Czech Technical University in Prague

Faculty of Transportation Sciences

Department of Applied Mathematics

Estimation of Models with Uniform Innovations and its Application on Traffic Data

Thesis

Lenka Pavelková

PhD study program: Engineering Informatics
Branch of study: Engineering Informatics in Transportation and Telecommunications

Prague, December 2008

I would like to thank my supervisor doc. Ing. Ivan Nagy, CSc.
and my boss Ing. Miroslav Kárný, DrSc.
for their support during the writing of this dissertation.

This thesis was written out during the combined PhD study in the Department of Applied Mathematics in the Faculty of Transportation Sciences, the Czech Technical University in Prague.

Candidate:	Ing. Lenka Pavelková Katedra aplikované matematiky Fakulta dopravní ČVUT Konviktská 20, 110 00 Praha 1
Supervisor:	Doc. Ing. Ivan Nagy, CSc. Katedra aplikované matematiky Fakulta dopravní ČVUT Konviktská 20, 110 00 Praha 1

Date of state exam: 21th June 2006

Date of thesis submission:

Contents

Chapter 1

Introduction

1.1 Motivation

The state estimation is an important subtask of a range decision making problems relying on a linear-in-state model. The standard method of solving this problem is Kalman filtering [1, 2]. The Kalman filter (KF) uses model with normally distributed innovations. The KF gives good results in many applications.

Nevertheless, an unbounded support of normal distribution may cause troubles in some application where real quantities are bounded. Moreover, still there is no methodology of choosing innovation covariances and it is difficult to combine KF with hard restrictions on state ranges.

The mentioned drawbacks can be avoided by assuming that the innovations involved have a distribution with restricted support.

A linear state-space model with uniform innovations (LU model) studied in this work, complements KF in problems of this type. Strictly speaking, LU model will be introduced and the Bayesian solution [3, 4] of estimation problem will be developed. The proposed estimation algorithms will be applied on the transportation data.

1.2 State of the art

This section summarizes selected results achieved world-wide in the area of system description and identification. The motivation has driven the selection.

In the summary, the term identification means the parameter or state estimation for a given mathematical model. The term recursive means that the estimation is running on-line and the estimates are permanently refined. On the other hand, off-line estimation is realized in one shot for given data.

The real system is often modelled by a black-box, locally valid, model. Autoregressive model with exogenous inputs (ARX model), see [3], is an important representative of this model class.

The ARX model is described by the equation reflecting the linear relation between system outputs and inputs. This relation is determined by the model parameters. The uncertainty of the input-output relation is characterized by the innovations, stochastic unobserved stimulus of the model. The innovations are supposed to be white, zero mean and with time-invariant variance. Mostly, the innovations are assumed to be normal. The estimation procedure employs the method of the least squares. This gives good results in many applications, see e.g. [5, 6].

To get more precise model, unobserved quantities (states) are considered as well. The system is then described by a state space model and not only parameter estimation but also the subtask of the state estimation arises. The innovations of state evolution as well as observation model are usually supposed to have normal distribution. Kalman filtering (KF), see [2], is then the first-option estimation method. The main advantage of the KF is the simplicity but its use is restricted by assumed knowledge of the parameters including innovation covariances. So the various extension of KF are used like the extended Kalman filter (EKF), see [1], the iterated EKF, and the unscented Kalman filter (UKF). All these extensions are compared in [7].

Above mentioned models deal with normally distributed innovations. The Gaussian distribution has unbounded support. This fact can often be accepted as a reasonable approximation of reality, which is mostly bounded. In some case, however, this assumption is unrealistic, e.g., in modelling of the transportation systems (for encountered problems see e.g. [8]) or do not fit subsequent processing, for instance, robust control design [9]. Then, techniques similar to those dealing with unknown-but-bounded equation errors are used, see references in [10]. They often intentionally give up stochastic interpretation of the innovations and develop and analyze various algorithms of a min-max type, cf. [11]. The unknown parameters (or states) lie then within the bounded set. The complexity of this set is very high so approximation is needed to obtain recursively feasible solution. The unknown-but-bounded approaches [12, 13] face this problem by a recursive construction of a simple (typically outer) approximation of the bounded set (support of the distribution describing knowledge about estimated quantities). The approximation by an ellipsoid is described in [12]. An alternative way is an approximation by a multivariate box. This methodology is used in [13]. It brings simplicity to the subsequent use, as it provides very simple description of uncertainty. The resulting support shape is intuitively more similar to the correct one than ellipsoid. This solution can be extended so that approximated area consists of an union of non-overlapping boxes. This algorithm is described in [14].

The min-max type algorithms are definitely useful but the related decision-making tasks are unnecessarily difficult because of the broken connection to the established statistical tools.

1.3 Aims of the work

We wish to overcome the drawbacks outlined in Section 1.1 that are connected with the use of KF on systems with bounded quantities. We want to keep the advantages of the probabilistic approach and the simplicity of the estimation algorithms. This determine the main aims of the work:

- to propose the probabilistic linear state-space model with bounded innovations;
- to design algorithms for the estimation of the unknown quantities of this model;
- to demonstrate the functionality of mentioned algorithms on the simulated and traffic data.

This work aims to join two approaches, that are commonly used separately. These are (i) probabilistic approach to the modelling and estimation and (ii) principle of the bounded errors. Here, the Bayesian theory [3, 4] is used for the model identification while the model innovations are assumed to be bounded – described by the uniform distribution. This approach provides a new view on models with bounded errors as well as useful estimation algorithms.

1.4 Layout of the work

Chapter 1 is the introductory one. It contains the motivation of the work and state of the art in the area of interest. It also summarizes the aims of the thesis.

Chapter 2 summarizes basic theory and notation used throughout the thesis.

Chapter 3 contains main theoretical results of the work. In this chapter, the linear uniform state model (LU model) is introduced. Subsequently, the estimation of this model is derived here.

Chapter 4 provides the introduction to the transportation problems. The LU model of the crossroads is constructed here.

Chapter 5 contains two illustrative examples with simulated and transportation data.

Chapter 6 concludes the work. It summarizes the results achieved and formulates open research problems.

Appendix contains the Matlab codes of the estimation algorithms.

Acknowledgement
This work was supported by grants MŠMT 1M0572 (Research Center DAR), MŠMT 2C06001 (project BAYES), GAČR 102/07/1596.

Chapter 2

Underlying theory and notations

This chapter contains notation used throughout the work, basic notions from the decision-making area and the necessary theoretical base.

2.1 Abbreviations used

LP linear programming

MAP maximum a posteriori probability

pdf probability density function

LU model linear state model with uniformly distributed innovations

KF Kalman filter

2.2 Notation

$\equiv$ equality by definition

$\propto$ equality up to a constant factor, i.e. $f \propto g \Leftrightarrow f = K\,g$, K is a positive scalar constant

x^* a set of x-values, $x \in x^*$

$\mathring{x}$ the number of members in the countable set x^*

x^ℓ the length of the vector x; vectors are always columns

$\chi_x(x^*) \equiv \chi(x^*)$ the indicator of the set x^* at the point x; it equals 1 if $x \in x^*$ and it is zero otherwise

x_t, u_t, y_t unobserved state, known input and observed output of the system, respectively; the subscript $t \in t^* \subset \{0, 1, 2, \ldots\}$ labels discrete time

$d_t = (y_t, u_t)$ the data record at time t

$x_{t;i}$ is an i-th entry of the vector x at time t

$x^{k:l} \equiv [x'_k, x'_{k+1}, \ldots, x'_l]'$ the ordered sequence of states $(x_k, x_{k+1}, \ldots, x_l)$, $1 \leq k \leq l$; it can be also used in downwards sequence, i.e., $x^{l:k}$

$'$ transposition

$\underline{x}$, $\overline{x}$ lower and upper bound on x, respectively (they are used entry-wise)

∂ memory length

${}^{x}r$ the quantity r with non-numerical superscript x; this superscript is always placed to the left of the basic symbol

$M_{(\alpha,\beta)}$ matrix with α rows and β columns

$I_{(\alpha)}$ square identity matrix of the order α

$\mathbf{0}_{(\alpha,\beta)}$ zero matrix of the indicated dimensions

$\mathbf{1}_{(\alpha)}$, $\mathbf{0}_{(\alpha)}$ column vectors of ones and zeros, respectively, both of length α

$f(\cdot|\cdot)$ probability density functions (pdf); respective pdfs are distinguished by the argument names; no formal distinction is made between a random variable, its realization and an argument of the pdf

$\mathrm{supp}\,[\,f(x)]$ the support of the pdf $f : x^* \rightarrow [0, \infty]$, i.e., the subset of x^* on which $f(x) > 0$

$\mathcal{U}_x\,(\mu, {}^{x}r)$ uniform pdf of the quantity x on the box with the center μ and half-width of the support interval ${}^{x}r$; again understood entry-wise

$\setminus$ the set subtraction or an omission of a term from a sequence

$\cap$ the set intersection

$\cup$ the set union

$\emptyset$ empty set

$\hat{\Theta}$ point estimate of the parameter Θ based on the available data

$\hat{x}_t$ point estimate of the state x_t based on the available data

$G_{(\alpha,\beta)} \otimes H \equiv \begin{bmatrix} G_{11}H & \dots & G_{1\beta}H \\ \vdots & & \vdots \\ G_{\alpha 1}H & \dots & G_{\alpha\beta}H \end{bmatrix}$ *Kronecker product*

$\mathcal{R}_{col}(M)$ an operator that extends a matrix $M_{(\alpha,\beta)}$ by the zero matrix $\mathbf{0}_{(\alpha,col)}$ from the right, $\mathcal{R}_{col}(M) \equiv [M, \mathbf{0}_{(\alpha,col)}]$

$\mathcal{L}_{col}(M)$ an operator that extends a matrix $M_{(\alpha,\beta)}$ by the zero matrix $\mathbf{0}_{(\alpha,col)}$ from the left, $\mathcal{L}_{col}(M) \equiv [\mathbf{0}_{(\alpha,col)}, M]$

$\mathrm{col}(M)$ an operator that converts matrix $M_{(\alpha,\beta)}$ into a column vector of the length $\alpha\beta$, i.e., $\mathrm{col}(M) = [M_{11} \dots M_{1\beta}\ M_{21} \dots M_{2\beta} \dots M_{\alpha 1} \dots M_{\alpha\beta}]'$ where $M_{i,j}$ is an i-th row and j-th column entry of the matrix $M_{(\alpha,\beta)}$

$\delta(x)$ Dirac delta function, $\delta(x) = 0 \Leftrightarrow x \neq 0$, $\delta(x) = \infty \Leftrightarrow x = 0$
$\int \delta(x)\, dx = 1$, $\int f(x)\, \delta(x)\, dx = f(0)$, $\int f(x)\, \delta(x - x_0)\, dx = f(x_0)$

2.3 Selected notions on system identification

Quantity is a multivariate mapping. This notion corresponds to random variable used in probability theory.

Realization is a value of the quantity for its fixed argument. Mostly, the quantity and its realization are not distinguished. The proper meaning is implied by the context.

System is the part of the world that is of our interest.

System behavior $\mathcal{Q}^*$ consists of all possible realizations $\mathcal{Q}$, i.e., values of all quantities related to the system and considered within the time span determined by the horizon of interest.

System input $u \in u^*$ is an optional quantity, which is supposed to influence system behavior.

System output $y \in y^*$ is an observable quantity that provides information about the system behavior. To be or not to be output or input is a relative property. The input is always directly manipulated.

Internal quantity $X \in X^*$ is a general unobservable quantity, which influences the system output.

System state $x \in x^*$ is an unobservable quantity, which influences the system output. It contains the information on the previous evolution of the system.

System parameter $\Theta \in \Theta^*$ is an unobservable time-invariant quantity, which influences the system output. It doesn't depend on the system input.

2.4 Basics of the Bayesian approach

In Bayesian view, the system is described by probability density functions (pdfs). The statistical inference about unknown quantity is understood as correction of prior subjective pdf by objective data. The resulting posterior pdf conditioned on observed data provides a basis for a subsequent decision making, see [3], [4]. The basic operations with pdfs are described below.

2.4.1 Calculus with pdfs

Here, the main rules of the manipulating with pdfs, see e.g. [4], are summarized. The system is described by the joint pdf $f(\mathcal{Q})$ on system behavior $\mathcal{Q}^* \equiv (a,b,c)^*$. For any $(a,b,c) \in (a,b,c)^*$, the following relationships between pdfs hold:

Non-negativity	$f(a,b\|c),\ f(a\|b,c),\ f(b\|a,c),\ f(b\|c) \geq 0.$
Normalization	$\int f(a,b\|c)\,dadb = \int f(a\|b,c)\,da = \int f(b\|a,c)\,db = 1.$
Chain rule	$f(a,b\|c) = f(a\|b,c)f(b\|c) = f(b\|a,c)f(a\|c).$
Marginalization	$f(b\|c) = \int f(a,b\|c)\,da,\ f(a\|c) = \int f(a,b\|c)\,db.$
Bayes rule	$f(b\|a,c) = \frac{f(a\|b,c)f(b\|c)}{f(a\|c)} =$

$$= \frac{f(a|b,c)f(b|c)}{\int f(a|b,c)f(b|c)\,db} \propto f(a|b,c)f(b|c). \quad (2.1)$$

where $\propto$ means equality up to a constant factor.

Note that integrals used are always definite and multivariate ones. The integration domain coincides with the support of the pdf in its argument.

The meaning of basic pdfs derived from $f(\mathcal{Q})$ is as follow

$f(a,b\|c)$	joint pdf on $(a,b)^*$ conditioned by c; it restricts $f(\mathcal{Q})$ on the cross-section of $\mathcal{Q}^*$ given by a fixed c
$f(a\|c)$	marginal pdf on a^* conditioned by c; it restricts $f(\mathcal{Q})$ on the cross-section of $\mathcal{Q}^*$ given by a fixed c with no information on b
$f(b\|a,c)$	marginal pdf on b^* conditioned by a, c; it restricts $f(\mathcal{Q})$ on the cross-section of $\mathcal{Q}^*$ given by a fixed a, c

Quantities a, b are *conditionally independent* under the condition c iff

$$f(a,b|c) = f(a|c)f(b|c) \Leftrightarrow f(a|b,c) = f(a|c) \text{ or } f(b|a,c) = f(b|c). \quad (2.2)$$

2.4.2 Bayesian learning

The behavior $\mathcal{Q}$ consists generally of observable outputs $y^{1:\mathring{t}}$, optional inputs $u^{1:\mathring{t}}$ and internal quantities $X^{0:\mathring{t}}$ that are never observed directly, see definitions in Section 2.3. The collection of the outputs and inputs is called data and denoted $d^{1:\mathring{t}}$, i.e. $d_t = (y_t, u_t)$, $t \in t^* = \{1, \dots, \mathring{t}\}$. We aim to describe or influence all considered quantities, including the unobservable ones. Here, we describe how to get the estimate of internal quantities $X^{0:\mathring{t}}$ using the theoretical background in [4].

Filtration

The joint pdf $f(\mathcal{Q}) = f(d^{1:t}, X^{0:t})$ describing both observed and internal quantities is constructed from the undermentioned elements.

1. The outputs y_t are related to the past data $d^{1:t-1}$ and inputs u_t through the *observation model*

$$\{f(y_t|u_t, d^{1:t-1}, X_t) \equiv f(y_t|u_t, d^{1:t-1}, X^{1:t})\}_{t\in t^*} \tag{2.3}$$

   that is known up to unknown internal quantities $X_t \in X_t^*$.

2. The evolution of the internal quantities $X^{0:\mathring{t}} \in X^*$ is described by the *time evolution model*

$$\left\{f(X_t|u_t, d^{1:t-1}, X_{t-1}) \equiv f(X_t|u_t, d^{1:t-1}, X^{1:t-1})\right\}_{t\in t^*}. \tag{2.4}$$

3. The *controller* is described by the pdfs

$$\{f(u_t|d^{1:t-1}) \equiv f(u_t|d^{1:t-1}, X^{1:t-1})\}_{t\in t^*} \tag{2.5}$$

   Here, the validity of the *natural conditions of control* (NCC) [3] is supposed. They require the independence of u_t on $X^{1:t-1}$ when conditioned on $d^{1:t-1}$, i.e., they assume that $X^{1:t-1}$ is unknown to the controller.

4. The initial data $d_0 \equiv d^{1:0}$ coincide with the prior information about the initial internal quantity X_0 so that the prior pdf $f(X_0)$ fulfills

$$f(X_0) \equiv f(X_0|d_0). \tag{2.6}$$

   Further

$$f(X_1|u_1, X_0) \equiv f(X_1|u_1, d_0, X_0)$$

   and

$$f(y_1|u_1, X_1) \equiv (y_1|u_1, d_0, X_1)$$
$$f(u_1) \equiv f(u_1|d_0)$$

   The term $f(X_0)$, called prior pdf, is by definition the first term in the sequence of pdfs (2.4).

The evolution of the pdf $f(X_t|u_t, d^{1:t-1})$, called Bayesian filtration of unknown internal quantities X_t, is described by the following recursion that starts from the prior pdf $f(X_0)$:

- *Data updating*

$$\begin{aligned} f(X_t|d^{1:t}) &= \frac{f(y_t|u_t, d^{1:t-1}, X_t)f(X_t|u_t, d^{1:t-1})}{f(y_t|u_t, d^{1:t-1})} \\ &\propto f(y_t|u_t, d^{1:t-1}, X_t)f(X_t|u_t, d^{1:t-1}) \end{aligned} \quad (2.7)$$

  that incorporates information on the output y_t and the input u_t, and

- *Time updating*

$$f(X_{t+1}|u_{t+1}, d^{1:t}) = \int f(X_{t+1}|u_{t+1}, d^{1:t}, X_t)f(X_t|d^{1:t})\, dX_t \quad (2.8)$$

  that reflects the time evolution $X_t \to X_{t+1}$.

The model of the system $f(y_t|u_t, d^{1:t-1})$ obtained by filtering is called *predictive pdf*.

The described Bayesian filtering combines prior information in $f(X_0)$, theoretical knowledge described by $f(y_t|u_t, d^{1:t-1}, X_t)$, $f(X_t|u_t, d^{1:t-1}, X_{t-1})$ and observed data $d^{1:t} = (y^{1:t},\, u^{1:t})$ by using deductive rules of the calculus with pdfs.

In the following paragraphs, the special case of the filtering are solved.

State filtration

Here, the unknown internal quantity X coincides with the system state x, see Section 2.3, i.e., $\{X_t \equiv x_t\}_{t \in t^*}$. All relations mentioned in the previous paragraph hold with X replaced by x.

Parameter estimation

Estimation is a special version of filtering. It arises when all internal quantities are time invariant, i.e., $\{X_t \equiv \Theta\}_{t \in t^*}$. The common value Θ is called parameter, see Section 2.3.

In the case of the parameter estimation, the pdfs (2.3) – (2.6) take the following form.

1. The observation model (2.3) is known up to unknown parameter Θ

$$\{f(y_t|u_t, d^{1:t-1}, \Theta)\}_{t \in t^*}.$$

   and it is called *parametric model*.

2. The time evolution model (2.4) is
$$\left\{f(\Theta_t|u_t, d^{1:t-1}, \Theta_{t-1}) \equiv \delta(\Theta_t - \Theta_{t-1})\right\}_{t\in t^*}.$$
where the employed Dirac delta function $\delta(\cdot)$ is a formal pdf of the measure fully concentrated on the zero argument, see definition on the page 9.

3. The controller fulfilling NCC (2.5) has the same form
$$\left\{f(u_t|d^{1:t-1})\right\}_{t\in t^*} \tag{2.9}$$
Consequently, it holds (see [3])
$$\{f(\Theta|d^{1:t-1}) \equiv f(\Theta|u_t, d^{1:t-1})\}_{t\in t^*}$$

4. The prior pdf on Θ fulfills
$$f(\Theta) \equiv f(\Theta|d_0) \underbrace{=}_{(2.9)} f(\Theta|u_1, d_0). \tag{2.10}$$
and
$$f(y_1|u_1, \Theta) \equiv (y_1|u_1, d_0, \Theta)$$
$$f(u_1) \equiv f(u_1|d_0)$$

The predictive model of the system $f(y_t|u_t, d^{1:t-1})$ is given by the formula
$$f(y_t|u_t, d^{1:t-1}) = \int f(y_t|u_t, d^{1:t-1}, \Theta) f(\Theta|d^{1:t-1})\, d\Theta. \tag{2.11}$$

The evolution of the pdf $f(\Theta|d^{1:t})$ is called Bayesian parameter estimation. It generates the most general parameter estimate that coincides with the posterior pdf of the unknown parameter. Its evolution is described by the recursion identical with data updating (2.7)

$$\begin{aligned} f(\Theta|d^{1:t}) &= \frac{f(y_t|u_t, d^{1:t-1}, \Theta) f(\Theta|d^{1:t-1})}{f(y_t|u_t, d^{1:t-1})} \\ &\propto f(y_t|u_t, d^{1:t-1}, \Theta) f(\Theta|d^{1:t-1}) \end{aligned} \tag{2.12}$$

The recursion starts from the prior pdf $f(\Theta)$ (2.10). Time updating "disappears", see definition of Dirac delta function on the page 9.

The non-recursive (batch) variant of the estimation formula (2.12) is obtained by its repetitive use and looks as follows

$$f(\Theta|d^{1:t}) = \frac{\prod_{\tau=1}^{t} f(y_\tau|u_\tau, d^{1:\tau-1}, \Theta) f(\Theta)}{\int \prod_{\tau=1}^{t} f(y_\tau|u_\tau, d^{1:\tau-1}, \Theta) f(\Theta)\, d\Theta} \equiv \frac{\mathcal{L}(\Theta, d^{1:t}) f(\Theta)}{\mathcal{I}(d^{1:t})}. \tag{2.13}$$

The introduced *likelihood function*

$$\mathcal{L}(\Theta, d^{1:t}) \equiv \prod_{\tau=1}^{t} f(y_\tau|u_\tau, d^{1:\tau-1}, \Theta)$$

evolves according to the recursion identical with that for the posterior pdf (2.12) but it starts from the $\mathcal{L}(\Theta, d_0)$ identically equal to 1.

The normalization factor $\mathcal{I}(\cdot)$ in (2.13) is defined by the formula

$$\mathcal{I}(d^{1:t}) = \int \mathcal{L}(\Theta, d^{1:t}) f(\Theta)\, d\Theta \propto f(y_t|u_t, d^{1:t-1}).$$

With it, the predictive model of the system (2.11) can alternatively be expressed

$$f(y_t|u_t, d^{1:t-1}) = \frac{\mathcal{I}(d^{1:t})}{\mathcal{I}(d^{1:t-1})}.$$

Joint state and parameter estimation

Here, the problem of joint parameter and state estimation is formulated and solved. In this case, the internal quantity X_t is supposed to contain both states x_t and parameter Θ, see Section 2.3, i.e.,

$$\{X_t = (x_t, \Theta)\}_{t \in t^*},$$

that are mutually dependent. Now, the system behavior $\mathcal{Q}$ consist of the data $d^{1:t}$, states $x^{0:t}$ and parameter Θ. The system is described by the following pdfs.

1. The observation model (2.3) reads

$$\left\{f(y_t|u_t, d^{1:t-1}, x_t, \Theta)\right\}_{t \in t^*}$$

2. The time evolution model (2.4) is

$$\left\{f(x_t|u_t, d^{1:t-1}, x_{t-1}, \Theta)\right\}_{t \in t^*}, \Theta \text{ does not change}$$

3. Controller fulfilling NCC (2.5) has again the form

$$\left\{f(u_t|d^{1:t-1})\right\}_{t \in t^*}$$

4. According to (2.6), expressing a prior information on the states and parameters reads

$$f(x_1|u_1, x_0, \Theta) \equiv f(x_1|u_1, d_0, x_0, \Theta)$$
$$f(x_0, \Theta) \equiv f(x_0|d_0, \Theta) f(\Theta|d_0)$$

and

$$f(y_1|u_1, x_1, \Theta) \equiv f(y_1|u_1, d_0, x_1, \Theta)$$
$$f(u_1) \equiv f(u_1|d_0).$$

The evolution of the pdf $f(x_t, \Theta|u_t, d^{1:t-1})$ is called the *joint Bayesian parameter and state estimation*. It is described by the following recursion that starts from the prior pdf $f(x_0, \Theta)$

- *Data updating*

$$\begin{aligned} f(x_t, \Theta|d^{1:t}) &= \frac{f(y_t|u_t, d^{1:t-1}, x_t, \Theta) f(x_t, \Theta|u_t, d^{1:t-1})}{f(y_t|u_t, d^{1:t-1})} \\ &\propto f(y_t|u_t, d^{1:t-1}, x_t, \Theta) f(x_t, \Theta|u_t, d^{1:t-1}) \end{aligned}$$

  that incorporates information about the output y_t and the input u_t, and

- *Time updating*

$$f(x_{t+1}, \Theta|u_{t+1}, d^{1:t}) = \int_{x_t^*} f(x_{t+1}|u_{t+1}, d^{1:t}, x_t, \Theta) f(x_t, \Theta|d^{1:t})\,\mathrm{d}x_t$$

  that reflects the time evolution $x_t \to x_{t+1}$ while Θ stays unchanged.

