. 641, 1997.

M. Larsson: On Modeling and Diagnosis of Discrete Event Dynamic systems. Thesis No. 648, 1997.

N. Bergman: Bayesian Inference in Terrain Navigation. Thesis No. 649, 1997.

V. Einarsson: On Verification of Switched Systems Using Abstractions. Thesis No. 705, 1998.

J. Blom, F. Gunnarsson: Power Control in Cellular Radio Systems. Thesis No. 706, 1998.

P. Spångéus: Hybrid Control using LP and LMI methods – Some Applications. Thesis No. 724, 1998.

M. Norrlöf: On Analysis and Implementation of Iterative Learning Control. Thesis No. 727, 1998.

A. Hagenblad: Aspects of the Identification of Wiener Models. Thesis no 793, 1999.

F. Tjärnström: Quality Estimation of Approximate Models. Thesis no 810, 2000.

C. Carlsson: Vehicle Size and Orientation Estimation Using Geometric Fitting. Thesis no 840, 2000.

J. Löfberg: Linear Model Predictive Control: Stability and Robustness. Thesis no 866, 2001.

O. Härkegård: Flight Control Design Using Backstepping. Thesis no 875, 2001.

J. Elbornsson: Equalization of Distortion in A/D Converters. Thesis No. 883, 2001.

J. Roll: Robust Verification and Identification of Piecewise Affine Systems. Thesis No. 899, 2001.

I. Lind: Regressor Selection in System Identification using ANOVA. Thesis No. 921, 2001.

R. Karlsson: Simulation Based Methods for Target Tracking. Thesis No. 930, 2002.

P-J. Nordlund: Sequential Monte Carlo Filters and Integrated Navigation. Thesis No. 945, 2002.

M. Östring: Identification, Diagnosis, and Control of a Flexible Robot Arm. Thesis No. 948, 2002.

J. Jansson: Tracking and Decision Making for Automotive Collision Avoidance. Thesis No. 965, 2002.

C. Olsson: Active Engine Vibration Isolation using Feedback Control. Thesis No. 968, 2002.

N. Persson: Event Based Sampling with Application to Spectral Estimation. Thesis No. 981, 2002.

D. Lindgren: Subspace Selection Techniques for Classification Problems. Thesis No. 995, 2002.

E. Geijer Lundin: Uplink Load in CDMA Cellular Systems. Thesis No. 1045, 2003.

M. Enqvist: Some Results on Linear Models of Nonlinear Systems. Thesis No. 1046, 2003.

T. Schön: On Computational Methods for Nonlinear Estimation. Thesis No. 1047, 2003.