2.5 MAP estimate

The method of maximum a posteriori (MAP) estimation provides a point estimate of an unobserved quantity X on the basis of observed data D and prior information about X, see e.g. [15],

$$\hat{X}_{MAP} = \arg\max_{X^*} \frac{f(D|X)f(X)}{\int_{X^*} f(D|X)f(X)d\,X} = \arg\max_{X^*} f(D|X)f(X) \quad (2.14)$$

The denominator of the posterior distribution does not depend on X and therefore plays no role in the optimization.

2.6 Taylor series

The Taylor series, see e.g. [16], represents the real function f as a sum of terms calculated from the values of its derivatives at a single point. The partial sums (the Taylor polynomials) of the series can be used as adequate approximations of the entire function.

In this work, the following expansion of logarithmic function is used.

$$\ln x = \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots, \text{ applicable for } 0 < x \leq 2.$$

We use only the first term, i.e.,

$$\ln x \approx \frac{x-1}{1}, \; 0 < x \leq 2. \quad (2.15)$$

2.7 Basics of the static optimization methods

The static optimization includes problems that can be described by a system of linear or non-linear equations and inequalities. The optimality criterion is given by a real function of a vector variable. The aim is to find a maximum or minimum of this function. Simultaneously, the constraints have to be fulfilled, that are given by the system of equalities and inequalities. Such task is called "mathematical programming".

A general problem of the mathematical programming, see e.g. [17], has the form

$$\begin{gathered}\text{Find a vector } X \text{ such that } J(X) \to \min \\ \text{while } \mathcal{A}X \leq \mathcal{B},\ \mathcal{A}_{eq}X = \mathcal{B}_{eq}, \\ g(X) \leq \mathbf{0},\ g_{eq}(X) = \mathbf{0},\ \underline{X} \leq X \leq \overline{X},\end{gathered}$$

where
$J(X)$ is known scalar function; it is called objective function
$g(X)$, $g_{eq}(X)$ are known real vector functions
$\mathcal{A}$, $\mathcal{A}_{eq}$ are known matrices of the appropriate size
$\mathcal{B}$, $\mathcal{B}_{eq}$, $\underline{X}$, $\overline{X}$ are known column vectors of the appropriate lengths

In Matlab, the nonlinear programming problems are solved by the function "fmincon". This function tries to find a constrained minimum of a scalar function of several variables starting at an initial estimate.

According to Matlab notation, the function "fmincon" can be used as follows, see [18],

$X = \mathbf{fmincon}\,(J(X), X_0, \mathcal{A}, \mathcal{B})$ starts at X_0 and attempts to find a minimum of the function J(X) subject to the linear inequalities $\mathcal{A}\,X <= \mathcal{B}$.

$X = \mathbf{fmincon}\,(J(X), X_0, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq})$ solves the problem above while additionally satisfying the equality constraints $\mathcal{A}_{eq}X = \mathcal{B}_{eq}$. Set $\mathcal{A} = \emptyset$ and $\mathcal{B} = \emptyset$ if no inequalities are employed.

$X = \mathbf{fmincon}\left(J(X), X_0, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq}, \underline{X}, \overline{X}\right)$ defines a set of lower and upper bounds on the design variables, X, so that the solution is always in the range $\underline{X} \leq X \leq \overline{X}$. Set $\mathcal{A}_{eq} = \emptyset$ and $\mathcal{B}_{eq} = \emptyset$ if no equalities are specified.

$X = \mathbf{fmincon}\left(J(X), X_0, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq}, \underline{X}, \overline{X}, \mathbf{nonlcon}\right)$ solves the problem above and simultaneously subjects the minimization to the nonlinear inequalities $g(X)$ and/or equalities $g_{eq}(X)$ defined in nonlcon; fmincon optimizes such that $g(X) \leq 0$ and $g_{eq}(X) = 0$. Set $\underline{X} = \emptyset$ and/or $\overline{X} = \emptyset$ if no bounds exist.

$X = \textbf{fmincon}\big(J(X), X_0, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq}), \underline{X}, \overline{X}, \textbf{nonlcon}, \textbf{optimset}\big)$ minimizes with the optimization options specified in the structure options. Use the function "optimset" to set these options. Set nonlcon $= \emptyset$ if there are no nonlinear inequality or equality constraints.

2.7.1 Linear programming

A substantial simplification arises when the objective function and the restrictive conditions are linear, see e.g. [17], [19]. Then, the vector X is searched for that minimizes the linear objective function, $J = \mathcal{C}'X$ and fulfills the system of inequalities $\mathcal{A}X \leq \mathcal{B}$, $\underline{X} \leq X \leq \overline{X}$ and equalities $\mathcal{A}_{eq}X = \mathcal{B}_{eq}$, i.e.,

$$\begin{aligned} &\text{Find a vector } X \text{ such that } J \equiv \mathcal{C}'X \to \min \\ &\text{while } \mathcal{A}X \leq \mathcal{B},\ \mathcal{A}_{eq}X = \mathcal{B}_{eq},\ \underline{X} \leq X \leq \overline{X}, \end{aligned} \tag{2.16}$$

where
$\mathcal{A}$, $\mathcal{A}_{eq}$ are known matrices of the appropriate size
$\mathcal{B}$, $\mathcal{B}_{eq}$, $\mathcal{C}$, $\underline{X}$, $\overline{X}$ are known vectors of the appropriate lengths

In this work, the LP problems are solved by the Matlab function "linprog" in experiments. According to Matlab notation, this function can be used as follows, see [18],

$X = \textbf{linprog}(\mathcal{C}, \mathcal{A}, \mathcal{B})$ solves min $\mathcal{C}'X$ such that $\mathcal{A}X \leq \mathcal{B}$.

$X = \textbf{linprog}(\mathcal{C}, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq})$ solves the same problem as above while additionally satisfying the equality constraints $\mathcal{A}_{eq}X = \mathcal{B}_{eq}$. Set $\mathcal{A} = \emptyset$ and $\mathcal{B} = \emptyset$ if no inequalities are employed.

$X = \textbf{linprog}(\mathcal{C}, \mathcal{A}, \mathcal{B}, \mathcal{A}_{eq}, \mathcal{B}_{eq}, \underline{X}, \overline{X})$ sets also a set of lower and upper bounds on the optimized variables, X, so that the solution is always in the range $\underline{X} \leq X \leq \overline{X}$. Set $\mathcal{A}_{eq} = \emptyset$ and $\mathcal{B}_{eq} = \emptyset$ if no equalities are specified.

Chapter 3

Linear uniform state-space model

Here, linear uniform state-space model (LU model) is introduced. Furthermore, the estimation of its parameters and state filtering are derived. The model has the form of the state space model that is known from Kalman filtering (KF) theory [1] but the assumed uniform distribution of innovations make it substantially different. The model generalizes the uniform state-space model treated by the author in [20]. Here, this model is extended by the considering the time-variant model matrices and the offsets.

The introduced LU model provides the following advantages: (i) it respects natural bounds on stochastic disturbances, (ii) it allows estimation of the innovation range, and (iii) it allows to respect "naturally" hard, physically given, prior bounds on model parameters and states.

3.1 Model description

The considered system is modelled by the following state (3.1) and observation (3.2) equations

$$x_t = {}^{c}A_t x_{t-1} + {}^{c}B_t u_t + {}^{c}F_t + {}^{x}e_t \tag{3.1}$$
$$y_t = {}^{c}C_t x_t + {}^{c}D_t u_t + {}^{c}G_t + {}^{y}e_t, \tag{3.2}$$

where

$t \in t^* \equiv \{1, 2, \dots, \mathring{t}\}$ labels discrete time;

x, u, y are state, input and output vectors respectively, see definition in Section 2.3;

${}^{c}A_t$, ${}^{c}B_t$, ${}^{c}F_t$, ${}^{c}C_t$, ${}^{c}D_t$, ${}^{c}G_t$ are model matrices of appropriate dimensions; they are sums of the form

$${}^{c}A_t = A_t + {}^{e}A, \ {}^{c}B_t = B_t + {}^{e}B, \ \text{etc.}, \tag{3.3}$$

where
A_t contains known, generally time-variant entries of ${}^{c}A_t$ and zeros,

${}^{e}A$ contains unknown time-invariant entries of ${}^{c}A_t$ and zeros (similarly for other system matrices); the unknown entries are collected into the "coefficient part" θ of the unknown parameter Θ (3.5)

$$\theta \equiv [\,{}^{e}A,\, {}^{e}B,\, {}^{e}F,\, {}^{e}C,\, {}^{e}D,\, {}^{e}G]\,;$$

${}^{x}e_t$, ${}^{y}e_t$ are the vectors of the state and output innovations respectively; they are assumed to be mutually conditionally independent (see (2.2)) and identically distributed.

The innovation are assumed to have uniform distributions

$$f\left({}^{x}e_t\right) = \mathcal{U}\left(0,\, {}^{x}r\right), \quad f\left({}^{y}e_t\right) = \mathcal{U}\left(0,\, {}^{y}r\right) \tag{3.4}$$

where
$\mathcal{U}(\mu,\, r)$ is uniform pdf on the box with the center μ and half-width of the support interval equal to r (understood entry-wise).

Equations (3.1) and (3.2) together with the assumptions (3.4) define the linear uniform state-space model (LU model).

We denote

$$\Theta \equiv [\theta,\, {}^{x}r,\, {}^{y}r] \tag{3.5}$$

and get, for $t \in t^* = \{1, 2, \ldots, \mathring{t}\}$

$$f\left(x_t | x_{t-1}, u_t, \Theta\right) = \mathcal{U}\left(\tilde{x}_t,\, {}^{x}r\right),\; \tilde{x}_t = {}^{c}A_t x_{t-1} + {}^{c}B_t u_t + {}^{c}F_t \tag{3.6}$$
$$f\left(y_t | x_t, u_t, \Theta\right) = \mathcal{U}\left(\tilde{y}_t,\, {}^{y}r\right),\; \tilde{y}_t = {}^{c}C_t x_t + {}^{c}D_t u_t + {}^{c}G_t \tag{3.7}$$

where (3.6) specifies the time evolution model (2.4) and (3.7) specifies the observation model (2.3) from Section 2.4.2.

Possible restrictions on the state values are in the form

$$\mathcal{S}2 = \{\underline{x} \le x_t \le \overline{x}\}, t \in t^*. \tag{3.8}$$

According to Section 2.4.2, we assume that generator of inputs $u^{1:\mathring{t}} \equiv [u_1', \ldots, u_{\mathring{t}}']'$ meets natural conditions of control (2.5), i.e., it uses explicitly neither state values nor unknown parameters.

Further, we suppose that the initial state x_0 and parameter Θ are uniformly distributed on the set $\mathcal{S}0$ defined by the inequalities

$$\mathcal{S}0 = \left\{\underline{x}_0 \le x_0 \le \overline{x}_0, \quad 0 < {}^{x}r \le {}^{x}\overline{r}, \quad 0 < {}^{y}r \le {}^{y}\overline{r}, \quad \underline{\theta} \le \theta \le \overline{\theta}\right\}. \tag{3.9}$$

They are assumed a priori mutually independent, hence

$$f\left(x_0,\, {}^{x}r,\, {}^{y}r, \theta\right) = f\left(x_0\right) f\left({}^{x}r\right) f\left({}^{y}r\right) f\left(\theta\right).$$

Then, the joint pdf of data $d^{1:\mathring{t}}$, $d_t = (y_t, u_t)$, the state trajectory $x^{0:\mathring{t}}$ and parameter Θ of the LU model is

$$f\left(d^{1:\mathring{t}}, x^{0:\mathring{t}}, \Theta\right) = f(x_0)\, f(\Theta) \prod_{t=1}^{\mathring{t}} f(y_t|\, u_t, x_t, \Theta)\, f(x_t|\, u_t, x_{t-1}, \Theta)\, f\left(u_t|\, d^{1:t-1}\right)$$

$$\propto \prod_{i=1}^{x^{\ell}} ({}^{x}r_i)^{-\mathring{t}} \prod_{i=1}^{y^{\ell}} ({}^{y}r_i)^{-\mathring{t}} \chi(\mathcal{S}) \tag{3.10}$$

where
x^{ℓ}, y^{ℓ} is the size of the state and output vector, respectively
$\chi(\mathcal{S})$ is the indicator of the support $\mathcal{S}$. The convex set $\mathcal{S}$ is given as follows

$$\mathcal{S} = \mathcal{S}0 \cap \mathcal{S}1 \cap \mathcal{S}2. \tag{3.11}$$

with $\mathcal{S}0$ given by (3.9) and $\mathcal{S}2$ by (3.8). The set $\mathcal{S}1$ is specified by inequalities

$$\begin{array}{rlcl} -\,{}^{x}r & \leq x_t - (A_t + {}^{e}A)x_{t-1} - (B_t + {}^{e}B)u_t - (F_t + {}^{e}F) & \leq & {}^{x}r \\ -\,{}^{y}r & \leq y_t - (C_t + {}^{e}C)x_t \quad - (D_t + {}^{e}D)u_t - (G_t + {}^{e}G) & \leq & {}^{y}r \end{array} \tag{3.12}$$

with $t \in t^*$.

Note that the inequalities (3.12) follow from the (3.1) and (3.2) with lower and upper noise values implied by (3.4).

The estimation tasks for the LU model will be solved in the following sections. For this purpose, the joint pdf (3.10) can be rewritten as follows

$$f\left(d^{1:\mathring{t}}, x^{0:\mathring{t}}, \Theta\right) = f(D, X) = f(D|\, X)\, f(X) \tag{3.13}$$

The estimated (unknown) internal quantities (see definition in Section 2.3) from (3.10) are collected into X. The remaining (known) quantities are collected into D. Here, D contains known elements from the set $\left\{d^{1:\mathring{t}}, x^{0:\mathring{t}}, \theta\right\}$ and X contains unknown elements from the set $\left\{x^{0:\mathring{t}}, \theta, {}^{x}r, {}^{y}r\right\}$. By construction, it holds $D \cap X = \emptyset$ and $D \cup X = \{d^{1:\mathring{t}}, x^{0:\mathring{t}}, \theta, {}^{x}r, {}^{y}r\}$.

3.2 General estimation problem

Using the notation (3.13), we can formulate the estimation problems in a common way and demonstrate its complexity. The posterior pdf $f(X|\, D)$ is searched for on the basis of the prior pdf $f(X)$ and of the pdf describing known quantities $f(D|\, X)$.

According to the Bayes rule (2.1), the required posterior pdf $f(X|\, D)$ is proportional to $f(D|\, X)\, f(X)$ on support $\mathcal{S}$ defined by (3.11). The number of vertices of the support is proportional to the number of data. This is a large number for realistic situations. Consequently, evaluation of moments of this pdf is computationally demanding. This is why we evaluate the maximum a posteriori probability (MAP) estimate $\hat{X}_{MAP}$ (2.14) of the unknown X. For the LU model, we get

$$\hat{X}_{MAP} = \arg\max_{X \in \mathcal{S}} \prod_{i=1}^{x^\ell} ({}^{x}r_i)^{-\mathring{t}} \prod_{j=1}^{y^\ell} ({}^{y}r_j)^{-\mathring{t}}. \tag{3.14}$$

The support $\mathcal{S}$ (3.11) defines the maximization domain. For the further computation, we use a negative logarithm of (3.14) divided by $\mathring{t}$. Then, the MAP estimation problem converts into

$$\hat{X}_{MAP} = \arg\min_{X \in \mathcal{S}} \left(\sum_{i=1}^{x^\ell} \ln({}^{x}r_i) + \sum_{j=1}^{y^\ell} \ln({}^{y}r_j) \right). \tag{3.15}$$

To obtain the MAP estimate of X, we have to find the minimum of (3.15) on the set $\mathcal{S}$ (3.11). We can approximate the particular entries of (3.15) with the help of Taylor series (2.15). Thus we obtain for $0 < {}^{x}r_i \leq 2,\ 0 < {}^{y}r_j \leq 2$

$$\ln({}^{x}r_i) \approx {}^{x}r_i - 1,\ i = 1, \ldots, x^\ell,\ \ \ln({}^{y}r_j) \approx {}^{y}r_j - 1,\ j = 1, \ldots, y^\ell.$$

Note that the condition on the magnitude of particular entries ${}^{x}r$ and ${}^{y}r$ can be always fulfilled by an appropriate data scaling. Using the above mentioned approximation, the optimization (3.15) simplifies to the following form

$$\hat{X}_{MAP} = \arg\min_{X \in \mathcal{S}} \left(\sum_{i=1}^{x^\ell} {}^{x}r_i + \sum_{j=1}^{y^\ell} {}^{y}r_j \right). \tag{3.16}$$

Then, if the inequalities describing the set $\mathcal{S}$ (3.11) are linear in the unknowns, the MAP estimate (3.16) can be found as the solution of the linear programming (LP), see Section 2.7.1,

$$\begin{aligned} &\textsf{Find a vector } X \textsf{ such that } J \equiv \mathcal{C}'X = \sum_{i=1}^{x^\ell} {}^{x}r_i + \sum_{j=1}^{y^\ell} {}^{y}r_j \rightarrow \min \\ &\textsf{while } \mathcal{A}X \leq \mathcal{B},\ \underline{X} \leq X \leq \overline{X}, \end{aligned} \tag{3.17}$$

where
$\mathcal{C}' \equiv [\mathbf{0}'_{(X^\ell - x^\ell - y^\ell, 1)}, \mathbf{1}'_{x^\ell + y^\ell}],\ \mathcal{C}^\ell = X^\ell$;
$\mathcal{A}$ and $\mathcal{B}$ are known matrix and vector, respectively; they result from the

inequalities describing the set $\mathcal{S}1$ (3.12);
$\underline{X}$, $\overline{X}$ are known vectors; they stem from $\mathcal{S}0$ (3.9) and $\mathcal{S}2$ (3.8).

Note that X is given by the type of the solved task. It contains always the half-widths ${}^{x}r$ and ${}^{y}r$. They are always placed at the end of X.

The above mentioned condition of the linearity is satisfied if either (i) parameters θ or (ii) states $x^{1:\mathring{t}}$ are known.

3.3 Off-line state and parameter estimation

In this section, two off-line estimation problems are solved: (i) one shot estimation of states $x^{0:\mathring{t}}$ and the innovation boundaries ${}^{x}r$, ${}^{y}r$ with the given θ and (ii) the one shot estimation of the parameters θ and ${}^{x}r$, ${}^{y}r$ with the known states $x^{0:\mathring{t}}$.

3.3.1 Estimation of the state and the noise bounds

In this case, we suppose that all entries of the model matrices (3.3) are known, i.e., ${}^{c}A_t \equiv A_t$, ${}^{c}B_t \equiv B_t$, etc. An unknown states $x^{0:\mathring{t}}$ and the noise bounds ${}^{x}r$, ${}^{y}r$ are estimated. Hence, the vector X in (3.17) is defined as follows

$$X = \left[\begin{array}{ccc} \left(x^{\mathring{t}:0}\right)' & {}^{x}r' & {}^{y}r' \end{array} \right]'. \tag{3.18}$$

The vector X has $X^{\ell} = (\mathring{t}+2)x^{\ell} + y^{\ell}$ entries, $\mathcal{C}' \equiv [\mathbf{0}'_{((\mathring{t}+1)x^{\ell},1)}, \mathbf{1}'_{(x^{\ell}+y^{\ell})}]$. $\mathcal{A}$ and $\mathcal{B}$ are given by the inequalities describing the set $\mathcal{S}_1$ (3.12) rearranged into the following form

$$\begin{array}{rcl} x_t - A_t x_{t-1} - {}^{x}r & \leq & B_t u_t + F_t \\ -x_t + A_t x_{t-1} - {}^{x}r & \leq & -B_t u_t - F_t \\ C_t x_t - {}^{y}r & \leq & y_t - D_t u_t - G_t \\ -C_t x_t - {}^{y}r & \leq & -y_t + D_t u_t + G_t \end{array}$$

for $t = \mathring{t}, \mathring{t}-1, \ldots, 1$.

With it, $\mathcal{A}$ and $\mathcal{B}$ have the following form

$$\mathcal{A} \ = \ \left[\begin{array}{cc} \mathcal{A}11 & \mathcal{A}12 \\ \mathcal{A}21 & \mathcal{A}22 \end{array} \right], \quad \mathcal{B} = \left[\begin{array}{c} \mathcal{B}1 \\ \mathcal{B}2 \end{array} \right],$$

with

$$\mathcal{A}11 = \left[\begin{array}{rrrcrrr} I_{(x^{\ell})} & -A_{\mathring{t}} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \ldots & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} \\ -I_{(x^{\ell})} & A_{\mathring{t}} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \ldots & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \ldots & \mathbf{0}_{(x^{\ell},x^{\ell})} & I_{(x^{\ell})} & -A_1 \\ \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \mathbf{0}_{(x^{\ell},x^{\ell})} & \ldots & \mathbf{0}_{(x^{\ell},x^{\ell})} & -I_{(x^{\ell})} & A_1 \end{array} \right],$$

$$\mathcal{A}12 = \mathbf{1}_{(2\mathring{t},1)} \otimes \left[-I_{(x^\ell)}\ \mathbf{0}_{(x^\ell,y^\ell)}\right],$$

$$\mathcal{A}21 = \begin{bmatrix} C_t & \dots & \mathbf{0}_{(}y^\ell, x^\ell) & \mathbf{0}_{(}y^\ell, x^\ell) & \mathbf{0}_{(}y^\ell, x^\ell) \\ -C_t & \dots & \mathbf{0}_{(}y^\ell, x^\ell) & \mathbf{0}_{(}y^\ell, x^\ell) & \mathbf{0}_{(}y^\ell, x^\ell) \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0}_{(}y^\ell, x^\ell) & \dots & \mathbf{0}_{(}y^\ell, x^\ell) & C_1 & \mathbf{0}_{(}y^\ell, x^\ell) \\ \mathbf{0}_{(}y^\ell, x^\ell) & \dots & \mathbf{0}_{(}y^\ell, x^\ell) & -C_1 & \mathbf{0}_{(}y^\ell, x^\ell) \end{bmatrix},$$

$$\mathcal{A}22 = \mathbf{1}_{(2\mathring{t},1)} \otimes \left[\mathbf{0}_{(y^\ell,x^\ell)}\ \ -I_{(y^\ell)}\right],$$

$$\mathcal{B}1 = \begin{bmatrix} B_t u_t + F_t \\ -B_t u_t - F_t \\ \vdots \\ B_1 u_1 + F_1 \\ -B_1 u_1 - F_1 \end{bmatrix}, \quad \mathcal{B}2 = \begin{bmatrix} y_t - D_t u_t - G_t \\ -y_t + D_t u_t + G_t \\ \vdots \\ y_1 - D_1 u_1 - G_1 u_1 \\ -y_1 + D_1 u_1 + G_1 u_1 \end{bmatrix},$$

where
$I_{(\alpha)}$ is the square identity matrix of the order α,
$\mathbf{0}_{(\alpha,\beta)}$ is the zero matrix of the indicated dimensions,
$\otimes$ denotes Kronecker product defined in Section 2.2.

The matrix $\mathcal{A}$ has $2\,\mathring{t}(x^\ell + y^\ell)$ rows and $X^\ell \equiv (\mathring{t}+2)x^\ell + y^\ell$ columns. The number of entries of the column vector $\mathcal{B}$ equals to the number of rows of $\mathcal{A}$, i.e., $\mathcal{B}^\ell \equiv 2\mathring{t}(x^\ell + y^\ell)$.

Under assumption that the LU model matrices are time-invariant, i.e., $A_t = A$, $B_t = B$, $F_t = F$, $C_t = C$, $D_t = D$, $G_t = G$, for $t = 1, \dots, \mathring{t}$, the construction of the $\mathcal{A}$, $\mathcal{B}$ can be simplified to the following form

$$\begin{aligned} \mathcal{A}11 &= \mathcal{R}_{x^\ell}(I_{(\mathring{t})} \otimes K \otimes I_{(x^\ell)}) - \mathcal{L}_{x^\ell}(I_{(\mathring{t})} \otimes K \otimes A), \\ \mathcal{A}12 &= -\mathbf{1}_{(2\mathring{t})} \otimes \mathcal{R}_{y^\ell}(I_{(x^\ell)}), \\ \mathcal{A}21 &= \mathcal{R}_{x^\ell}(I_{(\mathring{t})} \otimes K \otimes C), \\ \mathcal{A}22 &= -\mathbf{1}_{(2\mathring{t})} \otimes \mathcal{L}_{x^\ell}(I_{(y^\ell)}), \\ \mathcal{B}1 &= \left[I_{(\mathring{t})} \otimes K \otimes B\right] u^{\mathring{t}:1} + [\mathbf{1}_{(\mathring{t})} \otimes K \otimes F], \\ \mathcal{B}2 &= \left[I_{(\mathring{t})} \otimes K \otimes I_{(y^\ell)}\right] y^{\mathring{t}:1} - \left[I_{(\mathring{t})} \otimes K \otimes D\right] u^{\mathring{t}:1} - [\mathbf{1}_{(\mathring{t})} \otimes K \otimes G]. \end{aligned}$$

where
$\mathcal{R}_{col}(M)$ and $\mathcal{L}_{col}(M)$ are operators adding *col* zero columns from the right and left, respectively, see Section 2.2,
$K \equiv [1\ \ -1]'$.

Prior information on X reflecting $\mathcal{S}_0$ (3.9) and $\mathcal{S}_2$ (3.8) is assumed to be in the form $\underline{X} \leq X \leq \overline{X}$ with

$$\underline{X} = \begin{bmatrix} \mathbf{1}_{(2\mathring{t}x^{\ell},1)} \otimes \underline{x} \\ \underline{x}_0 \\ \mathbf{0}_{(x^{\ell},1)} \\ \mathbf{0}_{(y^{\ell},1)} \end{bmatrix}, \quad \overline{X} = \begin{bmatrix} \mathbf{1}_{(2\mathring{t}x^{\ell},1)} \otimes \overline{x} \\ \overline{x}_0 \\ {}^{x}\overline{r} \\ {}^{y}\overline{r} \end{bmatrix}, \tag{3.19}$$

where $\underline{x} \leq \overline{x}$, $\underline{x}_0 \leq \overline{x}_0$, ${}^{x}\overline{r} > 0$, ${}^{y}\overline{r} > 0$, entry-wise; these values are known and optional and specify our prior information.

Note that for $\mathring{t}$ too large, the estimation of $x_{\mathring{t}}, \ldots, x_{\mathring{t}-\partial}$ for some reasonably chosen fixed memory ∂ is to be considered, see Section 3.4.2.

3.3.2 Estimation of the parameters and the noise bounds

Here, the case of known state trajectory $x^{0:\mathring{t}}$ and the unknown time-invariant model parameters $\theta = ({}^{e}\!A, {}^{e}\!B, {}^{e}\!F, {}^{e}\!C, {}^{e}\!D, {}^{e}\!G)$ and noise bounds ${}^{x}r$, ${}^{y}r$ is considered. This case arises in situations with directly measurable state. Moreover, the result is needed for the joint swapping-based estimation of state and parameters, which is addressed in Section 3.4.4. Again, the MAP estimate is obtained by solving of LP problem. Now, the vector X in the LP (3.17) has the form:

$$X \equiv [\mathrm{col}({}^{e}\!A)', \mathrm{col}({}^{e}\!B)', \mathrm{col}({}^{e}\!F)', \mathrm{col}({}^{e}\!C)', \mathrm{col}({}^{e}\!D)', \mathrm{col}({}^{e}\!G)', {}^{x}r', {}^{y}r']'. \tag{3.20}$$

where $\mathrm{col}(M)$ stacks the non-zero rows of the matrix M into a column vector.

Note that the vector $\mathcal{C}$ in LP (3.17) contains $x^{\ell} + y^{\ell}$ units at the end and zeros otherwise. If no entry of the system matrices is known, then $\mathcal{C}$ reaches its maximal size

$$\mathcal{C} \equiv [\mathbf{0}'_{(x^{\ell}x^{\ell}+x^{\ell}u^{\ell}+x^{\ell}+y^{\ell}x^{\ell}+y^{\ell}u^{\ell}+y^{\ell},1)}, \mathbf{1}'_{(x^{\ell}+y^{\ell},1)}]'.$$

$\mathcal{A}$ and $\mathcal{B}$ result from inequalities describing the set $\mathcal{S}_1$ (3.12) that are rearranged into the following form with $t \in t^* = \{\mathring{t}, \mathring{t}-1, \ldots, 1\}$

$$\begin{array}{rcl} {}^{e}\!Ax_{t-1} + {}^{e}\!Bu_t + {}^{e}\!F - {}^{x}r & \leq & x_t - A_t x_{t-1} - B_t u_t - F_t \\ -{}^{e}\!Ax_{t-1} - {}^{e}\!Bu_t - {}^{e}\!F - {}^{x}r & \leq & -x_t + A_t x_{t-1} + B_t u_t + F_t \\ {}^{e}\!Cx_t + {}^{e}\!Du_t + {}^{e}\!G - {}^{y}r & \leq & y_t - C_t x_t - D_t u_t - G_t \\ -{}^{e}\!Cx_t - {}^{e}\!Du_t - {}^{e}\!G - {}^{y}r & \leq & -y_t + C_t x_t + D_t u_t + G_t \end{array}$$

The matrix $\mathcal{A}$ and vector $\mathcal{B}$ in LP (3.17) are then

$$\mathcal{A} \equiv \begin{bmatrix} \mathcal{A}11 & \mathcal{A}12 & \mathcal{A}13 \\ \mathcal{A}21 & \mathcal{A}22 & \mathcal{A}23 \end{bmatrix}, \ \mathcal{B} = \begin{bmatrix} \mathcal{B}1 \\ \mathcal{B}2 \end{bmatrix}$$

with

$$\mathcal{A}11 = \begin{bmatrix} SEL_{{}^eA,x_{\mathring{t}-1}} & SEL_{{}^eB,u_{\mathring{t}}} & SEL_{{}^eF,1} \\ -SEL_{{}^eA,x_{\mathring{t}-1}} & -SEL_{{}^eB,u_{\mathring{t}}} & -SEL_{{}^eF,1} \\ \vdots & \vdots & \vdots \\ SEL_{{}^eA,x_0} & SEL_{{}^eB,u_1} & SEL_{{}^eF,1} \\ -SEL_{{}^eA,x_0} & -SEL_{{}^eB,u_1} & -SEL_{{}^eF,1} \end{bmatrix}.$$

where

$$SEL_{M;v_t} = \begin{bmatrix} sel_M(v_t,1) & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & sel_M(v_t,v^\ell) \end{bmatrix}, \tag{3.21}$$

$sel_M(v_t, i)$ creates a reduced vector (scalar) $\tilde{v}_t$ from $v_t \in \{x_t, u_t, 1\}$, $t \in t^*$; it selects entries of v_t with index corresponding to the non-zero columns on the i-th row of the matrix (vector) M, $M \in \{{}^eA,\ {}^eB,\ {}^eF,\ {}^eC,\ {}^eD,\ {}^eG\}$; note that if all entries on i-th row are equal to zero, then $\tilde{v}_t$ is an "empty" vector.

$$\mathcal{A}12 = \mathbf{0}_{2\mathring{t}x^\ell,n}, \quad \text{the upper bound on } n \text{ is } y^\ell(x^\ell + u^\ell + 1)$$

$$\mathcal{A}13 \equiv \mathbf{1}_{(2\mathring{t},1)} \otimes \left[-I_{(x^\ell)}\ \mathbf{0}_{(x^\ell,y^\ell)}\right]$$

where $\otimes$ means Kronecker product defined in Section 2.2.

$$\mathcal{A}21 = \mathbf{0}_{2\mathring{t}y^\ell,m}, \ \text{the upper bound on } m \text{ is } x^\ell(x^\ell + u^\ell + 1)$$

$$\mathcal{A}22 = \begin{bmatrix} SEL_{{}^eC,x_{\mathring{t}}} & SEL_{{}^eD,u_{\mathring{t}}} & SEL_{{}^eG,1} \\ -SEL_{{}^eC,x_{\mathring{t}}} & -SEL_{{}^eD,u_{\mathring{t}}} & -SEL_{{}^eG,1} \\ \vdots & \vdots & \vdots \\ SEL_{{}^eC,x_1} & SEL_{{}^eD,u_1} & SEL_{{}^eG,1} \\ -SEL_{{}^eC,x_1} & -SEL_{{}^eD,u_1} & -SEL_{{}^eG,1} \end{bmatrix}$$

$$\mathcal{A}23 = \mathbf{1}_{(2\mathring{t},1)} \otimes \left[\mathbf{0}_{(y^\ell,x^\ell)}\ \ -I_{(y^\ell)}\right]$$

$$\mathcal{B}1 \equiv \begin{bmatrix} x_{\mathring{t}} - A_{\mathring{t}}x_{\mathring{t}-1} - B_{\mathring{t}}u_{\mathring{t}} - F_{\mathring{t}} \\ -x_{\mathring{t}} + A_{\mathring{t}}x_{\mathring{t}-1} + B_{\mathring{t}}u_{\mathring{t}} + F_{\mathring{t}} \\ \vdots \\ x_1 - A_1x_0 - B_1u_1 - F_1 \\ -x_1 + A_1x_0 + B_1u_1 + F_1 \end{bmatrix}, \quad \mathcal{B}2 \equiv \begin{bmatrix} y_{\mathring{t}} - C_tx_t - D_tu_t - G_t \\ -y_{\mathring{t}} + C_tx_t + D_tu_t + G_t \\ \vdots \\ y_1 - C_tx_1 - D_1u_1 - G_1 \\ -y_1 + C_tx_1 + D_1u_1 + G_1 \end{bmatrix}.$$

A prior information on X reflecting $\mathcal{S}_0$ (3.9) is assumed to be in the form $\underline{X} \leq X \leq \overline{X}$ with

$$\underline{X} = \left[\mathrm{col}({}^e\underline{A})', \mathrm{col}({}^e\underline{B})', \mathrm{col}({}^e\underline{F})', \mathrm{col}({}^e\underline{C})', \mathrm{col}({}^e\underline{D})', \mathrm{col}({}^e\underline{G})', \mathbf{0}_{(\mathring{x}^{\ell},1)}, \mathbf{0}_{(\mathring{y}^{\ell},1)}\right]', \tag{3.22}$$

$$\overline{X} = \left[\mathrm{col}({}^e\overline{A})', \mathrm{col}({}^e\overline{B})', \mathrm{col}({}^e\overline{F})', \mathrm{col}({}^e\overline{C})', \mathrm{col}({}^e\overline{D})', \mathrm{col}({}^e\overline{G})', {}^x\overline{r}, {}^y\overline{r}\right]'.$$

3.4 On-line state and parameter estimation

The real-time (on-line) estimation provides the state and/or parameter estimates in each time step. Standard Bayesian learning with a fixed lag $\partial \geq 0$ works with the data $d^{t-\partial:t}$ and states $x^{t-\partial:t}$. The superfluous state $x_{t-\partial-1}$ and data item $d_{t-\partial-1}$ are integrated out from the posterior pdf in every time step t. With increasing t, this operation soon becomes intractable because of increasing complexity of the support of the posterior pdf. Therefore, an approximation is needed.

The on-line Bayesian learning task is outlined and its approximation is proposed in Subsection 3.4.1. With the help of proposed approximation, the tasks of state and parameters estimation are solved in Subsections 3.4.2 and 3.4.3, respectively.

Then, the assumptions of Section 3.3, i.e., knowledge of either the state, or the parameters θ are relaxed and on-line joint parameter and state estimation is performed. This relaxation violates the assumptions of LP on linearity of the constraints. Two promising approaches are proposed to solve this problem in the Subsection 3.4.4.

3.4.1 Approximation for the on-line Bayesian learning

The Bayesian on-line estimation with restricted memory evolves the join pdf

$$f\left(d^{t-\partial:t}, x^{t-\partial:t}, \Theta\right), t \in t^*,$$

where integer $0 \leq \partial < \mathring{t}$ means memory length, i.e., the length of the sliding window; within this window the estimation is performed. Using a newest data d_t and state x_t, we can write (see Section 2.4.2)

$$f\left(d^{t-\partial:t}, x^{t-\partial:t}, \Theta\right) = \int f\left(y_t | x_t, u_t, \Theta\right) f\left(x_t | x_{t-1}, u_t, \Theta\right) f\left(u_t \middle| d^{1:t-1}\right)$$

$$\times f\left(d^{t-\partial-1:t-1}, x^{t-\partial-1:t-1}, \Theta\right) \mathrm{d}d_{t-\partial-1}\mathrm{d}x_{t-\partial-1}$$

The recursion starts in the time $t = \partial + 1$ with

$$f\left(d^{1:\partial}, x^{0:\partial}, \Theta\right) \equiv f\left(d^{0:\partial}, x^{0:\partial}, \Theta\right).$$

We suppose the pdf describing input generator as follows

$$f\left(u_t \middle| d^{1:t-1}\right) \equiv f\left(u_t \middle| d^{t-\partial:t-1}\right).$$

Recursive computation of the posterior pdf ($\partial \geq 2$)

1. For $t \leq \partial$:
   The computation is identical to the off-line case treated in Section 3.3.

2. For $t = \partial + 1$:

$$f\left(d^{1:\partial+1}, x^{1:\partial+1}, \Theta\right) = \int f\left(d^{1:\partial+1}, x^{0:\partial+1}, \Theta\right) \mathrm{d}\, x_0 \tag{3.23}$$

$$= \prod_{\tau=1}^{\partial+1} f\left(y_\tau | x_\tau, u_\tau, \Theta\right) \prod_{\tau=2}^{\partial+1} f\left(x_\tau | x_{\tau-1}, u_\tau, \Theta\right) \prod_{\tau=1}^{\partial+1} f\left(u_\tau \middle| d^{1:\tau-1}\right) f\left(\Theta\right)$$

$$\times \underbrace{\int f\left(x_1 | x_0, u_1, \Theta\right) f\left(x_0\right) \mathrm{d}\, x_0}_{\mathcal{I}(x_1, u_1, \Theta)}$$

3. For $t = \partial + 2$:

$$f\left(d^{2:\partial+2}, x^{2:\partial+2}, \Theta\right) \tag{3.24}$$

$$= \prod_{\tau=2}^{\partial+2} f\left(y_\tau | x_\tau, u_\tau, \Theta\right) \prod_{\tau=3}^{\partial+2} f\left(x_\tau | x_{\tau-1}, u_\tau, \Theta\right) f\left(u_{\partial+2} \middle| d^{2:\partial+1}\right) f\left(\Theta\right)$$

$$\times \underbrace{\int f\left(y_1 | x_1, u_1, \Theta\right) f\left(x_2 | x_1, u_2, \Theta\right) \prod_{\tau=1}^{\partial+1} f\left(u_\tau \middle| d^{1:\tau-1}\right) \mathcal{I}(x_1, u_1, \Theta) \mathrm{d}\, d_1 \, \mathrm{d}\, x_1}_{\mathcal{I}(x_2, u_{\partial+1}, d^{2:\partial}, \Theta)}$$

   with $f\left(u_1 | d^{1:0}\right) \equiv f\left(u_1\right)$

4. Generally for $\partial + 2 \leq t \leq \mathring{t}$:

$$f\left(d^{t-\partial:t}, x^{t-\partial:t}, \Theta\right) = \prod_{\tau=t-\partial}^{t} f\left(y_\tau | x_\tau, u_\tau, \Theta\right) \prod_{\tau=t-\partial+1}^{t} f\left(x_\tau | x_{\tau-1}, u_\tau, \Theta\right)$$

$$\times f\left(u_t \middle| d^{t-\partial:t}\right) f\left(\Theta\right) \mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta) \tag{3.25}$$

with

$$\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta) \tag{3.26}$$

$$= \int f\left(y_{t-\partial-1}|\, x_{t-\partial-1}, u_{t-\partial-1}, \Theta\right) f\left(x_{t-\partial}|\, x_{t-\partial-1}, u_{t-\partial}, \Theta\right)$$

$$\times f\left(u_{t-\partial}\middle|\, d^{t-\partial-1:t-2}\right)$$

$$\times \mathcal{I}(x_{t-\partial-1}, u_{t-2}, d^{t-\partial-1:t-3}, \Theta)\mathrm{d}\, d_{t-\partial-1}\, \mathrm{d}\, x_{t-\partial-1}.$$

The first term in the recursion, i.e., $\mathcal{I}(x_2, u_{\partial+1}, d^{2:\partial}, \Theta)$, is computed according to (3.24).

Note that for $\partial = 1$, the term $\mathcal{I}(x_1, u_1, \Theta)$ in (3.23) stays unchanged, while the term $\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$ (3.26) is simplified to $\mathcal{I}(x_{t-1}, u_{t-1}, \Theta)$ and its recursive update is as follows

$$\mathcal{I}(x_{t-1}, u_{t-1}, \Theta) = \tag{3.27}$$

$$\int f\left(y_{t-2}|\, x_{t-2}, u_{t-2}, \Theta\right) f\left(x_{t-1}|\, x_{t-2}, u_{t-1}, \Theta\right) f\left(u_{t-1}|\, d_{t-2}\right)$$

$$\times \mathcal{I}(x_{t-2}, u_{t-2}, \Theta)\mathrm{d}\, d_{t-2}\, \mathrm{d}\, x_{t-2}.$$

Illustrative example: Evolution of the factor $\mathcal{I}$

We consider simple system without input, with scalar state and output described by the LU model (3.1), (3.2)

$x_t \sim \mathcal{U}\left(a x_{t-1},\, {}^{x}r\right),\ a > 0;\ y_t \sim \mathcal{U}\left(c x_t,\, {}^{y}r\right),\ c > 0;\ x_0 \sim \mathcal{U}\left(\frac{\underline{x}_0 + \overline{x}_0}{2}, \frac{\overline{x}_0 - \underline{x}_0}{2}\right)$

$\partial = 1$; $d_t \equiv y_t$; $\Theta = (a, c, {}^{x}r, {}^{y}r)$, $\underline{x}_0 < \overline{x}_0$ given.

The first step of the integration corresponding to (3.23) is

$$f\left(x^{1:2}, y^{1:2}, \Theta\right)$$

$$= f\left(y_2|\, x_2, \Theta\right) f\left(y_1|\, x_1, \Theta\right) f\left(x_2|\, x_1, \Theta\right) f\left(\Theta\right) \int f(x_1|x_0) f(x_0) \mathrm{d}x_0$$

$$= \frac{1}{2\, {}^{x}r} \frac{1}{(2\, {}^{y}r)^2} \chi(x_2^*) \chi(y_2^*) \chi(y_1^*) f\left(\Theta\right)$$

$$\times \int_{\underline{x}_0}^{\overline{x}_0} \frac{1}{2\,{}^x r}\frac{1}{\overline{x}_0-\underline{x}_0}\chi(|x_1-ax_0|\le {}^x r)\chi(\underline{x}_0\le x_0\le \overline{x}_0)\mathrm{d}x_0$$
$$= \frac{1}{(2\,{}^x r)}\frac{1}{(2\,{}^y r)^2}\chi(x_2^*)\chi(y_2^*)\chi(y_1^*)f\left(\Theta\right)$$
$$\times \underbrace{\frac{1}{(2\,{}^x r)}\frac{1}{\overline{x}_0-\underline{x}_0}\left[\min\left(\frac{x_1+{}^x r}{a},\overline{x}_0\right)-\max\left(\frac{x_1-{}^x r}{a},\underline{x}_0\right)\right]}_{\mathcal{I}(x_1,\Theta)}$$

The term $\mathcal{I}(x_1,\Theta)$ represents a non-uniform distribution. Figure 3.1 shows the resulting distributions for x_1 with ${}^x r = 0.1$, $x_0 \in (0.8,\,1.2)$ and various values of a.

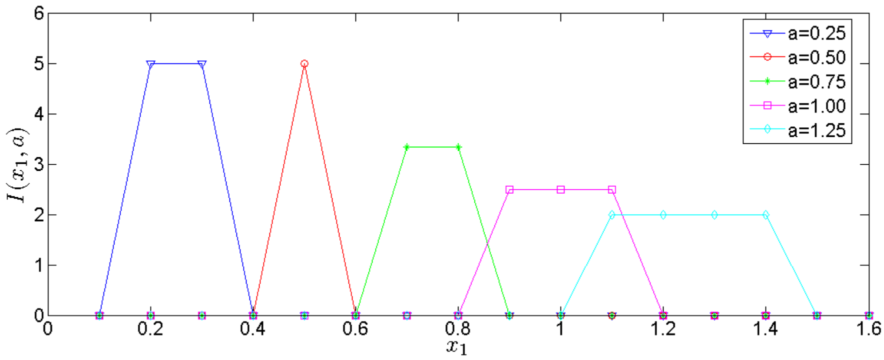

Figure 3.1: Resulting distribution $I(x_1,a)$ for various values of parameter a

In the next step, the pdf $f\left(y^{2:3},x^{2:3},\Theta\right)$ is to be computed using (3.27)

$$f\left(y^{2:3},x^{2:3},\Theta\right) = f\left(y_3|x_3,\Theta\right)f\left(y_2|x_2,\Theta\right)f\left(x_3|x_2,\Theta\right)f\left(\Theta\right)\mathcal{I}(x_2,\Theta)$$

with

$$\mathcal{I}(x_2,\Theta) = \int \frac{1}{2\,{}^x r}\frac{1}{2\,{}^y r}\chi(x_2^*)\chi(y_1^*)\mathcal{I}(x_1,\Theta)\mathrm{d}y_1\mathrm{d}x_1$$
$$= \frac{1}{(2\,{}^x r)^2}\frac{1}{\overline{x}_0-\underline{x}_0}\int \chi\left(\frac{x_2-{}^x r}{a}\le x_1\le \frac{x_2+{}^x r}{a}\right)$$
$$\times\left[\min\left(\frac{x_1+{}^x r}{a},\overline{x}_0\right)-\max\left(\frac{x_1-{}^x r}{a},\underline{x}_0\right)\right]\mathrm{d}y_1\mathrm{d}x_1$$

The term $\mathcal{I}(x_2,\Theta)$ contains quadratic expressions. Generally, the term $\mathcal{I}(x_t,\Theta)$, $t>0$ is described by the power function containing expressions up to the power t, $\mathcal{I}(x_t,\Theta)=\int f\left(x_t|x_{t-1},\Theta\right)\mathcal{I}(x_{t-1},\Theta)\mathrm{d}\,x_{t-1}$

This simple example demonstrates the complexity of the on-line estimation task with restricted memory and the necessity of the appropriate approximation in the applied on-line estimation.

Approximation methods

Generally, the terms $\mathcal{I}(x_1, u_1, \Theta)$ from (3.23) and $\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$, $t \in t^*$ (3.26) are to be approximated. We aim at approximating of these terms by the uniform pdfs as follows.

The factor $\mathcal{I}$ in (3.26) is to be approximated by a pdf $\tilde{f}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$, $t \in t^*$ so that it holds for measured data $d^{t-\partial:t-2}$, u_{t-1},

$$\frac{\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)}{\tilde{f}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)} \approx 1, \ x \in x^*, \Theta \in \Theta^*.$$

Similarly, $\mathcal{I}(x_1, u_1, \Theta)$ is approximated by the pdf $\tilde{f}(x_1, u_1, \Theta)$.

Two proposed methods of the approximation for the on-line estimation of the LU model are described below.

1. „Cut off" the superfluous old states

for $t = \partial + 1$ (it corresponds to the step 2 on the page 28):
the pdf $\mathcal{I}(x_1, u_1, \Theta)$ from (3.23) is replaced by $f\left(x_1 | x_0, u_1, \Theta\right)_{x_0 = \hat{x_0}}$, where $\hat{x_0}$ is the point estimate of the state x_0

for $\partial + 2 \leq t \leq \mathring{t}$ (it correspond to the steps 3 and 4 on the page 28):
the term $\mathcal{I}(x_2, u_{\partial+1}, d^{2:\partial}, \Theta)$ from (3.24) and $\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$ (3.26) from (3.25) are replaced by the
$f\left(x_{t-\partial} | x_{t-\partial-1}, u_{t-\partial}, \Theta\right)_{x_{t-\partial-1} = \hat{x}_{t-\partial-1}} \prod_{\tau=t-\partial}^{t-1} f\left(u_\tau \middle| d^{t-\partial:\tau-1}\right)$,
where $\hat{x}_{t-\partial-1}$ is the point estimate of $x_{t-\partial-1}$ from the previous estimation step and it holds

$$f\left(d^{t-\partial:t}, x^{t-\partial:t}, \Theta\right)$$

$$\approx \prod_{\tau=t-\partial}^{t} f\left(y_\tau | x_\tau, u_\tau, \Theta\right) \prod_{\tau=t-\partial}^{t} f\left(x_\tau | x_{\tau-1}, u_\tau, \Theta\right)_{x_{t-\partial-1} = \hat{x}_{t-\partial-1}}$$

$$\times \prod_{\tau=t-\partial}^{t} f\left(u_\tau \middle| d^{t-\partial:\tau-1}\right) f\left(\Theta\right)$$

where $f\left(u_{t-\partial} \middle| d^{t-\partial:t-\partial-1}\right) \equiv f\left(u_{t-\partial}\right)$

2. Substitution of the non-uniform pdf by the uniform one
The principle is that the non-uniform pdf $\mathcal{I}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$ is replaced with the uniform pdf $\tilde{f}(x_{t-\partial}, u_{t-1}, d^{t-\partial:t-2}, \Theta)$, which has the same support, see illustrative Figure 3.2 for the one-dimensional x with pdf $f(x)$.

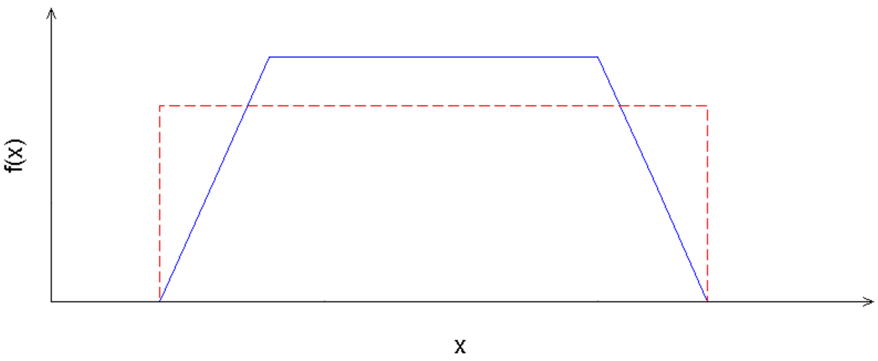

Figure 3.2: Approximation of the non-uniform distribution $\mathcal{I}$ (full line) by the uniform one $\tilde{f}$ (dashed line)

Approximate on-line MAP estimation

In the following sections, an approximate on-line state and/or parameter estimation is performed. The approximation by "cutting-off" method is used. The estimated quantities are concentrated in the vector X_t. The estimation run on the sliding window of the length ∂.

With the proposed "cut-off" approximation, the MAP estimation problem converts into

$$\hat{X}_{MAP} = \arg\min_{X_t \in \mathcal{S}_t} \left(\sum_{i=1}^{x\ell} {}^{x}r_i + \sum_{j=1}^{y\ell} {}^{y}r_j \right). \tag{3.28}$$

where $\mathcal{S}_t$ outgoing from the set $\mathcal{S}$ (3.11) is as follows

$$\mathcal{S}_t = \mathcal{S}0_t \cap \mathcal{S}1_t \cap \mathcal{S}2_t \tag{3.29}$$

where

$$\mathcal{S}0_t = \left\{ x_{t-\partial-1} = \hat{x}_{t-\partial-1}, \quad 0 < {}^{x}r \leq {}^{x}\overline{r}, \quad 0 < {}^{y}r \leq {}^{y}\overline{r}, \quad \underline{\theta} \leq \theta \leq \overline{\theta} \right\}, \tag{3.30}$$

$\hat{x}_{t-\partial-1}$ is the estimate of $x_{t-\partial-1}$ from the previous step

$$\tilde{\mathcal{S}}1_t = \{ -{}^{x}r \leq x_\tau - {}^{c}A_\tau x_{\tau-1} - {}^{c}B_\tau u_\tau - {}^{c}F_\tau \leq {}^{x}r, \tag{3.31}$$

$$-{}^{y}r \leq y_\tau - {}^{c}C_\tau x_\tau \quad - {}^{c}D_\tau u_\tau - {}^{c}G_\tau \leq {}^{y}r \}$$

$$\mathcal{S}2_t = \{ \underline{x} \leq x_\tau \leq \overline{x} \}, \tag{3.32}$$

with $\tau \in \{t-\partial, \ldots, t\}$, $t \in \{\partial+1, \ldots, \mathring{t}\}$.

In the same way as in the off-line case, see page 22, the MAP estimate (3.28) can be found as the solution of the LP under condition that the inequalities describing the set $\mathcal{S}_t$ (3.29) are linear in the unknowns. Then, the on-line LP task has the following form

$$\begin{aligned}&\text{Find a vector } X_t,\ t \in t^* \text{ such that } J \equiv \mathcal{C}' X_t = \sum_{i=1}^{x^\ell} {}^x r_i + \sum_{j=1}^{y^\ell} {}^y r_j \rightarrow \min \\ &\text{while } \mathcal{A}_t X_t \leq \mathcal{B}_t,\ \underline{X_t} \leq X_t \leq \overline{X_t},\end{aligned} \tag{3.33}$$

where
$\mathcal{C}' \equiv [\mathbf{0}'_{(X^\ell - x^\ell - y^\ell, 1)}, \mathbf{1}'_{x^\ell + y^\ell}]$, $\mathcal{C}^\ell = X^\ell$;
$\mathcal{A}_t$ and $\mathcal{B}_t$ are known matrix and vector, respectively; they result from the inequalities describing the set $\mathcal{S}1_t$ (3.31);
$\underline{X}_t$, $\overline{X}_t$ are known vectors; they stem from $\mathcal{S}0_t$ (3.30) and $\mathcal{S}2_t$ (3.32).

Note that X_t contains always the half-widths ${}^x r$ and ${}^y r$ that are placed at the end of the vector.

The above mentioned condition of the linearity is satisfied if either (i) parameters θ or (ii) states $x^{t-\partial:t}$ are known.

3.4.2 Estimation of the state and the noise bounds

In the case of "cutt-off" approximation, the on-line estimation of the states and the noise bounds is similar to the off-line estimation that is described in the Section 3.3.1. The differences resulting from the use of the sliding window ∂ are as follows. The vector of an estimated quantities, c.f. (3.18), takes the form

$$X_t = \left[\begin{array}{ccc} \left(x^{t:t-\partial}\right)' & {}^x r' & {}^y r' \end{array} \right]'$$

where $t \in t^* = \{\partial + 1, \ldots, \mathring{t}\}$.

The prior information described by (3.19) in the off-line version, is modified so that the original condition $\underline{x}_0 \leq x_0 \leq \overline{x}_0$ changes to $x_{t-\partial-1} = \hat{x}_{t-\partial-1}$, $t \in t^* = \{\partial + 1, \ldots \mathring{t}\}$, where $\hat{x}_{t-\partial-1}$ is the estimate of $x_{t-\partial-1}$ from the previous step. Using the sets $\mathcal{S}0_t$ (3.30) and $\mathcal{S}2_t$ (3.32), we obtain $\underline{X}_t \leq X_t \leq \overline{X}_t$ with

$$\underline{X}_t = \begin{bmatrix} \mathbf{1}_{(2(\partial+1)x^\ell,1)} \otimes \underline{x} \\ \hat{x}_{t-\partial-1} \\ \mathbf{0}_{(x^\ell,1)} \\ \mathbf{0}_{(y^\ell,1)} \end{bmatrix}, \quad \overline{X}_t = \begin{bmatrix} \mathbf{1}_{(2(\partial+1)x^\ell,1)} \otimes \overline{x} \\ \hat{x}_{t-\partial-1} \\ {}^x\overline{r} \\ {}^y\overline{r} \end{bmatrix}, \tag{3.34}$$

where vectors $\underline{x} \leq \overline{x}$, ${}^x\overline{r} > 0$, ${}^y\overline{r} > 0$, entry-wise; these values are known and optional and specify our prior information.

The matrix $\mathcal{A}_t$ and vector $\mathcal{B}_t$ for LP (3.17) imply from the set $\mathcal{S}1_t$ (3.31) and have the following form

$$\mathcal{A}_t = \begin{bmatrix} \mathcal{A}11_t & \mathcal{A}12_t \\ \mathcal{A}21_t & \mathcal{A}22_t \end{bmatrix}, \quad \mathcal{B}_t = \begin{bmatrix} \mathcal{B}1_t \\ \mathcal{B}2_t \end{bmatrix},$$

with

$$\mathcal{A}11_t = \begin{bmatrix} I_{(x^\ell,x^\ell)} & -A_t & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\ -I_{(x^\ell,x^\ell)} & A_t & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\ \mathbf{0}_{(x^\ell,x^\ell)} & I_{(x^\ell,x^\ell)} & -A_{t-1} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\ \mathbf{0}_{(x^\ell,x^\ell)} & -I_{(x^\ell,x^\ell)} & A_{t-1} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{(x^\ell,x^\ell)} & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \dots & I_{(x^\ell,x^\ell)} \\ \mathbf{0}_{(x^\ell,x^\ell)} & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \dots & -I_{(x^\ell,x^\ell)} \end{bmatrix},$$

$$\mathcal{A}12_t = \mathbf{1}_{(2(\partial+1),1)} \otimes \left[-I_{(x^\ell)} \ \mathbf{0}_{(x^\ell,y^\ell)}\right],$$

$$\mathcal{A}21_t = \begin{bmatrix} C_t & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & \mathbf{0}_{(y^\ell,x^\ell)} \\ -C_t & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & \mathbf{0}_{(y^\ell,x^\ell)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{(y^\ell,x^\ell)} & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & C_{t-\partial} \\ \mathbf{0}_{(y^\ell,x^\ell)} & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & -C_{t-\partial} \end{bmatrix},$$

$$\mathcal{A}22_t = \mathbf{1}_{(2(\partial+1),1)} \otimes \left[\mathbf{0}_{(y^\ell,x^\ell)} \ -I_{(y^\ell)}\right],$$

$$\mathcal{B}1_t = \begin{bmatrix} B_t u_t + F_t \\ -B_t u_t - F_t \\ \vdots \\ B_{t-\partial+1}u_{t-\partial+1} + F_{t-\partial+1} \\ -B_{t-\partial+1}u_{t-\partial+1} - F_{t-\partial+1} \\ A_{t-\partial}\hat{x}_{t-\partial-1} + B_{t-\partial}u_{t-\partial} + F_{t-\partial} \\ -A_{t-\partial}\hat{x}_{t-\partial-1} - B_{t-\partial}u_{t-\partial} - F_{t-\partial} \end{bmatrix},$$

$$\mathcal{B}2_t = \begin{bmatrix} y_t - D_t u_t - G_t \\ -y_t + D_t u_t + G_t \\ \vdots \\ y_{t-\partial} - D_{t-\partial}u_{t-\partial} - G_{t-\partial} \\ -y_{t-\partial} + D_{t-\partial}u_{t-\partial} + G_{t-\partial} \end{bmatrix}.$$

Comparing with the off-line version described in Section 3.3.1, we can see following differences:

- the number of the rows of the matrices decreases according to the memory length ∂
- the number of the columns in the matrices $\mathcal{A}11$, $\mathcal{A}21$ decreases because of the substitution of the oldest state by its estimate
- the term $A_{t-\partial}\hat{x}_{t-\partial-1}$ appears in two last rows of $\mathcal{B}1$ because of the substitution of the oldest state by its estimate

3.4.3 Estimation of the parameters and the noise bounds

Here, the vector of estimated quantities X_t, $t \in t^* = \{\partial+1,\ldots,\mathring{t}\}$, has the identical form with X (3.20) for the off-line case, i.e.,

$$X_t = [\mathrm{col}({}^eA)', \mathrm{col}({}^eB)', \mathrm{col}({}^eF)', \mathrm{col}({}^eC)', \mathrm{col}({}^eD)', \mathrm{col}({}^eG)', {}^xr', {}^yr']',$$

where $\mathrm{col}(M)$ stacks the non-zero rows of the matrix M into a column vector.

The prior information on X_t is also identical to the off-line case (3.22), i.e., it holds $\underline{X_t} \le X_t \le \overline{X_t}$, $t \in t^* = \{\partial+1,\ldots,\mathring{t}\}$, with

$$\underline{X_t} = \left[\mathrm{col}({}^e\underline{A})', \mathrm{col}({}^e\underline{B})', \mathrm{col}({}^e\underline{F})', \mathrm{col}({}^e\underline{C})', \mathrm{col}({}^e\underline{D})', \mathrm{col}({}^e\underline{G})', \mathbf{0}_{(x^\ell,1)}, \mathbf{0}_{(y^\ell,1)}\right]', \tag{3.35}$$

$$\overline{X_t} = \left[\mathrm{col}({}^e\overline{A})', \mathrm{col}({}^e\overline{B})', \mathrm{col}({}^e\overline{F})', \mathrm{col}({}^e\overline{C})', \mathrm{col}({}^e\overline{D})', \mathrm{col}({}^e\overline{G})', {}^x\overline{r}, {}^y\overline{r}\right]'.$$

The matrix $\mathcal{A}_t$ and vector $\mathcal{B}_t$ for LP (3.17) have the identical structure to the off-line case, see Section 3.3.1. They differ only in the number of the rows which correspond to the memory length ∂. $\mathcal{A}$ and $\mathcal{B}$ have the following form

$$\mathcal{A}_t = \begin{bmatrix} \mathcal{A}11_t & \mathcal{A}12_t & \mathcal{A}13_t \\ \mathcal{A}21_t & \mathcal{A}22_t & \mathcal{A}23_t \end{bmatrix}, \ \mathcal{B}_t = \begin{bmatrix} \mathcal{B}1_t \\ \mathcal{B}2_t \end{bmatrix}$$

with

$$\mathcal{A}11_t = \begin{bmatrix} SEL_{{}^eA,x_{t-1}} & SEL_{{}^eB,u_t} & SEL_{{}^eF,1} \\ -SEL_{{}^eA,x_{t-1}} & -SEL_{{}^eB,u_t} & -SEL_{{}^eF,1} \\ \vdots & \vdots & \vdots \\ SEL_{{}^eA,x_{t-\partial-1}} & SEL_{{}^eB,u_{t-\partial}} & SEL_{{}^eF,1} \\ -SEL_{{}^eA,x_{t-\partial-1}} & -SEL_{{}^eB,u_{t-\partial}} & -SEL_{{}^eF,1} \end{bmatrix},$$

where $sel_M(v_t, i)$ is defined by (3.21)

$$\mathcal{A}12_t = \mathbf{0}_{(2(\partial+1)x^\ell,n)}, \ \text{the upper bound on } n \text{ is } y^\ell(x^\ell+u^\ell+1)$$

$$\mathcal{A}13_t \equiv \mathbf{1}_{(2(\partial+1),1)} \otimes \left[-I_{(x^\ell)} \ \mathbf{0}_{(x^\ell,y^\ell)}\right]$$

$\mathcal{A}21_t = \mathbf{0}_{(2(\partial+1)y^\ell,m)}$, the upper bound on m is $x^\ell(x^\ell + u^\ell + 1)$,

$$\mathcal{A}23_t = \mathbf{1}_{(2(\partial+1),1)} \otimes \left[\mathbf{0}_{(y^\ell,x^\ell)} \ \ -I_{(y^\ell)}\right],$$

$$\mathcal{A}22_t \equiv \begin{bmatrix} SEL_{^eC,x_t} & SEL_{^eD,u_t} & SEL_{^eG,1} \\ -SEL_{^eC,x_t} & -SEL_{^eD,u_t} & -SEL_{^eG,1} \\ \vdots & \vdots & \vdots \\ SEL_{^eC,x_{t-\partial}} & SEL_{^eD,u_{t-\partial}} & SEL_{^eG,1} \\ -SEL_{^eC,x_{t-\partial}} & -SEL_{^eD,u_{t-\partial 1}} & -SEL_{^eG,1} \end{bmatrix},$$

where $sel_M(v_t, i)$ is defined by (3.21).

$$\mathcal{B}1_t = \begin{bmatrix} x_t - A_t x_{t-1} - B_t u_t - F_t \\ -x_t + A_t x_{t-1} + B_t u_t + F_t \\ \vdots \\ x_{t-\partial} - A_{t-\partial} x_{t-\partial-1} - B_{t-\partial} u_{t-\partial} - F_{t-\partial} \\ -x_{t-\partial} + A_{t-\partial} x_{t-\partial-1} + B_{t-\partial} u_{t-\partial} + F_{t-\partial} \end{bmatrix},$$

$$\mathcal{B}2_t = \begin{bmatrix} y_t - C_t x_t - D_t u_t - G_t \\ -y_t + C_t x_t + D_t u_t + G_t \\ \vdots \\ y_{t-\partial} - C_{t-\partial} x_{t-\partial} - D_{t-\partial} u_{t-\partial} - G_{t-\partial} \\ -y_{t-\partial} + C_{t-\partial} x_{t-\partial} + D_{t-\partial} u_{t-\partial} + G_{t-\partial} \end{bmatrix}.$$

To summarize, the on-line case of the parameter estimation differs from the off-line case only in the number of the inequalities that are involved into the LP task.

3.4.4 Joint estimation of the parameters, states and the noise bounds

In the case of the joint estimation parameters and states, the conditions for the performance of the LP (3.33) are not fulfilled. Thus, we cannot use LP (see Section 2.7.1) for the point estimation straightforwardly. We propose the following solutions of this situation.

1. Swapping-based joint estimation

The idea of this approach is to estimate the state $x^{t-\partial:t}$ using technique from Section 3.4.2, with parameters θ fixed at their last point estimates. The resulting estimates of states, $\hat{x}^{t-\partial:t}$, are subsequently used in technique from Section 3.4.3 to obtain new estimates of the parameters θ. Initial values of the estimates can be found in off-line mode using, for instance, a subspace method [21] or sampling methods, e.g. [22].

2. Expansion-based joint estimation

Here, the parameter and state estimation is performed in one step. We suppose that both state values and model parameters are unknown. Generally, only data set (inputs and outputs) and prior information are at disposal. The vector of estimated quantities takes the form, $t \in t^* = \{\partial + 1, \ldots, \mathring{t}\}$,

$$X_t = \left[(x^{t:t-\partial})', \mathrm{col}({}^{e}A)', \mathrm{col}({}^{e}B)', \mathrm{col}({}^{e}F)', \mathrm{col}({}^{e}C)', \mathrm{col}({}^{e}D)', \mathrm{col}({}^{e}G)', {}^{x}r', {}^{y}r'\right]', \tag{3.36}$$

where

col(M) stacks the non-zero rows of the matrix M into a column vector.

The MAP estimate (3.28) is searched for. The LP task (3.33) is solved here with X_t given by (3.36), $\mathcal{C} \equiv [\mathbf{0}'_{(m,1)}, \mathbf{1}'_{(x^\ell+y^\ell,1)}]'$, the upper bound on m is $(\partial+1)x^\ell + x^\ell x^\ell + x^\ell u^\ell + x^\ell + y^\ell x^\ell + u^\ell y^\ell + y^\ell$. The matrix $\mathcal{A}_t$ and vector $\mathcal{B}_t$ are obtained from the set $\mathcal{S}1_t$ (3.12), $t \in t^*$. The inequalities are to be reorganized with respect to the estimated quantity X_t (3.36). Now, the above mentioned inequalities contain also the nonlinear terms ${}^{e}Ax_{t-1}$ and ${}^{e}Cx_t$ with two unknown variables. Therefore, the conditions on linearity required by LP are not fulfilled. The linearization by means of the Taylor expansion [16] is proposed as follows

$$\begin{aligned}
{}^{e}Ax_{t-1} &= ({}^{e}A - \hat{A})x_{t-1} + \hat{A}x_{t-1} \\
&= ({}^{e}A - \hat{A})(x_{t-1} - \hat{x}_{t-1}) + ({}^{e}A - \hat{A})\hat{x}_{t-1} + \hat{A}x_{t-1} \\
&\approx {}^{e}A\hat{x}_{t-1} - \hat{A}\hat{x}_{t-1} + \hat{A}x_{t-1}, t \in t^*,
\end{aligned}$$

where $\hat{A}, \hat{x}_{t-1}$ are newest available estimates of parameters and states, respectively. It is supposed that mean of $({}^{e}A - \hat{A})(x_{t-1} - \hat{x}_{t-1}) \approx 0$.

Using similar expansion for ${}^{e}Cx_t$, $t \in t^*$, we get

$${}^{e}Cx_t \approx {}^{e}C\hat{x}_t - \hat{C}\hat{x}_t + \hat{C}x_t$$

Then, the resulting inequalities for LP (3.33), $t \in \{\partial + 1, \ldots, \mathring{t}\}$, $\tau \in \{t - \partial, \ldots, t\}$ are as follows

$$
\begin{array}{rcl}
x_\tau - {}^eA\hat{x}_{\tau-1} - \hat{A}x_{\tau-1} - A_\tau x_{\tau-1} - {}^eBu_\tau - {}^eF - {}^xr & \leq & -\hat{A}\hat{x}_{\tau-1} + B_\tau u_\tau + F_\tau \\
-x_\tau + {}^eA\hat{x}_{\tau-1} + \hat{A}x_{\tau-1} + A_\tau x_{\tau-1} + {}^eBu_\tau + {}^eF - {}^xr & \leq & +\hat{A}\hat{x}_{\tau-1} - B_\tau u_\tau - F_\tau \\
{}^eC\hat{x}_\tau + \hat{C}x_\tau + C_\tau x_\tau + {}^eDu_\tau + {}^eG - {}^yr & \leq & +y_\tau + \hat{C}\hat{x}_\tau - D_\tau u_\tau - G_\tau \\
-{}^eC\hat{x}_\tau - \hat{C}x_\tau - C_\tau x_\tau - {}^eDu_\tau - {}^eG - {}^yr & \leq & -y_\tau - \hat{C}\hat{x}_\tau + D_\tau u_\tau + G_\tau
\end{array}
$$

Note that the estimates based on the data up to time $t-1$, i.e., $d^{t-\partial-1:t-1}$ are used for LP performed in the time t. Therefore only estimates from $\hat{x}_{t-\partial-1}$ up to $\hat{x}_{t-1}$ are at disposal. The estimate $\hat{x}_t$ is replaced by its prediction, i.e.

$$
\hat{x}_t = (A_t + \hat{A})\hat{x}_{t-1} + (B_t + \hat{B})u_t + (F_t + \hat{F}). \tag{3.37}
$$

Then, after proposed linearization, $\mathcal{A}_t$, $\mathcal{B}_t$ are in the form

$$
\mathcal{A}_t = \begin{bmatrix} \mathcal{A}11_t & \mathcal{A}12_t & \mathcal{A}13_t & \mathcal{A}14_t \\ \mathcal{A}21_t & \mathcal{A}22_t & \mathcal{A}23_t & \mathcal{A}24_t \end{bmatrix}, \; \mathcal{B}_t = \begin{bmatrix} \mathcal{B}1_t \\ \mathcal{B}2_t \end{bmatrix}
$$

with

$$
\mathcal{A}11_t = \begin{bmatrix}
I_{(x^\ell,x^\ell)} & -\hat{A} - A_t & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\
-I_{(x^\ell,x^\ell)} & \hat{A} + A_t & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\
\mathbf{0}_{(x^\ell,x^\ell)} & I_{(x^\ell,x^\ell)} & -\hat{A} - A_{t-1} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\
\mathbf{0}_{(x^\ell,x^\ell)} & -I_{(x^\ell,x^\ell)} & \hat{A} + A_{t-1} & \dots & \mathbf{0}_{(x^\ell,x^\ell)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0}_{(x^\ell,x^\ell)} & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \dots & I_{(x^\ell,x^\ell)} \\
\mathbf{0}_{(x^\ell,x^\ell)} & \mathbf{0}_{(x^\ell,x^\ell)} & \dots & \dots & -I_{(x^\ell,x^\ell)}
\end{bmatrix},
$$

$$
\mathcal{A}12_t = \begin{bmatrix}
-SEL_{{}^eA,\hat{x}_{t-1}} & -SEL_{{}^eB,u_t} & -SEL_{{}^eF,1} \\
SEL_{{}^eA,\hat{x}_{t-1}} & SEL_{{}^eB,u_t} & SEL_{{}^eF,1} \\
\vdots & \vdots & \vdots \\
-SEL_{{}^eA,\hat{x}_{t-\partial-1}} & -SEL_{{}^eB,u_{t-\partial}} & -SEL_{{}^eF,1} \\
SEL_{{}^eA,\hat{x}_{t-\partial-1}} & SEL_{{}^eB,u_{t-\partial}} & SEL_{{}^eF,1}
\end{bmatrix},
$$

where $sel_M(v_t, i)$ is defined by (3.21),
$\mathcal{A}12_t$ has $2(\partial+1)x^\ell$ rows and m columns, upper bound on m is $x^\ell(x^\ell+u^\ell+1)$,
$\mathcal{A}13_t = \mathbf{0}_{(2(\partial+1)x^\ell,n)}$, maximal size of n is $y^\ell(x^\ell+u^\ell+1)$.

$$
\mathcal{A}14_t \equiv \mathbf{1}_{(2(\partial+1),1)} \otimes \left[-I_{(x^\ell)} \; \mathbf{0}_{(x^\ell,y^\ell)}\right],
$$

$$
\mathcal{A}21_t = \begin{bmatrix}
\hat{C} + C_t & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & \mathbf{0}_{(y^\ell,x^\ell)} \\
-\hat{C} - C_t & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & \mathbf{0}_{(y^\ell,x^\ell)} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}_{(y^\ell,x^\ell)} & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & \hat{C} + C_{t-\partial} \\
\mathbf{0}_{(y^\ell,x^\ell)} & \mathbf{0}_{(y^\ell,x^\ell)} & \dots & -\hat{C} - C_{t-\partial}
\end{bmatrix},
$$

$\mathcal{A}22_t = \mathbf{0}_{2(\partial+1)y^\ell,m}$, m is defined by $\mathcal{A}11$,

$$\mathcal{A}23_t \equiv \begin{bmatrix} SEL_{^eC,\hat{x}_t} & SEL_{^eD,u_t} & SEL_{^eG,1} \\ -SEL_{^eC,\hat{x}_t} & -SEL_{^eD,u_t} & -SEL_{^eG,1} \\ \vdots & \vdots & \vdots \\ SEL_{^eC,\hat{x}_{t-\partial}} & SEL_{^eD,u_{t-\partial}} & SEL_{^eG,1} \\ -SEL_{^eC,\hat{x}_{t-\partial}} & -SEL_{^eD,u_{t-\partial 1}} & -SEL_{^eG,1} \end{bmatrix},$$

where $sel_M(v_t, i)$ is defined by (3.21), $\hat{x}_t$ is obtained as the prediction (3.37). $\mathcal{A}23_t$ has $2(\partial+1)y^\ell$ rows and n columns with n defined by $\mathcal{A}12$.
$\mathcal{A}24_t = \mathbf{1}_{(2(\partial+1),1)} \otimes \left[\mathbf{0}_{(y^\ell,x^\ell)} \; - I_{(y^\ell)}\right]$,

$$\mathcal{B}1_t = \begin{bmatrix} -\hat{A}\hat{x}_{t-1} + B_t u_t + F_t \\ +\hat{A}\hat{x}_{t-1} - B_t u_t - F_t \\ \vdots \\ -\hat{A}\hat{x}_{t-\partial} + B_{t-\partial+1}u_{t-\partial+1} + F_{t-\partial+1} \\ +\hat{A}\hat{x}_{t-\partial} - B_{t-\partial+1}u_{t-\partial+1} - F_{t-\partial+1} \\ A_{t-\partial}\hat{x}_{t-\partial-1} + B_{t-\partial}u_{t-\partial} + F_{t-\partial} \\ -A_{t-\partial}\hat{x}_{t-\partial-1} - B_{t-\partial}u_{t-\partial} - F_{t-\partial} \end{bmatrix},$$

where $\hat{x}_t$ is obtained as the prediction (3.37),

$$\mathcal{B}2_t = \begin{bmatrix} \hat{C}\hat{x}_t + y_t - D_t u_t - G_t \\ -\hat{C}\hat{x}_t - y_t + D_t u_t + G_t \\ \vdots \\ \hat{C}\hat{x}_{t-\partial} + y_{t-\partial} - D_{t-\partial}u_{t-\partial} - G_{t-\partial} \\ -\hat{C}\hat{x}_{t-\partial} - y_{t-\partial} + D_{t-\partial}u_{t-\partial} + G_{t-\partial} \end{bmatrix}.$$

The prior information on X_t combines (3.34) and (3.35). It has the form

$$\underline{X}_t = \begin{bmatrix} \mathbf{1}_{(2(\partial+1)x^\ell,1)} \otimes \underline{x} \\ \hat{x}_{t-\partial-1} \\ \mathrm{col}(\,^e\underline{A}) \\ \mathrm{col}(\,^e\underline{B}) \\ \mathrm{col}(\,^e\underline{F}) \\ \mathrm{col}(\,^e\underline{C}) \\ \mathrm{col}(\,^e\underline{D}) \\ \mathrm{col}(\,^e\underline{G}) \\ \mathbf{0}_{(x^\ell,1)} \\ \mathbf{0}_{(y^\ell,1)} \end{bmatrix}, \quad \overline{X}_t = \begin{bmatrix} \mathbf{1}_{(2(\partial+1)x^\ell,1)} \otimes \overline{x} \\ \hat{x}_{t-\partial-1} \\ \mathrm{col}(\,^e\overline{A}) \\ \mathrm{col}(\,^e\overline{B}) \\ \mathrm{col}(\,^e\overline{F}) \\ \mathrm{col}(\,^e\overline{C}) \\ \mathrm{col}(\,^e\overline{D}) \\ \mathrm{col}(\,^e\overline{G}) \\ x_{\overline{r}} \\ y_{\overline{r}} \end{bmatrix}. \qquad (3.38)$$

Note that the resulting algorithms have two principal distinctions from the extended KF: (i) the algorithm updates estimates on the whole window of length ∂ and (ii) the realistic hard bounds on the estimated quantities reduce the ambiguity of the model arising from estimating a product of two unknowns.

3.5 Specification of the minimal memory length

The maximal value of the memory length $\partial_{\max}$ is limited by the computational feasibility. For the specification of the minimal memory length $\partial_{\min}$, we stem from the assumption that the number of the inequalities for LP should be comparable with the size X^{ℓ} of the estimated quantity X. For the memory length ∂, the number of the inequalities for LP (3.17) is $2(\partial+1)(x^{\ell}+y^{\ell})$. That number follows from the length of the vector $\mathcal{B}$ in (3.17).

The requirements on $\partial_{\min}$ for the particular tasks are as follows.

State estimation
- $X^{\ell} = x^{\ell}(\partial+1) + x^{\ell} + y^{\ell}$
- $\partial_{\min} = 1$

Parameters estimation
- $X^{\ell} = (x^{\ell}+y^{\ell})(x^{\ell}+y^{\ell}+2)$
- $\partial_{\min} = (x^{\ell}+u^{\ell})/2$

Joint estimation
- $X^{\ell} = x^{\ell}(\partial+1) + p^{\ell} + x^{\ell} + y^{\ell}$
- $\partial_{\min} = (p^{\ell}-y^{\ell})/(x^{\ell}+2y^{\ell}), \quad 1 \le p^{\ell} \le (x^{\ell}+y^{\ell})(x^{\ell}+u^{\ell}+1)$

Note that all tasks include the estimation of the innovations boundary.

Chapter 4

LU model of intersection

In this section, the LU state model (see Section 3.1) of the intersection is constructed. First, the the main transportation characteristics are introduced. Then, the the state and output model equations are designed. They stem from the hydro-dynamical analogy described in [23]. There, the law of mass conservation is exploited. In short: "The number of the cars that arrive to the intersection is equal to the number of cars that depart it plus the number of cars that stay in the queue." The proposed LU state model of the intersection substantially differ from the state model defined in [23]. The main differences are (i) in the type of the innovations and (ii) in the style of the queue indicator.

4.1 Basic transportation characteristics

The considered intersection is controlled by the traffic lights. For the purpose of the traffic control design, it is important to know lengths of the car queues that are formed on the arms of the intersection. The queues cannot be measured directly. The queue length has to be estimated. The estimation is based on the knowledge of characteristics, which are measured on the traffic detectors.

All quantities needed for construction of the intersection model are summarized in the following table where period means duration of the whole traffic light (TL) cycle.

name	notation	unit	description
queue length	$\xi_{t;i}$	u.c.	the number of the cars before the TL (for the i-th arm)
occupancy	$O_{t;i}$	%	relative time of detector activation, i.e., the ratio detector activation to the period length
input intensity	$I_{t;i}$	u.c/per.	amount of cars passing per period through the input detector
passage	$P_{t;i}$	u.c/per.	amount of cars passing per period from arm i into the intersection space
output intensity	$Y_{t;i}$	u.c/per.	amount of cars passing per period through the output detector
saturated flow	S_i	u.c/per.	saturated flow, i.e., maximal amount of cars that can go through the arm i of the intersection per period
turning rate	α_{ji}	–	ratio of cars that from direction j turn to the direction i to all cars from direction j
green time	$z_{t;i}$	–	ratio of the "green" time and period (TL cycle time)
–	κ_i, β_i, λ_i	–	constants describing linear relation between queue length and occupancy

where
u.c. means unit car
t is time index
i, j denote i-th and j-th arm of the intersection, respectively
TL means the traffic light
per. is period

Note that the period of the TL cycle is taken as unit for the simplicity and therefore omitted in the following computations.

Note that the input intensity I is denoted by the same symbol as the unit matrix. We use this notation only in this Section and in the Section 5.2 where an example with transportation data is presented. In these Sections, the unit matrix is not used. Therefore, the misunderstanding cannot occur.

The following relations hold for each arm of the n-arm intersection (see an example of the 4-arm intersection on Figure 4.1):

$$
\begin{aligned}
\xi_{t+1;i} &= \xi_{t;i} - P_{t;i} + I_{t;i} && (4.1)\\
P_{t;i} &= \min(\xi_{t;i} + I_{t;i} z_{t;i},\ S_i z_{t;i}) && (4.2)\\
O_{t+1;i} &= \kappa_i \xi_{t;i} + \beta_i O_{t;i} + \lambda_i && (4.3)\\
Y_{t;i} &= \sum_{j=1}^{n} \alpha_{ji}\, P_{t;j} = \sum_{j=1}^{n} \alpha_{ji} \min\left[\xi_{t;j} + I_{t;j} z_{t;j},\ S_j z_{t;j}\right] && (4.4)\\
\alpha_j &= [\alpha_{j\,1}, \ldots, \alpha_{j\,n}]' && (4.5)\\
\sum_{i=1}^{n} \alpha_{ji} &= 1,\ \alpha_{jj} = 0,\ \alpha_{ji} \geq 0,\ i, j = 1, \ldots, n && (4.6)\\
Y_{t+1;i} &= \sum_{j=1}^{n} \alpha_{ji}(-\xi_{t+1;j} + I_{t;j} + \xi_{t;j}) && (4.7)
\end{aligned}
$$

where
$i, j \in \{1, 2, \ldots, n\}$.

We shall use the matrix aggregation $\xi_t = [\xi_{t;1}, \ldots, \xi_{t;n}]'$, $P_t = [P_{t;1}, \ldots, P_{t;n}]'$, etc.

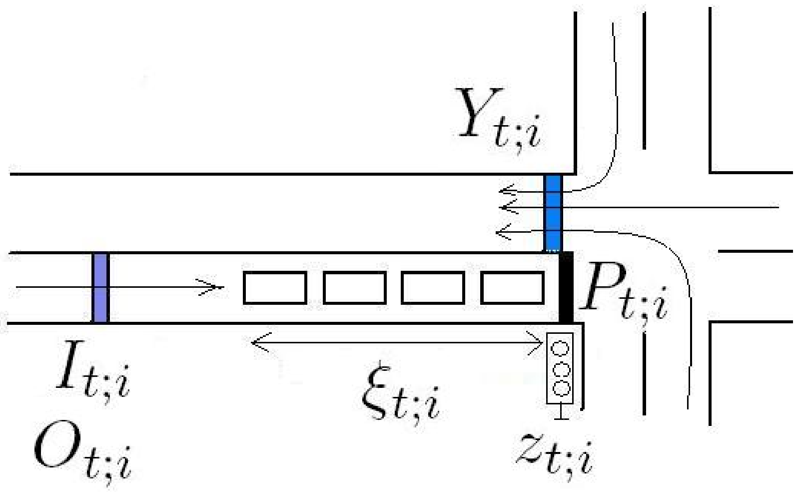

Figure 4.1: The 4-arm intersection with quantities related to i-th arm

4.2 Intersection model

Now, the n-arm intersection will be described by LU state model that is introduced in Section 3.1. Here, the state x_t is represented by the pair of queue length and occupancy, i.e., $x_t = [\xi_t', O_t']'$, the output y_t contains output intensity and occupancy, i.e., $y_t = [Y_t', O_t']'$. The "relative green" z_t represents the system input, i.e., $u_t = z_t$. Then, the model of the n-arm intersection has the form

$$\begin{bmatrix} \xi_t \\ O_t \end{bmatrix} = {}^cA_t \begin{bmatrix} \xi_{t-1} \\ O_{t-1} \end{bmatrix} + {}^cB_t z_t + {}^cF_t + {}^xe_t \tag{4.8}$$

$$\begin{bmatrix} Y_t \\ O_t \end{bmatrix} = {}^cC_t \begin{bmatrix} \xi_t \\ O_t \end{bmatrix} + {}^cD_t z_t + {}^cG_t + {}^ye_t, \tag{4.9}$$

where the specific form of system matrices ${}^cA_t$, ${}^cB_t$, etc. will be constructed using the relations (4.1) - (4.7) as follows.

First, the nonlinear form (4.2) is treated. We can write it equivalently

$$P_{t;i} = (1 - p_{t;i})(\xi_{t;i} + I_{t;i} z_{t;i}) + p_{t;i} S_i z_{t;i},\ i = 1, \ldots, n,\ t \in t^* \tag{4.10}$$

with the queue indicator $p_{t;i}$

$$\begin{aligned} p_{t;i} = 0 \quad &\text{for} \quad \xi_{t;i} + I_{t;i} z_{t;i} < S_i z_{t;i}, \\ p_{t;i} = 1 \quad &\text{for} \quad \xi_{t;i} + I_{t;i} z_{t;i} > S_i z_{t;i}. \end{aligned}$$

We don't know the true value of $\xi_{t;i}$, we have only its estimate at disposal. Therefore, an approximation of $p_{t;i}$ is needed. We interpret $p_{t;i}$ as "weight", i.e., $p_{t;i} \in [0, 1]$. We propose for (4.10) the following requirements on $p_{t;i}$

$$\begin{aligned} p_{t;i} = 0 \quad &\text{for} \quad \hat{\xi}_{t;i} + I_{t;i} z_{t;i} \leq S_i z_{t;i} - k, \\ p_{t;i} = 0.5 \quad &\text{for} \quad \hat{\xi}_{t;i} + I_{t;i} z_{t;i} = S_i z_{t;i}, \\ p_{t;i} = 1 \quad &\text{for} \quad \hat{\xi}_{t;i} + I_{t;i} z_{t;i} \geq S_i z_{t;i} + k. \end{aligned} \tag{4.11}$$

with a small $k > 0$.

The value of $p_{t;i}$ will be set in the each step according to the current value of the intensity $I_{t;i}$ and the current estimate $\hat{\xi}_{t;i}$.

The requirements (4.11) can be met by the use of the function

$$p(y) = \frac{1}{1 + e^{b(a-y)}} \tag{4.12}$$

where $p(y) \in [0; 1]$, $p(a) = 0.5$, b determines the rate of transition between 0 and 1, see Figure 4.2.

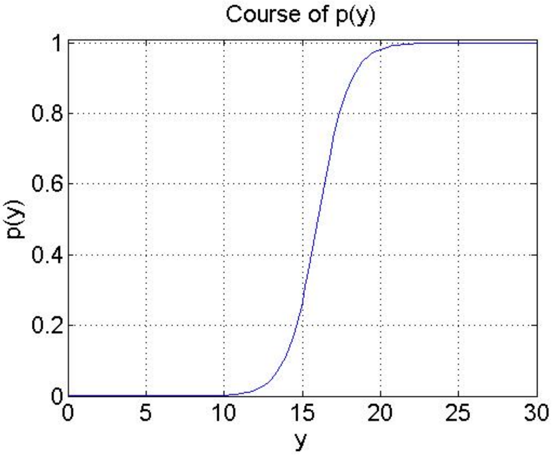

Figure 4.2: Weight $p_{t;i} = p(y)$ depending on value $\hat{\xi}_{t;i} + I_{t;i} z_{t;i} = y$

Here, $y = \hat{\xi}_{t;i} + I_{t;i} z_{t;i}$, $a = S_i z_{t;i}$.

Note that the original model defined in [23] instead of the weight p works with a "two-throw switch" δ.

With the proposed weight p, the model matrices in (4.8) and (4.9) are as follow

$$^cA_t = \begin{bmatrix} \mathrm{diag}(p_t) & \mathbf{0}_{(n,n)} \\ \mathrm{diag}(\kappa_t) & \mathrm{diag}(\beta_t) \end{bmatrix}, \; ^cB_t = \begin{bmatrix} -\mathrm{diag}(p_t \,.*\, S + (1-p_t) \,.*\, I_t) \\ \mathbf{0}_{(n,n)} \end{bmatrix}, \; ^cF_t = \begin{bmatrix} I_t \\ \lambda_t \end{bmatrix}$$

$$^cC_t = \begin{bmatrix} c_t & \mathbf{0}_{(n,n)} \\ \mathbf{0}_{(n,n)} & I_{(n,n)} \end{bmatrix}, \; ^cD_t = \begin{bmatrix} d_t \\ \mathbf{0}_{(n,n)} \end{bmatrix}, \; ^cG_t = \mathbf{0}_{(2n,1)}, \tag{4.13}$$

where (n,n) sub-matrices c_t and d_t have the entries

$$c_{t;ij} = \alpha_{ji}(1 - p_{t;j}), \;\; d_{t;ij} = \alpha_{ji}\left[(1 - p_{t;j}) I_{t;j} + p_{t;j} S_j\right]$$

and

p_t, κ_t, β_t, λ_t are column vectors, $p_t = [p_{t;1}, \ldots, p_{t;n}]', etc.$

$$\mathrm{diag}(v) = \begin{bmatrix} v_1 & 0 & \ldots & 0 \\ 0 & v_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & v_{m^\ell} \end{bmatrix}, \; v \text{ is a column vector;}$$

$\alpha = \begin{bmatrix} \alpha_1 & \ldots & \alpha_n \end{bmatrix}'$;

$\hat{\xi}_t$ is the point estimate of the state ξ_t;

$.*$ means element-by-element multiplication

4.3 Joint on-line estimation of the intersection

Joint on-line parameter and state estimation, see page 37, consists in searching for the MAP estimate of X_t (3.36), $t \in t^*$. This problem is solved by the LP (3.33). Here, the estimated quantity

$$X_t = \left[(x^{t:t-\partial})', \mathrm{col}({}^{e}A)', \mathrm{col}({}^{e}B)', \mathrm{col}({}^{e}F)', \mathrm{col}({}^{e}C)', \mathrm{col}({}^{e}D)', \mathrm{col}({}^{e}G)', {}^{x}r', {}^{y}r'\right]'$$

has the following entries (col(M) stacks the non-zero rows of the matrix M into a column vector)

$$ {}^{e}A = \begin{bmatrix} \mathbf{0}_{(n,n)} & \mathbf{0}_{(n,n)} \\ \mathrm{diag}(\kappa_t) & \mathrm{diag}(\beta_t) \end{bmatrix}, \; {}^{e}B = \mathbf{0}_{(2n,n)}, \; {}^{e}F = \begin{bmatrix} \mathbf{0}_{(n,1)} \\ \lambda_t \end{bmatrix} \tag{4.14}$$

$$ {}^{e}C = \mathbf{0}_{(2n,2n)}, \; {}^{e}D = \mathbf{0}_{(2n,n)}, \; {}^{e}G = \mathbf{0}_{(2n,1)}. $$

The known part of the model matrices (4.13) are as follows

$$ A_t = \begin{bmatrix} \mathrm{diag}(p_t) & \mathbf{0}_{(n,n)} \\ \mathbf{0}_{(n,n)} & \mathbf{0}_{(n,n)} \end{bmatrix}, \; B_t = {}^{c}B_t, \; {}^{c}F_t = \begin{bmatrix} I_t \\ \mathbf{0}_{(1,n)} \end{bmatrix} \tag{4.15}$$

$$ C_t = {}^{c}C_t, \; D_t = {}^{c}D_t, \; G_t = {}^{c}G_t, $$

where

- the value of the saturated flow S is given by the intersection properties
- the value of the green ratio z_t is given by the traffic scheme
- the values of the input intensity I_t are obtained from the traffic detectors
- the values of the queue indicator p_t are obtained in the each time step by the help of (4.12).
- the values of the turning rates α are either substitute by their mean values obtained by the long-term measuring on the intersection or estimated on-line by the help of equations (4.4) and (4.6)

The boundaries for LP, $\underline{X}_t$ and $\overline{X}_t$, are in the form (3.38) with
$0 \le \xi_{t;i}$, $0 \le O_{t;i} \le 100$, $0 \le {}^{x}r_i \le 2$, $0 \le {}^{x}r_i \le 2$, $0 \le \kappa_i$, $0 \le \beta_i$, $0 \le \lambda_i$, $i \in \{1, \le, n\}$.

Chapter 5

Experiments

Here, two examples are presented: (i) a simple example that illustrates the properties of the proposed algorithms and (ii) an application of the estimation algorithm to the transportation data that were obtained from the Aimsun environment [1] (see www.aimsun.com).

5.1 Simulated example

The purpose of the simple example is to present the main features of the estimation algorithms.

5.1.1 Model description

The two state system with scalar input and output and uniform noise is simulated. The model is described by the LU model (3.1), (3.2) and (3.4) with constant model matrices (with time indexes omitted), i.e.,

$$x_t = \; {}^cAx_{t-1} + {}^cBu_t + {}^cF + {}^xe_t, \quad f\left({}^xe_t\right) = \mathcal{U}\left(0, {}^xr\right), \tag{5.1}$$

$$y_t = \; {}^cCx_t + {}^cDu_t + {}^cG + {}^ye_t, \quad f\left({}^ye_t\right) = \mathcal{U}\left(0, {}^yr\right) \tag{5.2}$$

with

$${}^cA = \begin{bmatrix} 1 & 0.5 \\ -0.5 & 0 \end{bmatrix}, \; {}^cB = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \; {}^cF = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \; {}^xr = \begin{bmatrix} 10^{-1} \\ 10^{-1} \end{bmatrix},$$

$${}^cC = \begin{bmatrix} 1 & 1 \end{bmatrix}, \; {}^cD = 0, \; {}^cG = 1, \; {}^yr = 10^{-1}.$$

[1]AIMSUN is a microscopic traffic simulator that can deal with different traffic networks: urban networks, freeways, highways, ring roads, arterial and any combination thereof. It has been designed and implemented as a tool for traffic analysis to help traffic engineers in the design and assessment of traffic systems. It has proven to be very useful for testing new traffic control systems and management policies, either based on traditional technologies or as implementation of Intelligent Transport Systems.

The input is simulated as a random signal, uniformly distributed on the interval $[-1, 1]$, with independent values for different time moments. The data set consists of $\mathring{t} = 500$ data pairs (inputs and outputs).

Using the notation (3.3), we get

$$^{c}M = M + {}^{e}M$$

where M contains known entries of ^{c}M and ^{e}M contains estimated entries of ^{c}M, $M \in \{A,\ B,\ F,\ C,\ D,\ G\}$.

5.1.2 Evaluation of experiments

To evaluate the quality of the estimation, the mean error (ME) of the output predictions and ME of state or parameter estimates are used, respectively. Generally, ME of the quantity E is computed entry-wise in the following way

$$ME_E = \frac{1}{\mathring{t}} \sum_{t=1}^{\mathring{t}} |E_t - R_t| \tag{5.3}$$

where E_t are the predicted outputs, state estimates and parameter estimates, respectively, R_t are the true values of the outputs and states, respectively, $\mathring{t}$ is the number of the samples. In the case of the parameter estimation $R_t = R,\ t \in \{1, 2, ..., \mathring{t}\}$, $R \in \{{}^{c}A,\ {}^{c}B,\ {}^{c}F,\ {}^{c}C,\ {}^{c}D,\ {}^{c}G\}$.

Further, the Matlab function "polyfit" is used to fit a polynomial to data

$$p = \mathrm{polyfit}(x, y, n) \tag{5.4}$$

that finds the coefficients of a polynomial $p(x)$ of degree n that fits the data y best in a least-squares sense; p is a row vector of length $n+1$ containing the polynomial coefficients in descending powers, i.e.,

$$p_1 x^n + p_2 x^{(n-1)} + ... + p_n x + p_{n+1}.$$

In this way, we can find tendencies in the interrelationships of the observed quantities.

5.1.3 State estimation

Here, the states x_t, $t \in t^*$, and the innovation boundaries ^{x}r, ^{y}r are estimated while all system matrices are supposed to be known. The estimation algorithm (described in Section 3.4.2) is running on-line with the various memory lengths ∂.

Figure 5.1 shows how the ME (5.3) of the output predictions depends on the memory length ∂. We can see that ME has from the beginning the lower values than for higher values of ∂ but there is a sharp decrease around $\partial \approx 20$.

Figure 5.2 shows how the ME (5.3) of the state estimates depends on the memory length ∂. We can see that the ME of the state estimates decreases at the beginning relatively quickly and then stays within a narrow interval. A jump is observable again around $\partial \approx 20$.

The dependence of the computation time t_∂ on the memory lengths ∂ for chosen values of ∂ is displayed in Table 5.1. The computational time t_∂ increases with the increasing memory length ∂ approximately according to the relation

$$t_\partial \approx 0.075\partial^2 - 1.19\partial + 19.09\,[s].$$

This relation was obtained by a function "polyfit" (5.4) from the known values t_∂ and ∂ and has only a relative meaning. The absolute values depend on the efficiency of used computer.

∂	5	10	15	20	25	30	35	40	45	50	55	60
time t_∂ [s]	11	18	27	37	54	73	98	123	159	197	249	304

Table 5.1: Computation time t_∂ depending on memory length ∂

5.1.4 Parameter estimation

Here, the model matrices ${}^eA$, ${}^eB$, ${}^eF$, ${}^eC$, ${}^eD$, ${}^eG$ and the innovation boundaries ${}^xr$, ${}^yr$ are estimated while the states x_t, $t \in t^*$, are supposed to be known. The estimation algorithm described in Section 3.4.3 is running on-line with various memory lengths ∂.

Figure 5.3 shows how ME (5.3) of output prediction depends on the memory length ∂. Similarly to the case of the state estimation, ME is smaller for the small values of ∂. The jump around $\partial \approx 20$ is less obvious.

To illustrate how ME (5.3) of the parameter estimates depends on the ∂, the estimate of ${}^eC$ in (5.2) were chosen as a representant, see Figure 5.4. We can see that ME decreases at the beginning and then oscillates within a narrow interval.

The dependence of the computation time t_∂ on the memory lengths ∂ is displayed in Table 5.2. From measured values of t_∂ and ∂, the following relation was obtained by the function "polyfit" (5.4):

$$t_\partial \approx 0.055\partial^2 - 0.731\partial + 20.53[s]$$

and it has again only a relative meaning. The absolute values depend on the efficiency of used computer.

∂	5	10	15	20	25	30	35	40	45	50	55	60
time (s)	13	19	25	32	41	53	67	82	99	121	142	163

Table 5.2: Computation time t_∂ depending on memory length ∂

5.1.5 Joint parameter and state estimation

Here, the states x_t, $t \in t^*$, the model matrices ${}^eA$, ${}^eB$, ${}^eF$, ${}^eC$, ${}^eD$, ${}^eG$ and the innovation boundaries ${}^xr$, ${}^yr$ are estimated. The estimation algorithm described in the Section 3.4.4 is running on-line with various memory lengths ∂. Two cases are considered.

1. All model matrices are completely estimated, i.e., ${}^eA = {}^cA$, ${}^eB = {}^cB$, etc. The restrictions on the entries of the model matrices are chosen so that all entries of the estimated model matrices should be within the interval given by

$$ {}^cM(i,j) - K_1 \leq {}^eM(i,j) \leq {}^cM(i,j) + K_2, $$

where ${}^cM(i,j)$ and ${}^eM(i,j)$, $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,n\}$, are i-th rows and j-th column entry of the simulated model matrix ${}^cM$ and estimated model matrix ${}^eM$, respectively, both of the size (m,n), K_1, K_2 are positive scalars.

2. The 2nd row and 2nd column entry of the model matrix A and the vector F are supposed to be known. The above mentioned restrictions are used on the remaining entries.

The comparison of the ME (5.3) of the state estimates for cases **1** and **2** and various memory lengths ∂ is on Figure 5.5. We can see that ME of the state estimates is smaller in the case of the partially known model matrices.

The comparison of the ME (5.3) of model parameters is on the Figure 5.6. The parameter ${}^cG$ in (5.2) was chosen as a representative because of the most remarkable difference between the case 1 and 2.

5.1.6 Discussion of the results

State estimation

The smaller values of ∂ provide more freedom in the fitting of the available data items, i.e., smaller fitting errors can be achieved. On the other hand, we have not enough data for more precise state estimation and their uncertainty increases prediction errors.

For our purposes, it is important to have the small state estimation errors. Therefore, the ∂ need to be sufficiently long to reach the interval of the minimal state errors. Although the computation time increases nonlinearly with the increasing ∂, the experiment shows that the estimation is feasible because of the relatively fast decreasing of ME of the state estimates.

The jumps around $\partial \approx 20$ indicate this value as a proper one for the studied example.

Parameter estimation

We can summarize similarly to the previous case of the state estimation.

The smaller values of ∂ provide more freedom in the fitting of the available data items, i.e., smaller fitting errors can be achieved. On the other hand, we have not enough data for more precise parameter estimation.

For our purposes, it is important to have the small state estimation errors. Although the computation time increases nonlinearly with the increasing ∂, the experiment shows that the estimation is feasible because of the relatively fast decreasing of ME of the parameters estimates.

In this case, $\partial \approx 20$ could be also taken as a reasonable choice but less supported by the results.

Joint estimation

The experiment with joint parameter and state estimation supports the intuitive expectation that the knowledge of some model matrices entries improves the quality of the state and parameter estimation measured by the ME of estimate errors, see Figures 5.5 and 5.6. Moreover, the partial knowledge of the model matrices decreases or even remove the ambiguity of the state and parameter estimates caused by the products ${}^{e}\!A\, x_{t-1}$ and ${}^{e}\!C\, x_t$ in the model equations (5.1) and (5.2), respectively. This fact is important in the real data application. There, every small piece of the information about model matrices improves the quality of the estimation. In the LU model, the incorporation of our prior knowledge into the model is very easy. It consists on the adding of the constraints, i.e., additional inequalities, into the linear programming.

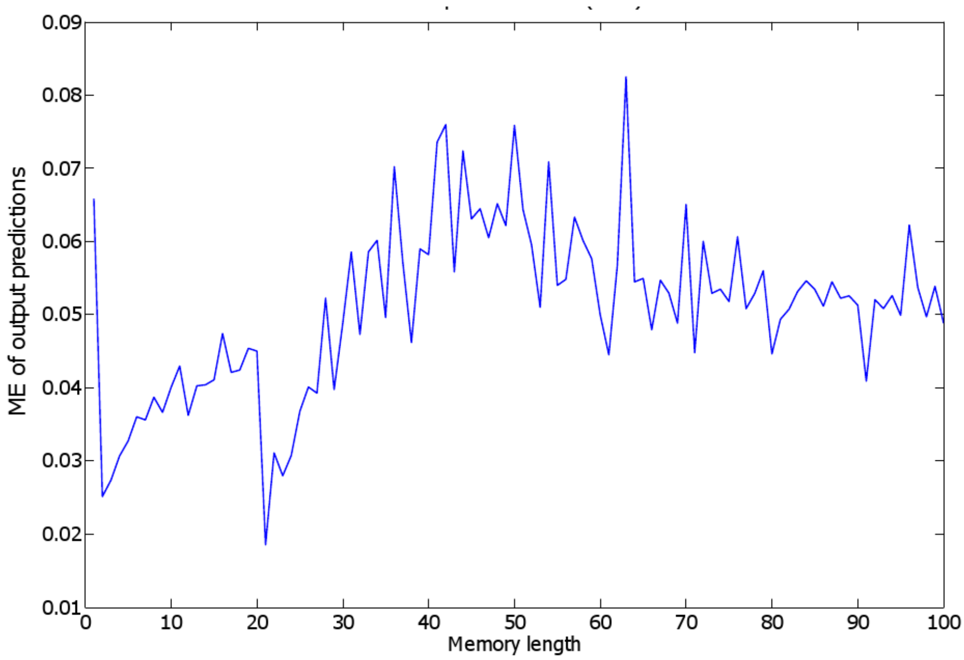

Figure 5.1: ME of the output predictions depending on the memory length ∂

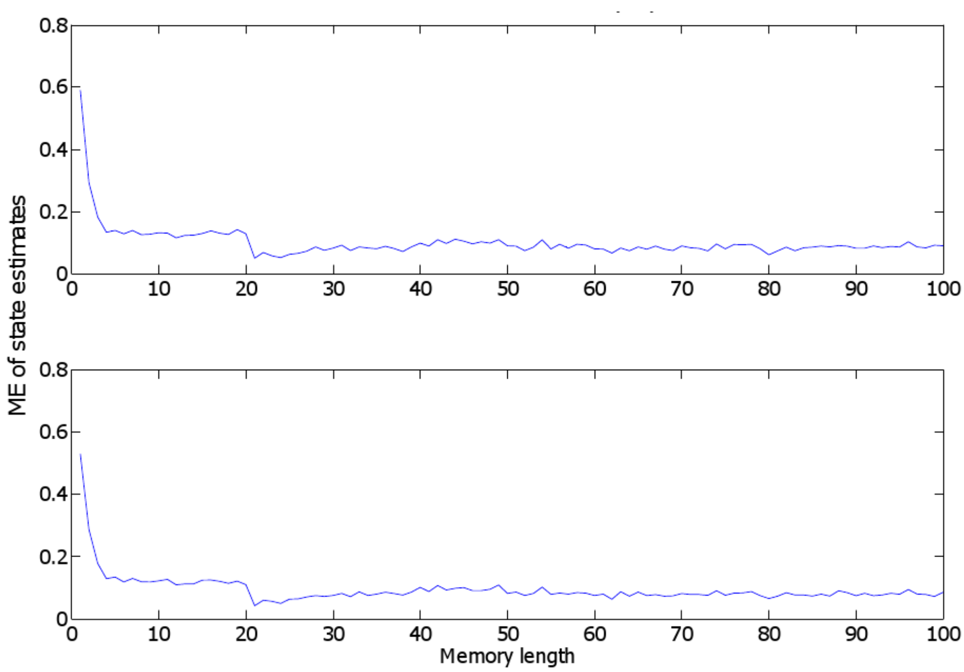

Figure 5.2: ME of the state estimates depending on the memory length ∂

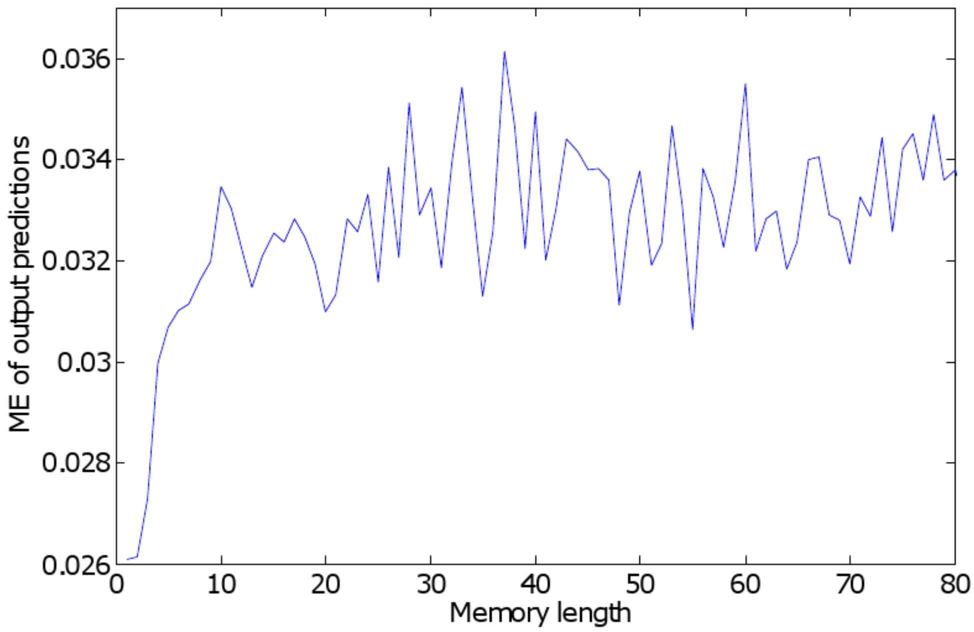

Figure 5.3: ME of the output predictions depending on the memory length ∂

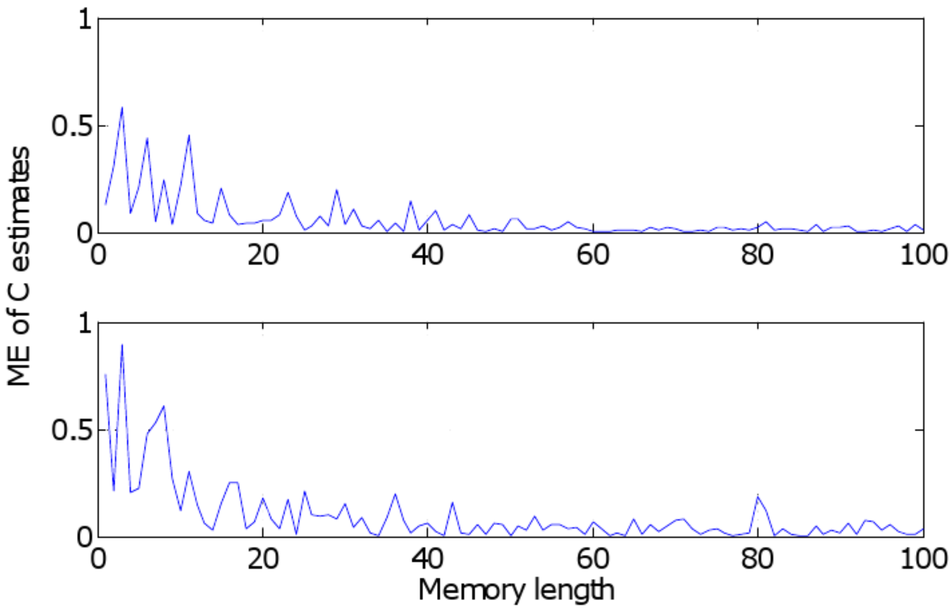

Figure 5.4: ME of ${}^{e}C$ estimates depending on memory length ∂

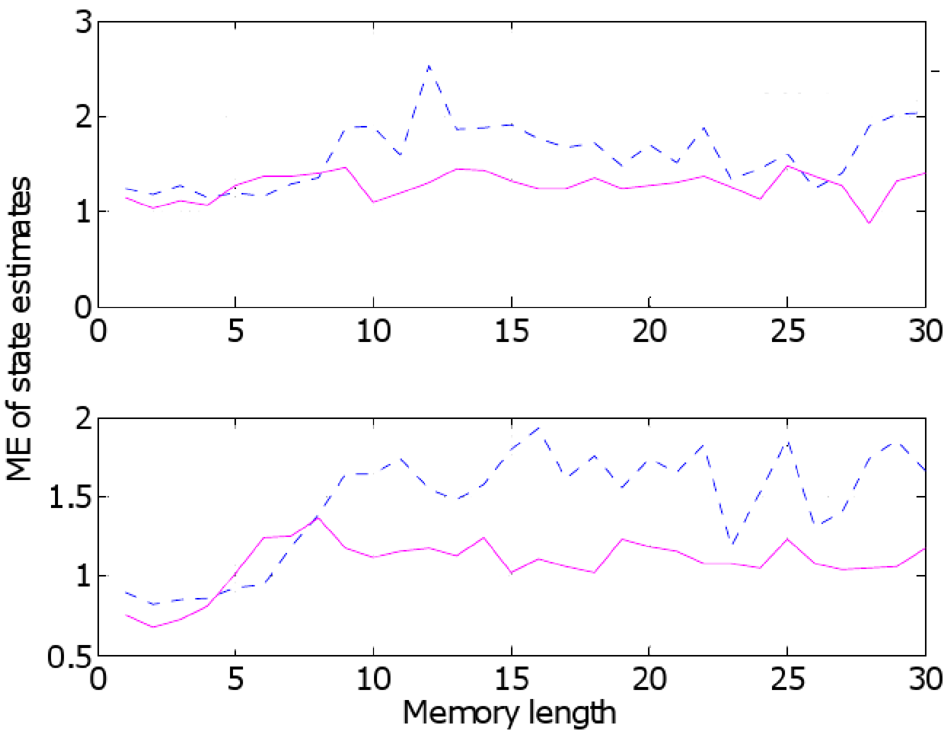

Figure 5.5: ME of the state estimates depending on memory length ∂ for all model matrices unknown (dashed line) and partially known model matrices (solid line)

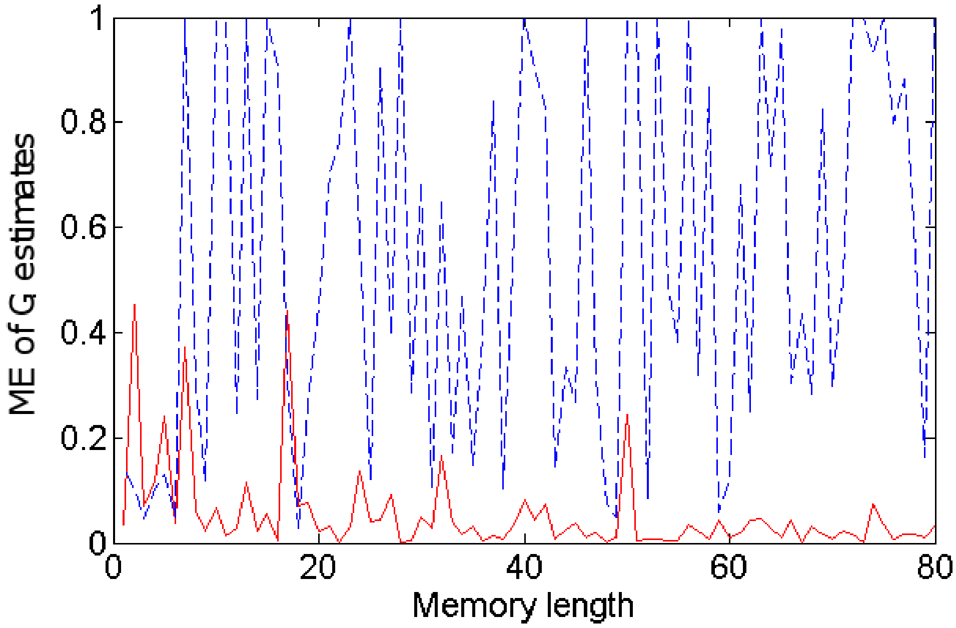

Figure 5.6: ME of ${}^{e}G$ estimates depending on memory length ∂ for all model matrices unknown (dashed line) and partially known model matrices (solid line)

5.2 Application to the traffic data

5.2.1 Data description

Here, the estimation algorithms are applied on the traffic data obtained from the Aimsun environment, see footnote on the page 47. Data describe a four-arm controlled intersection. The data set contains the occupancies, input and output intensities on the individual intersection arms, see Figure 4.1, for the whole workweek, i.e., five days. Time difference between two data items is 90 seconds. Total amount of the data entries is about 5000.

For the intersection specification, the LU model equations (4.8) and (4.9) are used with the model matrices (4.13) that can be divided into the estimated part (4.14) and known part (4.15).

We suppose that saturated flow S, green time z_t, turning rates α, parameter p are known or estimated outside of the LU model, see page 46.

Note that all above mentioned transportational quantities and constants are introduces in Section 4.1.

5.2.2 Joint estimation with partially known parameters

Here, we perform the on-line joint estimation of the states, i.e., queue length ξ and parameters κ, β, λ. Simultaneously, the parameter p is estimated using the newest possible estimate of ξ by the help of the function (4.12) and the estimate of p is subsequently used in the LU model. For the parameters α their off-line estimates are used.

Figure 5.7 illustrates the dependence of the ME (5.3) of the state estimates on the memory length ∂.

Figures 5.8 and 5.9 compare the state estimates for two different values of ∂. On these Figures, the segments with 100 time entries were chosen to illustrate whole course.

5.2.3 Discussion of the results

We suppose that the higher ME (5.3) of the state estimates occurs for the higher values of ∂ because of the changes in the system dynamics. The lesser ∂ corresponds to the higher level of the forgetting. The more is old information forgotten the more quickly can the estimation algorithm react on the changes as we can see on Figures 5.8 and 5.9.

In the experiment, the parameters κ, β, λ were estimated. The parameter p was figured out by the help of deterministic relation (4.12) on the basis of the last state estimate. The turning rates α weren't estimated in this experiment.

Comparing the mean value of the queue length ξ and the ME of its estimates, the ME reaches approximately 20% of the mean value ξ except of the 3rd arm. Here, the ME is around 50% of the mean value of ξ.

We suppose that the big error on the 3rd arm is caused mainly by the using of the off-line estimates of the turning rates α. We expect an improvement of the estimation after the introduction of the on-line estimates α.

We would like to point out that this example is only illustrative. We are going from the intersection description [23]. This description is currently under further development, see e.g. [24], but it is not finished yet. The main modification consists in the considering of the varying saturation flow S, in the modification of the relations for the occupancy O and in the including of the input intensity I into the state vector.

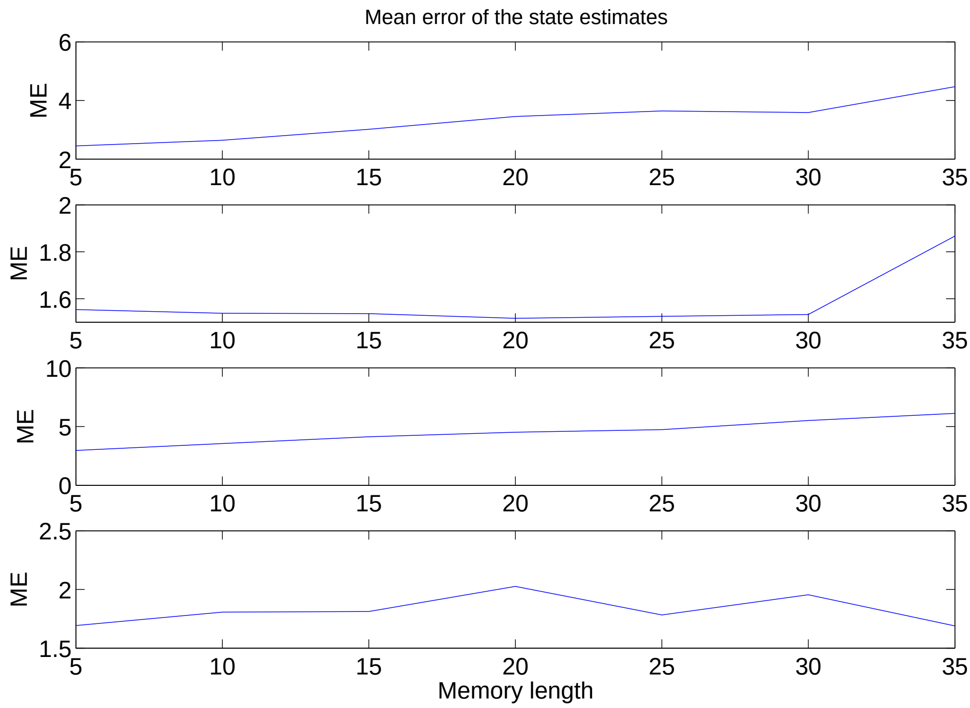

Figure 5.7: Mean errors of queue length estimates depending on memory length ∂

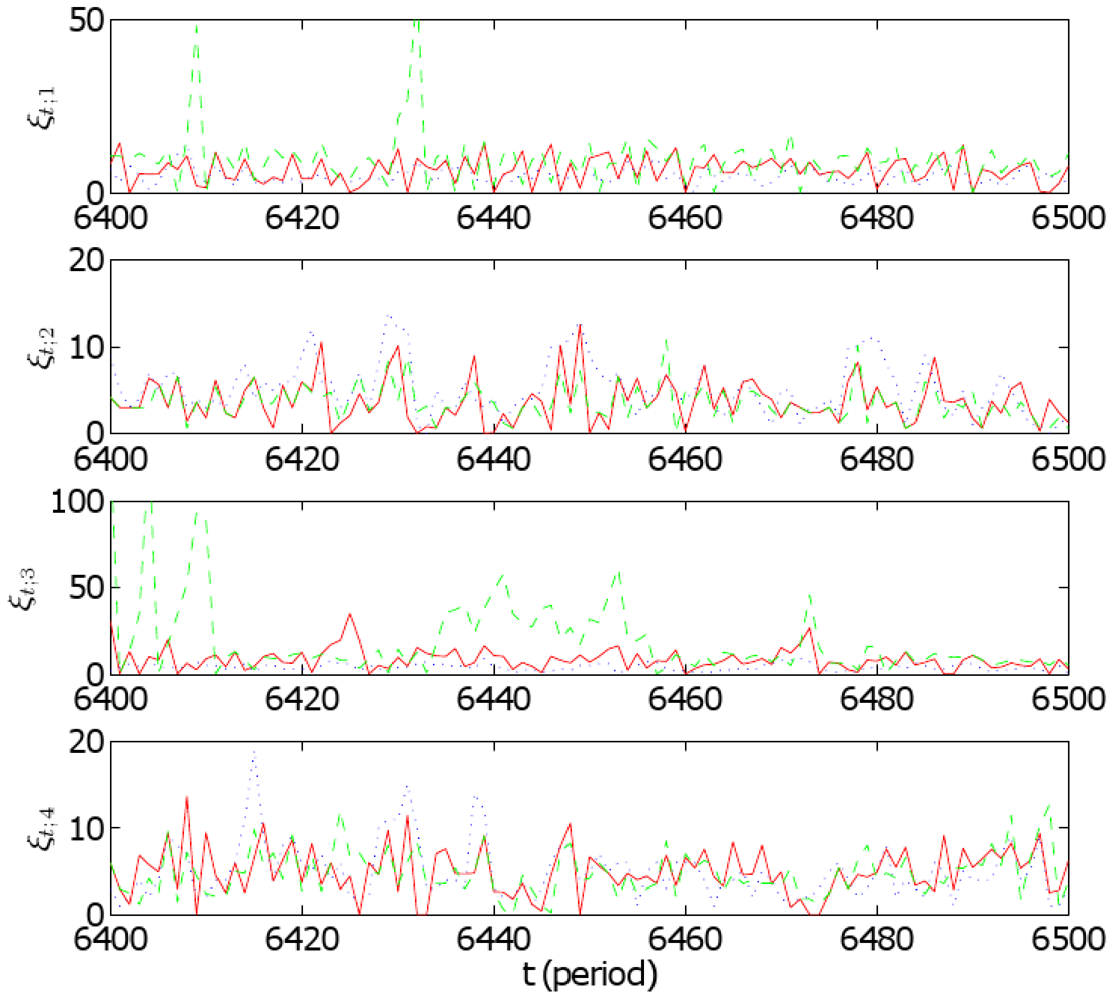

Figure 5.8: Simulated queues (dotted line), estimated queues with $\partial = 5$ (full line) and with $\partial = 35$ (dashed line)

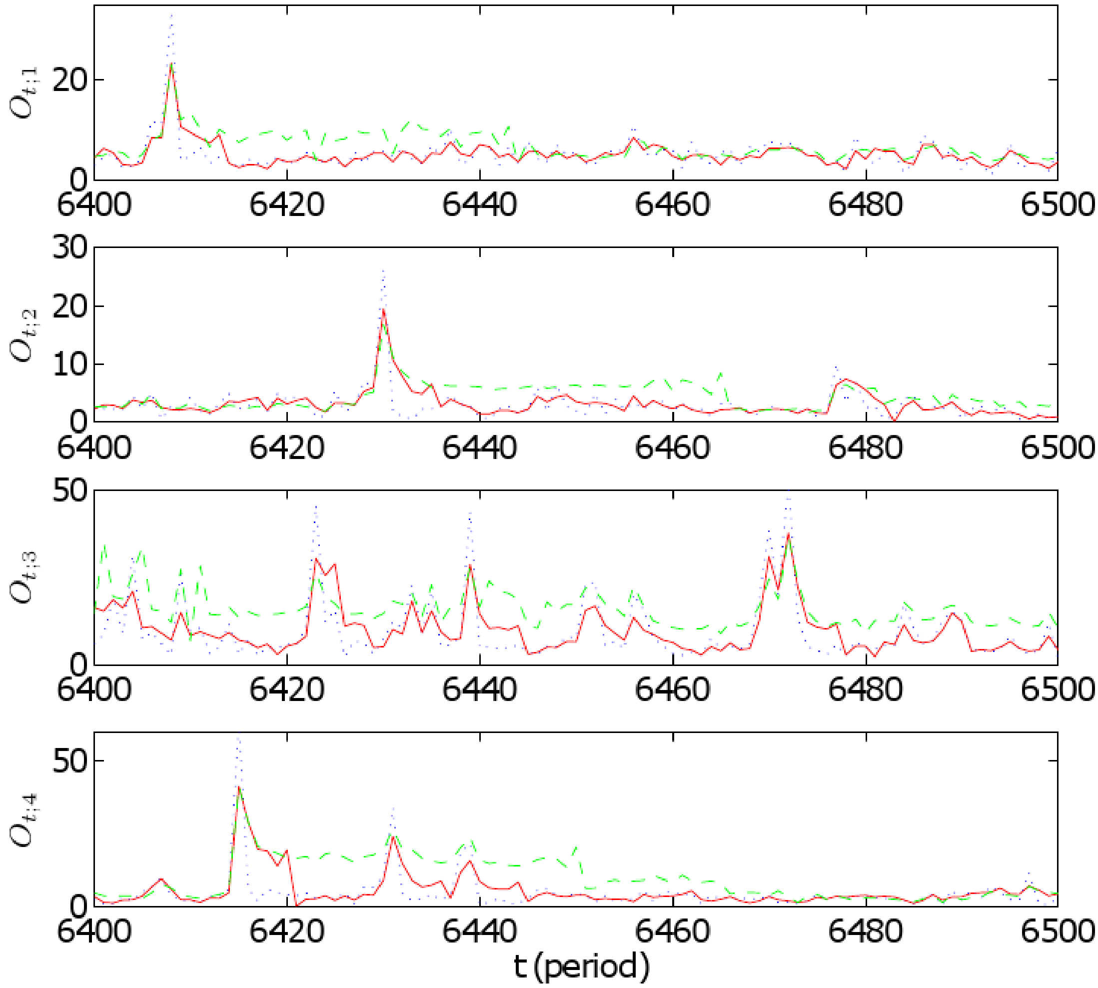

Figure 5.9: Simulated occupancies (dotted line), estimated occupancies with $\partial = 5$ (full line) and with $\partial = 35$ (dashed line)

Chapter 6

Conclusions

6.1 Results

In this thesis, the linear state model with uniform innovations (LU model) is proposed as an alternative to the Kalman filter. The algorithms are evaluated for the parameter estimation, state filtration and finally also for the joint parameter and state estimation including the estimation of the innovations ranges. For the on-line estimation, the approximation is proposed that keeps the computational feasibility of the algorithms in acceptable ranges. The functionality of the proposed algorithms is demonstrated on the illustrative examples, one of them with transportation data.

6.2 Contribution of the work

We see especially the following contributions of the thesis:

- proposed LU model is a promising alternative for the KF in the case of bounded data;
- compared with the "unknown-but-bounded" approaches, the LU model estimation keeps the advantages of the probabilistic approach;
- the produced Matlab algorithms are computationally feasible - the MAP estimation of the LU model converts into the problem of linear programming;
- LU model allows to respect hard, physically given, prior bounds on model parameters and states; the realistic hard bounds on the estimated quantities reduce the ambiguity of the model (arising from estimating a product of two unknowns) in the case of the joint parameter and state estimation;
- LU model respects natural bounds on stochastic disturbances and it allows estimation of the innovation ranges;

- the on-line algorithm updates estimates on the whole window of chosen length ∂;
- the off-line estimates of the innovations boundaries can be used for the initial setting of covariances for Kalman filtering;
- application on the traffic data is promising; we avoid, among others, the eventual negative estimates of the queue length that can occur by the use of the KF, caused by the unbounded support of the normal distribution;
- the proposed approach opens a way for on-line parameter and state estimation for a class of non-uniform distributions with restricted support as well as for Bayesian filtering of non-linear systems.

The actual contribution of the author

The development of the LU model and its estimation required many particular tasks. The most important subtasks made by the author are:

- embedding of the model with bounded innovations into the Bayesian framework
- discussion of the finite memory problem
- design of the approximate algorithm for the on-line joint estimation of the states and parameters
- conversion of the MAP estimation problem into the LP task
- design of the working algorithms in Matlab
- improving of the original traffic model by the continuous queue indicator

6.3 Future research

The future research will be oriented mainly on the further improving of the estimates quality.

Generally, the LU model estimation can be refined by:

- the method of selection of the inequalities for LP according to [25], see Subsection 6.3.1;
- the computation of the parameter estimates precision, see Subsection 6.3.2;
- the approximation of the non-uniform pdf by the uniform one, a principle of this method is outlined on the page 31; up to now we use "cutting off" method

- the use of the nonlinear programming that removes the linearization needed for the performing of the LP, see its description on the page 16.

Particularly, the LU model of the intersection can be improved by:

- the introduction of the model for the input intensity, see Subsection 6.3.3;

- by the on-line estimation of α, the possible way is suggested on the page 46.

6.3.1 Choice of the inequalities for LP

In the thesis, we use the method of the sliding window of the length ∂, i.e., $\partial + 1$ inequalities is taken into the LP. If the new data item is available, the oldest one is thrown away to keep the constant number of the inequalities. Sometimes, a newer data can be less informative than an older one. The method how to recognize the "worst" data is described in [25] and applied here on the autoregressive model. We aim to modify it for the state model.

6.3.2 Computation of the parameter estimates precision

Till now, we worked only with MAP estimates. We aim to refine our results also by an estimation of the precision of these point estimates. The original MAP estimation task gives result $\hat{X}$ for the posterior pdf $f(X|D)$. Now, we aim to find the approximative uniform pdf $\tilde{f}(X|D) = \mathcal{U}_X\left(\hat{X},\, {}^{X}R\right)$ so that the distance between f and $\tilde{f}$ is minimal. For this purpose, the Kullback-Leibler divergence [26] will be used that measures well proximity of a pair of pdfs.

6.3.3 Model of the input intensity

In the LU model of the intersection, we take the measured input intensity I_t as a known time-varying part of the model matrix. An introduction of the relation

$$I_t = H\ I_{t-1} + {}^{i}e_t$$

where H is matrix of the appropriate size, ${}^{i}e_t$ are the uniformly distributed innovations, changes the meaning of I_t. Then, I_t will play the role of a measurable system state.

6.4 Publications of the author

6.4.1 Publications relating to thesis

L. Pavelková, “Approximate on-line estimation of uniform state model with application on traffic data”, Tech. Rep. DAR 2008/2, ÚTIA AV ČR, Praha, 2008.

L. Pavelková, “Problem of state filtering in case of partially known system matrices”, in *Proceedings of the 9th International PhD Workshop Information Technologies & Control. Young Generation Viewpoint*, Ljubljana, October 1-3 2008, pp. 1–6.

M. Kárný and L. Pavelková, “Projection-based Bayesian recursive estimation of ARX model with uniform innovations”, *Systems & Control Letters*, vol. 56, no. 9/10, pp. 646–655, 2007.

L. Pavelková, M. Kárný, and V. Šmídl, “Towards Bayesian filtering on restricted support.”, in *Proceedings of NSSPW ’06. Cambridge*, Cambridge, October 4-8 2006, pp. 1–4, University of Cambridge.

L. Pavelková and M. Kárný, “Estimation of ARX model with uniform noise - algorithm and example.”, in *Proceedings of the 6th International PhD Workshop Information Technologies & Control. Young Generation Viewpoint*, Ljubljana, October 4-8 2005, pp. 1–6.

6.4.2 Other publications

I. Nagy, L. Pavelková, and K. Dedecius, “Partial forgetting in estimation of regression models”, Tech. Rep. 2200, ÚTIA AV ČR, Praha, 2007.

E. Suzdaleva, I. Nagy, L. Pavelková, and Jitka Homolová, “Edukalibre. guide”, Tech. Rep. 2145, ÚTIA AV ČR, Praha, October 2005.

E. Suzdaleva, I. Nagy, L. Pavelková, and M. Kárný, *Edukalibre Examples. (Program)*, ÚTIA AV ČR, Praha, 2005.

M. Novák and L. Pavelková, “Optimization of controller design using Monte-Carlo simulation”, in *Multiple Participant Decision Making. CMP’04*, J. Andrýsek, M. Kárný, and J. Kracík, Eds., Praha, May 2004, pp. 1–15, ÚTIA AV ČR.

M. Kárný, P. Nedoma, N. Khailova, and L. Pavelková, “Prior information in structure estimation”, *IEE Proceedings. Control Theory and Applications*, vol. 150, no. 6, pp. 643–653, 2003.

L. Pavelková, "Influence of prior knowledge on structure estimation. Abstract", in *Proceedings of the 4th International PhD Workshop Information Technologies & Control. Young Generation Viewpoint*, K. Belda, Ed., Praha, September 2003, p. 23, ÚTIA AV ČR.

L. Pavelková and I. Nagy, "Search for an Optimal Setup of Stabilized Forgetting in Estimation", Tech. Rep. 2089, ÚTIA AV ČR, Praha, 2003.

I. Nagy, P. Nedoma, M. Kárný, L. Pavelková, and P. Ettler, "O bayesovském učení", *AUTOMA*, , no. 7, pp. 56–60, 2002.

I. Nagy, P. Nedoma, M. Kárný, L. Pavelková, and P. Ettler, "Modelování chování složitých systémů pro podporu operátorů", *Automa*, vol. 8, no. 11, 12, pp. 54–57, 44–49, 2002.

L. Pavelková, "Webovské stránky oddělení AS", Tech. Rep. 2043, ÚTIA AV ČR, Praha, 2002.

M. Kárný, L. Pavelková, A. Halousková, and P. Nedoma, "Estimation of approximate Markov chains", in *Adaptive Systems in Control and Signal Processing. Preprints*, Cs. Bányász, Ed., Budapest, June 1995, pp. 317–322, IFAC.

L. Pavelková, "Approximate identification of Markov chains", in *Computer-Intensive Methods in Control and Signal Processing*, L. Kulhavá, M. Kárný, and K. Warwick, Eds., Praha, September 1994, pp. 335–340, ÚTIA AV ČR.

M. Kárný, A. Halousková, and L. Zörnigová, "Bayesian Pooling of Expert Opinions", Tech. Rep. 1799, ÚTIA AV ČR, Praha, 1994.

M. Kárný, A. Halousková, and L. Zörnigová, "On pooling expert opinions", in *10th IFAC Symposium on System Identification. Preprints*, M. Blanke and T. Söderström, Eds., Copenhagen, July 1994, pp. 477–478, IFAC.

L. Zörnigová, "Přibližná identifikace markovského řetězce", Tech. Rep. 1769, ÚTIA AV ČR, Praha, 1993.

6.4.3 Responses on the authors publications

Searching inside the databases "Scopus", "Web of Sciences" and by the help of scholar.google.com, we found the following responses to the paper [25]

K. Ramesh, N. Aziz and S. R. A. Shukor, "Development of NARX Model for Distillation Column and Studies on Effect of Regressors", Journal of Applied Sciences 8 (7): pp. 1214-1220, 2008.

K. Ramesh, S. R. A. Shukor and N. Aziz, "Development of sigmoidnet based NARX model for a distillation column", Chemical Product and Process Modeling 3 (2), art. no. 4, 2008.

Bibliography

[1] A.M. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[2] B. D. O. Anderson and S. B. Moore, *Optimal Filtering*, Englewood Cliffs, New Jersey: Prentice Hall Ins., 1979.

[3] V. Peterka, "Bayesian system identification", in *Trends and Progress in System Identification*, P. Eykhoff, Ed., pp. 239–304. Pergamon Press, Oxford, 1981.

[4] M. Kárný, J. Böhm, T. V. Guy, L. Jirsa, I. Nagy, P. Nedoma, and L. Tesař, *Optimized Bayesian Dynamic Advising: Theory and Algorithms*, Springer, London, 2005.

[5] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, London, 1987.

[6] P.E. Wellstead and M.B. Zarrop, *Self-tuning Systems*, John Wiley, Chichester, 1991.

[7] C Antoniou, M Ben-Akiva, and HN Koutsopoulos, "Online calibration of traffic prediction models", *TRAFFIC FLOW THEORY 2005*, , no. 1934, pp. 235 – 245, 2005.

[8] K.K. Srinivasan and Z.Y. Guo, "Day-to-day evolution of network flows under departure time dynamics in commuter decisions", *Travel Demand and Land Use 2003 Transportation Research Record*, vol. 1831, pp. 47–56, 2003.

[9] M. Green and D.J.N. Limebeer, *Linear robust control*, Prentice Hall, New Jersey, 1995, ISBN 0-13-102278-4.

[10] J. Roll, A. Nazin, and L. Ljung, "Nonlinear system identification via direct weight optimization", *Automatica*, vol. 41, no. 3, pp. 475–490, 2004.

[11] B.M. Ninness and G.C. Goodwin, "Rapprochment between bounded-error and stochastic estimation theory", *International Journal of Adaptive Control and Signal Processing*, vol. 9, no. 1, pp. 107–132, 1995.

[12] B.T. Polyak, S.A. Nazin, C. Durieu, and E. Walter, “Ellipsoidal parameter or state estimation under model uncertainty”, *Automatica*, vol. 40, no. 7, pp. 1171–1179, 2004.

[13] M. Milanese and G. Belforte, “Estimation theory and uncertainty intervals evaluation in presence of unknown but bounded errors linear families of models and estimators”, *IEEE Transactions on Automatic Control*, vol. 27, no. 2, pp. 408–414, 1982.

[14] A. Bemporad, C. Filippi, and F. Torrisi, “Inner and outer approximations of polytopes using boxes”, *Computational Geometry*, vol. 27, pp. 151–178, 2004.

[15] J.O. Berger, *Statistical Decision Theory and Bayesian Analysis*, Springer-Verlag, New York, 1985.

[16] K. Rektorys and spolupracovníci, *Přehled užité matematiky*, SNTL Praha, 1981.

[17] Z. Kotek and M. Razím, *Teorie nelineárních, optimálních a adaptivních řídících systémů*, ČVUT Praha, 1990.

[18] The Matlab Inc., *Matlab documentation*, vol. MA 01760-2098, Natick, 2000.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004, ISBN 0-521-83378-7.

[20] L. Pavelková, M. Kárný, and V. Šmídl, “Towards bayesian filtering on restricted support”, *internal publication of the project DAR, MŠMT ČR 1M0572*, 2006.

[21] S. Qin, W. Lin, and L. Ljung, “A novel subspace identification approach with enforced causal models”, *Automatica*, vol. 41, pp. 2043 – 2053, 2005.

[22] P. Li, R. Goodall, and V. Kadirkamanathan, “Estimation of parameters in a linear state space model using a Rao-Blackwellised particle filter”, *IEE Proceedings-control theory and applications*, vol. 151, pp. 728 – 738, 2004.

[23] Jitka Homolová and Ivan Nagy, “Model dopravní mikrooblasti”, *Automatizace*, vol. 47, no. 12, pp. 752 – 758, 2004, ISSN 0005-125X.

[24] P. Pecherková and J. Duník, “Modelling of traffic system with time-variant saturation flow.”, in *Proceedings of the 9th International PhD Workshop Information Technologies & Control. Young Generation Viewpoint*, Ljubljana, October 1-3 2008, pp. 1–6, Jozef Stefan Institute, accepted.

[25] M. Kárný and L. Pavelková, "Projection-based Bayesian recursive estimation of ARX model with uniform innovations", *Systems & Control Letters*, vol. 56, no. 9/10, pp. 646–655, 2007.

[26] S. Kullback and R. Leibler, "On information and sufficiency", *Annals of Mathematical Statistics*, vol. 22, pp. 79–87, 1951.

Appendix - Algorithms

```
function FacMem = facstate(stsize, ysize, usize, mem)
%
% build cell array of the state factors with uniform noise
%
% x_t  =  (A+Ae) x_{t-1}  +  (B+Be) u_t  +  (F+Fe) + ex_t,
%                  ex_t ~ (-rx,rx), rx is vector, rx_i > 0
%
% y_t  =  (C+Ce) x_t      +  (D+De) u_t  +  {G+Ge} + ey_t,
%                  ey_t ~ (-ry,rx), ry is vector, ry_j > 0
%
%     A   ... known possibly time-variant part
%     Ae  ... estimated time-invariant part
%     Ac = A + Ae ... complete model matrix
%
% FacMem = facstate(stsize, ysize, usize)...mem=0  ...only current values
% FacMem = facstate(stsize, ysize)       ...usize=1...scalar input
% FacMem = facstate(stsize)         ........ysize=1...scalar output
%
% stsize : size (length) of state vector
% ysize  : size (length) of output vector
% usize  : size (length) of input vector
% noise boundary  rx, ry
%
% Design : LP, June 2007
% Project: DAR

if nargin<4   % test of arguments
   mem=0;     % no memory, only current values
end
if nargin<3
   usize=1;   % scalar input
end
if nargin<2
   ysize=1;   % scalar output
end           % end of the test
```

```
% known parts of model matrices
A = zeros(stsize,stsize);
B = zeros(stsize,usize);
F = zeros(stsize,1);
C = zeros(ysize,stsize);
D = zeros(ysize,usize);
G = zeros(ysize,1);

% estimated parts of model matrices, unknown parameters
Ae = zeros(stsize,stsize);
Be = zeros(stsize,usize);
Fe = zeros(stsize,1);
Ce = zeros(ysize,stsize);
De = zeros(ysize,usize);
Ge = zeros(ysize,1);

% complete model matrices - used in predictor and simulator
Ac = zeros(stsize,stsize);
Bc = zeros(stsize,usize);
Fc = zeros(stsize,1);
Cc = zeros(ysize,stsize);
Dc = zeros(ysize,usize);
Gc = zeros(ysize,1);

% matrices of the indicators for the estimation
% known entries   ... 0
% unknown entries ... 1
Ai = zeros(stsize,stsize);
Bi = zeros(stsize,usize);
Fi = zeros(stsize,1);
Ci = zeros(ysize,stsize);
Di = zeros(ysize,usize);
Gi = zeros(ysize,1);

% half-width of innovation boundary
xr = 1e-6*ones(stsize,1);
yr = 1e-6*ones(ysize,1);

state=zeros(stsize,1);    % current state
state_old=zeros(stsize,1);% state in previous time instant

% build the resulting factor
Fac = struct('stsize',stsize,'ysize',ysize,'usize',usize,...
```

```
    'A',A,'B',B,'F',F,'C',C,'D',D,'G',G,...
    'Ae',Ae,'Be',Be,'Fe',Fe,'Ce',Ce,'De',De,'Ge',Ge,...
    'Ac',Ac,'Bc',Bc,'Fc',Fc,'Cc',Cc,'Dc',Dc,'Gc',Gc,...
    'Ai',Ai,'Bi',Bi,'Fi',Fi,'Ci',Ci,'Di',Di,'Gi',Gi,...
    'xr',xr,'yr',yr,'state',state,'mem',mem,'state_old',state_old);

% build the cell array of mem + 1 factors
FacMem=cell(1,mem+1); %  mem=0 ... only current value is stored
    for i=1:mem+1
       FacMem{i}=Fac;
    end
```

```
function  [Sim,y,x_new] = facsimulust(Sim,u,x_new)
%
% simulation of state factor with uniform noise
% Sim = faxsimulust(Sim,u,x_new)
%
% Sim     : simulated state factor with uniform noise
%   u     : current input
%   x_new : current state
%
% Sim     : factor with updated state vector
%
% nargin==2 ... state and output computation
% nargin==3 ... only output computation with given state
%
% Design: LP, July 2006
% Project: DAR

global TIME DATA SEED

SEED = rand('seed');           % seed of noise generator
rand('seed',SEED);             % set the seed

if nargin<2
     error(' Too little input arguments! ');
end
if isempty(TIME), error('TIME is not defined'); end
if isempty(DATA), error('DATA are not defined'); end

A=Sim{1}.Ac; B=Sim{1}.Bc; F=Sim{1}.Fc;
C=Sim{1}.Cc; D=Sim{1}.Dc; G=Sim{1}.Gc;
xr = Sim{1}.xr;
yr = Sim{1}.yr;
x=Sim{1}.state; % past state

% new data and state items
if nargin==2 % x_new is not given
    x_new = A*x     + B*u + F + xr*(rand-0.5);
end
    y     = C*x_new + D*u + G + yr*(rand-0.5);

%state update
Sim{1}.state=x_new;
```

```
function [FacMem,xest,rxest,ryest] = facupdtust(FacMem,lbx,ubx,ubrx,ubry)
% update of statistics for the state model with uniform noise
% state and noise boundaries estimation
% for vectors u,y
%
% x_t  =  A_t x_{t-1} +  B_t u_t  +  F_t + ex_t,
%             ex_t ~ (-rx,rx), rx is vector, rx_i > 0
%
% y_t  =  C_t x_t     +  D_t u_t  +  G_t + ey_t,
%             ey_t ~ (-ry,rx), ry is vector, ry_j > 0
%
% [FacMem,xest,rxest,ryest] = facupdtust(FacMem,lbx,ubx,ubrx,ubry)
%
% FacMem  : state factor with uniform noise, memory mem
% lbx     : lower boundaries of the estimated state
% ubx     : upper boundaries of the estimated state
% ubrx    : upper boundary for the state noise
% ubry    : upper boundary for the output noise
%
% FacMem  : state factor with unif. noise updated by current data
% xest    : state vector [x(TIME), ..., x(TIME-mem)] estimate
% rxest   : state noise estimate
% ryest   : output noise estimate
%
% Design  : LP, March 2007
% Project : DAR
%
global DATA TIME EF
ysize=FacMem{1}.ysize;        % output vector size
ychns=1:ysize;                % output channels
usize=FacMem{1}.usize;        % input vector size
uchns=ysize+1:ysize+usize;    % input channels
stsize=FacMem{1}.stsize;      % state vector size
mem=min(FacMem{1}.mem,TIME-1);%

if nargin < 5      % default for output noise boundary
    ubry=2*ones(ysize,1);
end
if nargin < 4      % default for state noise boundary
    ubrx=2*ones(stsize,1);
end
if nargin < 3      % default for state upper boundary
    ubx=Inf*ones((mem+1)*stsize,1);
end
```

```
if nargin < 2      % default for state lower boundary
    lbx=-Inf*ones((mem+1)*stsize,1);
end
lbx=lbx(1:(mem+1)*stsize);
ubx=ubx(1:(mem+1)*stsize);
% check of the boundaries
if size(lbx,1)~=(mem+1)*stsize
    error('Incorrect lower boundaries for the estimated state')
end
if size(ubx,1)~=(mem+1)*stsize
    error('Incorrect upper boundaries for the estimated state')
end
if lbx>=ubx
    error('Lower boundary exceeds upper boundary')
end

% LP task - search for xx=[x_t', ... , x_(t-mem)', xr', yr']'
%
% minimize kriterion
%           min JJ=ff*xx= sum rx_i + sum ry_j
% under conditions
% AA*xx<=bb
%
% construction of the matrix AA
%       | AA11  AA12 |
%  AA = |            |
%       | AA21  AA22 |
%

% AA11 ... size=[2*(mem+1)*stsize, (mem+1)*stsize];
K=[1;-1]; % constant
A=FacMem{1}.Ac;
M1=kron(eye(mem+1),kron(K,eye(stsize)));
M2=kron(eye(mem+1),kron(K,A));
M3=lcol(M2,stsize);
M4=M3(:,1:size(M2,2));
AA11=M1-M4;

% AA12 ... size[2*(mem+1)*stsize, stsize + ysize];
AA12=-kron(ones(2*(mem+1),1),rcol(eye(stsize),ysize));

% AA21 ... size=[2*(mem+1)*ysize, (mem+1)*stsize];
C=FacMem{1}.Cc; % only for constant C
AA21=kron(eye(mem+1),kron(K,C));
```

```
% AA22 ... size=[2*(mem+1)*ysize, stsize + ysize];
AA22=-kron(ones(2*(mem+1),1),lcol(eye(ysize),stsize));

AA=[AA11 AA12;AA21 AA22];

% construction of the vector bb:  bb=bb1+bb2

% bb1 ... size=[2*(mem+1)*stsize,1];
bb1=[];
for i=1:mem
   B=FacMem{i}.Bc; F=FacMem{i}.Fc;u=DATA(uchns,TIME-i+1);
   bb1=[bb1;kron(K,B*u+F)];
end
%last row: i=mem+1
i=mem+1;
A=FacMem{i}.Ac; B=FacMem{i}.Bc; F=FacMem{i}.Fc;
u=DATA(uchns,TIME-i+1); x=FacMem{i}.state_old;
bb1=[bb1;kron(K,A*x+B*u+F)];

% bb2 ... size=[2*(mem+1)*ysize,1];
bb2=[];
for i=1:mem+1
   D=FacMem{i}.Dc; G=FacMem{i}.Gc;
   u=DATA(uchns,TIME-i+1);y=DATA(ychns,TIME-i+1);
   bb2=[bb2;kron(K,y-D*u-G)];
end

% bb ... size=[2*(mem+1)*(stsize + ysize),1];
bb=[bb1;bb2];

% xx
sizexx=[(mem+1)*stsize+stsize+ysize,1];
xx=zeros(sizexx);

% ff
ff=zeros(size(xx));
% ones on the place with rx, ry
ff((mem+1)*stsize+1:size(xx,1),1)=1;

% LP
exitflag=-1;
while exitflag<=0
   % boundaries for the xx
```

```
   lbX=[lbx;zeros(stsize,1);zeros(ysize,1)];
   ubX=[ubx;ubrx;ubry];
   % LP computation
   options = optimset('Display','off');
   [xx,fval,exitflag]=linprog(ff,AA,bb,[],[],lbX,ubX,[],options);
   EF=[EF, [exitflag; TIME]];
   % increasing of noise boundaries in the case of LP failure
   ubrx=1.5*ubrx; ubry=1.5*ubry;
end
% transformation of the vector xx
% of the size (mem*stsize+stsize+ysize,1)
% to the state set xest(stsize,mem+1) - without rx,ry
% and rxest, ryest with noises estimated
%
% in xx are states ordered by decreasing time (t, t-1, ..., 1)
% change of the time course - xx: t->t-mem, xest: t-mem -> t
xest=[];
for i=1:stsize:(mem+1)*stsize
   xestcol=xx(i:i+stsize-1,1);
   xest=[xestcol,xest];
end
rxest=xx((mem+1)*stsize+1:(mem+1)*stsize+stsize,1);
ryest=xx((mem+1)*stsize+stsize+1:max(size(xx)),1);

% update of FacMem
    for i=1:mem+1                 % save the old state
        FacMem{i}.state_old=FacMem{i}.state;
    end
% update of Fac - state shift
    for i=mem+1:-1:2              % shift of state values
        FacMem{i}.state=FacMem{i-1}.state;
    end
    FacMem{1}.state = xx(1:stsize);
    FacMem{1}.xr    = rxest;
    FacMem{1}.yr    = ryest;
```

```
function [FacMem,Aest,Best,Fest,Cest,Dest,Gest,rxest,ryest] = ...
                          facupdtupar(FacMem,lbpar,ubpar,ubrx,ubry)
%
% update of statistics for the state model with uniform noise
% model parameter and noise boundaries estimation
% for vectors u,y
%
% FacMem     : state factor
%     x_t  =  (A+Ae) x_{t-1}+  (B+Be) u_t  +  (F+Fe) + ex_t,
%                      ex_t ~ (-rx,rx), rx is vector, rx_i > 0
%
%     y_t  =  (C+Ce) x_t    +  (D+De) u_t  +  {G+Ge} + ey_t,
%                       ey_t ~ (-ry,rx), ry is vector, ry_j > 0
%
%     A  ... known possibly time-variant part
%     Ae ... estimated time-invariant part
%
% lbpar      : lower boundary on parameters
% ubpar      : upper boundary on parameters
% ubrx       : upper boundary of the state noise
% ubry       : upper boundary of the output noise
%
% FacMem       : st. factor with unif. noise updated with current data
% Aest,Best,...: estimates of model parameters
% rxest        : state noise estimate
% ryest        : output noise estimate
% EF           : store exitflags from LP
%
% Design  : LP, June 2007
% Project : DAR

global DATA TIME EF
ysize=FacMem{1}.ysize;      % output vector size
ychns=1:ysize;              % output channels
usize=FacMem{1}.usize;      % input vector size
uchns=ysize+1:ysize+usize;  % input channels
stsize=FacMem{1}.stsize;    % state vector size

if nargin < 5
    ubry=2*ones(ysize,1);   % default for output noise boundary
end
if nargin < 4
 ubrx=2*ones(stsize,1);  % default for state noise boundary
end
```

```
if nargin < 3 % default for parameters upper boundary
 ubpar=struct('ubA',[],'ubB',[],'ubF',[],'ubC',[],'ubD',[],'ubG',[]);
end
if nargin < 2 % default for parameters lower boundary
 lbpar=struct('lbA',[],'lbB',[],'lbF',[],'lbC',[],'lbD',[],'lbG',[]);
end

Ae=FacMem{1}.Ae; Be=FacMem{1}.Be; Fe=FacMem{1}.Fe;
Ce=FacMem{1}.Ce; De=FacMem{1}.De; Ge=FacMem{1}.Ge;

Ai=FacMem{1}.Ai; Bi=FacMem{1}.Bi; Fi=FacMem{1}.Fi;
Ci=FacMem{1}.Ci; Di=FacMem{1}.Di; Gi=FacMem{1}.Gi;

xr=FacMem{1}.xr; yr=FacMem{1}.yr;

% parameter boundaries
lbA=lbpar.lbA; lbB=lbpar.lbB; lbF=lbpar.lbF;
lbC=lbpar.lbC; lbD=lbpar.lbD; lbG=lbpar.lbG;% lower boundaries
ubA=ubpar.ubA; ubB=ubpar.ubB; ubF=ubpar.ubF;
ubC=ubpar.ubC; ubD=ubpar.ubD; ubG=ubpar.ubG;% upper boundaries

mem=min(FacMem{1}.mem,TIME-1);   % memory

% sequence of the state vectors [x(t-mem), ..., x(t)]
State=zeros(stsize,mem+1);
for i=0:mem
    State(:,i+1)=FacMem{mem+1-i}.state; % t-mem, ..., t-1, t
end
State=[FacMem{mem+1}.state_old State];% state x_{t-mem-1} added

% LP task: search for xx' containing Ae,Be,Fe,Ce,De,Ge,rx,ry
% matrices rearranged into vectors
%
% minimize kriterion
%          min JJ=ff*xx= sum rx_i + sum ry_j
% under conditions
% AA*xx<=bb
%
% construction of the matrix AA
%
%       | AA11  AA12 AA13 |
%  AA = |                 |
%       | AA21  AA22 AA23 |
```

```
K=[1;-1]; % constant

% AA11 ... size=[2*(mem+1)*stsize, stsize*stsize+stsize*usize+stsize];
AA11=[];
if mem==0
    AA11a=zeros(2*stsize,stsize*stsize);                        % A part
    AA11b=kron(eye(stsize),kron( K,DATA(uchns,1)' ));          % B part
    AA11f=kron(eye(stsize),kron( K,1));                         % F part
    AA11=[AA11;AA11a, AA11b AA11f];
else
    for i=1:mem+1
        AA11a=kron(K,kron(eye(stsize),State(:,mem+2-i)'));      % A part
        AA11b=kron(K,kron(eye(stsize),DATA(uchns,TIME+1-i)')); % B part
        AA11f=kron(K,kron(eye(stsize),1));                      % F part
        AA11=[AA11;AA11a, AA11b AA11f];
    end
end

% AA12 ... size=[2*(mem+1)*stsize, ysize*stsize+ysize*usize+ysize*1];
AA12=zeros(size(AA11,1),ysize*stsize+ysize*usize+ysize*1);

% AA13 ... size=[2*(mem+1)*stsize, stsize+ysize];
AA13=-kron(ones(2*(max(1,mem+1)),1),rcol(eye(stsize),ysize));

% AA21 ... size=[2*(mem+1)*ysize, stsize*stsize+stsize*usize+stsize];
 AA21=zeros(2*(mem+1)*ysize, stsize*stsize+stsize*usize + stsize*1);

% AA22 ... size=[2*(mem+1)*ysize, ysize*stsize+ysize*usize+ysize*1];
AA22=[];
for i=0:mem
AA22c=kron(eye(ysize),kron( K,State(:,mem+2-i)' ));     % C part
AA22d=kron(eye(ysize),kron( K,DATA(uchns,TIME-i)' ));   % D part
AA22g=kron(eye(ysize),kron( K,1));                      % G part
AA22=[AA22; AA22c,AA22d,AA22g];
end

% AA23 ... size=[2*(mem+1)*ysize, stsize+ysize];
AA23=-kron(ones(2*(mem+1),1),lcol(eye(ysize),stsize));

AA=[AA11,AA12,AA13;AA21,AA22,AA23];

% construction of the vector bb:  bb=bb1+bb2
% bb1 ... size=[2*(mem+1)*stsize,1];
bb1=[];
```

```
if mem==0, bb1=zeros(2*stsize,1);
else
  for i=1:mem+1
  A=FacMem{i}.A; B=FacMem{i}.B; F=FacMem{i}.F;
  bb1_=State(:,mem-i+3)-A*State(:,mem-i+2)...
             -B*DATA(uchns,(TIME-i+2))-F;
  bb1_=kron(K,bb1_);
  bb1=[bb1;bb1_];
  end
end

% bb2 ... size=[2*(mem+1)*ysize,1];
bb2=[];
for i=1:mem+1
  C=FacMem{i}.C; D=FacMem{i}.D; G=FacMem{i}.G;
  bb2_=DATA(ychns,(TIME-i+1))-C*State(:,mem-i+2)...
              -D*DATA(uchns,(TIME-i+1))-G;
  bb2=[bb2; bb2_];
end
Kii=kron(eye((mem+1)),kron(K,eye(ysize)));
bb2=Kii*bb2;

bb=[bb1;bb2];

% xx
sizexx=[size(AA,2),1];
xx=zeros(sizexx);

% ff
ff=zeros(size(xx));
% ones on the place of rx, ry
ff((stsize+ysize)*(stsize+usize+1)+1:size(xx,1),1)=1;

% boundary construction
lbAt=lbA';lbBt=lbB';lbFt=lbF';lbCt=lbC';lbDt=lbD';lbGt=lbG';
ubAt=ubA';ubBt=ubB';ubFt=ubF';ubCt=ubC';ubDt=ubD';ubGt=ubG';

% LP
exitflag=0;
while exitflag<=0;
    lbX=[lbAt(:);lbBt(:);lbFt(:);lbCt(:);lbDt(:);...
            lbGt(:);zeros(stsize,1);zeros(ysize,1)];
    ubX=[ubAt(:);ubBt(:);ubFt(:);ubCt(:);ubDt(:);...
                                 ubGt(:);ubrx;ubry];
```

```
    options = optimset('Display','off');
    [xx,fval,exitflag]=linprog(ff,AA,bb,[],[],lbX,ubX,[],options);
    EF=[EF, [exitflag; TIME]];
    % increasing of noise boundaries in the case of LP failure
    ubrx=1.5*ubrx; ubry=1.5*ubry;
end

% transformation of the vector xx
Aestcol=xx(1:stsize^2,1);
maxim=length(Aestcol);
Aest=zeros(stsize,stsize);
for i=1:stsize
    Aest(i,:)=Aestcol((i-1)*stsize+1:(i-1)*stsize+stsize)';
end;

Bestcol=xx(maxim+1:maxim+stsize*usize,1);
maxim=maxim+length(Bestcol);
Best=zeros(stsize,usize);
for i=1:stsize
    Best(i,:)=Bestcol((i-1)*usize+1:(i-1)*usize+usize)';
end;

Fest=xx(maxim+1:maxim+stsize,1);
maxim=maxim+length(Fest);

Cestcol=xx(maxim+1:maxim+ysize*stsize,1);
maxim=maxim+length(Cestcol);
Cest=zeros(ysize,stsize);
for i=1:ysize
    Cest(i,:)=Cestcol((i-1)*stsize+1:(i-1)*stsize+stsize)';
end;

Destcol=xx(maxim+1:maxim+ysize*usize,1);
maxim=maxim+length(Destcol);
Dest=zeros(ysize,usize);
for i=1:ysize
    Dest(i,:)=Destcol((i-1)*usize+1:(i-1)*usize+usize)';
end;

Gest=xx(maxim+1:maxim+ysize,1);
maxim=maxim+length(Gest);
rxest=xx(maxim+1:maxim+stsize,1);
maxim=maxim+length(rxest);
ryest=xx(maxim+1:maxim+ysize,1);
```

```
maxim=maxim+length(ryest);

% update of FacMem
% shift of the older values
for i=mem+1:-1:2
    FacMem{i}.Ae=FacMem{i-1}.Ae;FacMem{i}.Be=FacMem{i-1}.Be;
    FacMem{i}.Fe=FacMem{i-1}.Fe;
    FacMem{i}.Ce=FacMem{i-1}.Ce;FacMem{i}.De=FacMem{i-1}.De;
    FacMem{i}.Ge=FacMem{i-1}.Ge;
    FacMem{i}.xr=FacMem{i-1}.xr;FacMem{i}.yr=FacMem{i-1}.yr;
end
% current estimates
FacMem{1}.Ae=Aest; FacMem{1}.Be=Best; FacMem{1}.Fe=Fest;
FacMem{1}.Ce=Cest; FacMem{1}.De=Dest; FacMem{1}.Ge=Gest;
FacMem{1}.rx=rxest; FacMem{1}.ry=ryest;

% for the time invariant known part
FacMem{1}.Ac=FacMem{1}.A+Aest; FacMem{1}.Bc=FacMem{1}.B+Best;
FacMem{1}.Fc=FacMem{1}.F+Fest;
FacMem{1}.Cc=FacMem{1}.C+Cest; FacMem{1}.Dc=FacMem{1}.D+Dest;
FacMem{1}.Gc=FacMem{1}.G+Gest;
```

```
function [FacMem,xest,Aest,Best,Fest,Cest,Dest,Gest,rxest,ryest]=...
                    facupdtall(FacMem,lbx,ubx,lbpar,ubpar,ubrx,ubry)
%
% update of statistics
% for the linear state-space model with uniform innovations
%
%     x_t  =  (A+Ae) x_{t-1} + (B+Be) u_t + (F+Fe) + ex_t,
%                   ex_t ~ (-rx,rx), rx is vector, rx_i > 0
%
%     y_t  =  (C+Ce) x_t     + (D+De) u_t + {G+Ge} + ey_t,
%                   ey_t ~ (-ry,rx), ry is vector, ry_j > 0
%
%     A  ... known possibly time-variant part
%     Ae ... estimated time-invariant part
%
% state estimation
% model parameter  estimation
% noise boundaries estimation
%
% FacMem              : state factor with uniform noise
% lbx                 : lower boundaries of the estimated state
% ubx                 : upper boundaries of the estimated state
% lbpar               : lower boundary on parameters
% ubpar               : upper boundary on parameters
% ubrx                : upper boundary of the state noise
% ubry                : upper boundary of the output noise
%
% FacMem              : st. factor with unif. noise updated by current data
% xest                : state estimate
% Aest,Best,Cest,Dest
% Fest, Gest          : model parameters estimate
% rxest               : state noise estimate
% ryest               : output noise estimate
%
% Design  : LP, May 2008
% Project : DAR

global DATA TIME EF

ysize=FacMem{1}.ysize;     % output vector size
ychns=1:ysize;             % output channels
usize=FacMem{1}.usize;     % input vector size
uchns=ysize+1:ysize+usize; % input channels
stsize=FacMem{1}.stsize;   % state vector size
```

```
mem=min(FacMem{1}.mem,TIME-1); % memory length

% default values of the boundaries:
if nargin<7, ubry=2*ones(ysize,1); end % output noise
if nargin<6, ubrx=2*ones(stsize,1);end % state noise
if nargin<5, ubpar=[]; end             % parameters
if nargin<4, lbpar=[];end
if nargin < 3
    ubx=Inf*ones((mem+1)*stsize,1);    % states
end
if nargin < 2
    lbx=-Inf*ones((mem+1)*stsize,1);
end

lbx=lbx(1:(mem+1)*stsize);       % boundaries on states
ubx=ubx(1:(mem+1)*stsize);       % according to actual memory length
if lbx>ubx
    error('Lower boundary exceeds upper boundary')
end

% parameter boundaries
lbA=lbpar.lbA;                  % lower boundaries
lbB=lbpar.lbB;
lbF=lbpar.lbF;
lbC=lbpar.lbC;
lbD=lbpar.lbD;
lbG=lbpar.lbG;
ubA=ubpar.ubA;                   % upper boundaries
ubB=ubpar.ubB;
ubF=ubpar.ubF;
ubC=ubpar.ubC;
ubD=ubpar.ubD;
ubG=ubpar.ubG;

% indicators of the estimated parts of model matrices
Ai=FacMem{1}.Ai; Bi=FacMem{1}.Bi; Fi=FacMem{1}.Fi;
Ci=FacMem{1}.Ci; Di=FacMem{1}.Di; Gi=FacMem{1}.Gi;

% LP task formulation: Search for
% xx'=[x_t, ... x_(t-mem), A_col, B_col, ...
%           F_col, C_col, D_col, G_col, rx, ry]
% with matrices rearranged into column vectors
% which minimize the criterion
```

```
%           min JJ=ff*xx= sum rx_i + sum ry_j
%
% under conditions
% AA*xx<=bb
% AAeq*xx=bbeq
% lb <= xx <= ub
%
% Construction of the matrix AA:
%
%       | AA11  AA12 AA13 AA14 |
%  AA = |                      |
%       | AA21  AA22 AA23 AA24 |

% AA11 ... size=[2*(mem+1)*stsize, (mem+1)*stsize];
K=[1;-1]; % constant
Ac=FacMem{1}.Ae+FacMem{1}.A;
M1=kron(eye(mem+1),kron(K,eye(stsize)));
M2=kron(eye(mem+1),kron(K,Ac));
M3=lcol(M2,stsize);
M4=M3(:,1:size(M2,2));
AA11=M1-M4;

% AA12 ... for A,B,F
% size=[2*(mem+1)*stsize, stsize*stsize+stsize*usize+stsize];
AA12=[];
for i=1:mem+1 % choice of the estimated parts of parameters
    MA=selallzeros(Ai,FacMem{i}.state);
    MB=selallzeros(Bi,DATA(uchns,TIME+1-i));
    MF=selallzeros(Fi,1);
    AA12a=kron(-K,MA);    % A part
    AA12b=kron(-K,MB);    % B part
    AA12f=kron(-K,MF);    % F part
    AA12=[AA12;AA12a, AA12b AA12f];
end

% AA13
AA13=zeros(size(AA11,1),ysize*stsize+ysize*usize+ysize*1);

% AA14 ... size=[2*(mem+1)*stsize, stsize+ysize];
AA14=-kron(ones(2*(max(1,mem+1)),1),rcol(eye(stsize),ysize));

% AA21 ... size=[2*(mem+1)*ysize, (mem+1)*stsize];
Cc=FacMem{1}.Ce+FacMem{1}.C; % only for constant C
AA21=kron(eye(mem+1),kron(K,Cc));
```

```
% AA22 ... size=[2*(mem+1)*ysize, stsize*(stsize+usize+1)];
AA22=zeros(2*(mem+1)*ysize, stsize*(stsize+usize + 1));

% AA23 ... size=[2*(mem+1)*ysize, ysize*(stsize+usize+1)];
AA23=[];
% first row with predicted state
Bc=FacMem{1}.Be+FacMem{1}.B;
Fc=FacMem{1}.Fe+FacMem{1}.F;
% the newest state prediction
xpred=Ac*FacMem{1}.state+ Bc*DATA(uchns,TIME)'+ Fc;
% row for estimated x_t ...  prediction
AA23_ =[selallzeros(Ci,xpred),selallzeros(Di,DATA(uchns,TIME)),...
                                                selallzeros(Gi,1);
      -selallzeros(Ci,xpred),-selallzeros(Di,DATA(uchns,TIME)),...
                                              -selallzeros(Gi,1)];
for i=1:mem  % choice of the estimated parts of parameters
    MC=selallzeros(Ci,FacMem{i}.state);
    MD=selallzeros(Di,DATA(uchns,TIME-i));
    MG=selallzeros(Gi,1);
    AA23c=kron(K,MC);      % C part
    AA23d=kron(K,MD);      % D part
    AA23g=kron(K,MG);      % G part
    AA23=[AA23; AA23c,AA23d,AA23g];
end
AA23=[AA23_;AA23];

% AA24 ... size=[2*(mem+1)*ysize, stsize+ysize];
AA24=-kron(ones(2*(mem+1),1),lcol(eye(ysize),stsize));

AA=[AA11,AA12,AA13,AA14;AA21,AA22,AA23,AA24];

% construction of the vector bb:  bb=bb1+bb2
bb1=[];
Ae=FacMem{1}.Ae;
for i=1:mem
  B=FacMem{i}.B; F=FacMem{i}.F;
  bb1=[bb1;kron(K,-Ae*FacMem{i}.state+B*DATA(uchns,TIME+1-i)+F)];
end
%last row
i=mem+1;
A=FacMem{i}.A; B=FacMem{i}.B; F=FacMem{i}.F;
u=DATA(uchns,TIME-i+1); x=FacMem{i}.state;
bb1=[bb1; kron(K,A*x + B*u + F)];
```

```
% bb2
Ce=FacMem{1}.Ce; D=FacMem{1}.D; G=FacMem{1}.G;
%first item with state prediction
bb2=[Ce*xpred+DATA(ychns,TIME)-D*DATA(uchns,TIME)-G;
    -Ce*xpred-DATA(ychns,TIME)+D*DATA(uchns,TIME)+G];
for i=1:mem
    bb2a=[Ce*FacMem{i}.state+DATA(ychns,TIME-i)-D*DATA(uchns,TIME-i)-G;
         -Ce*FacMem{i}.state-DATA(ychns,TIME-i)+D*DATA(uchns,TIME-i)+G];
    bb2=[bb2;bb2a];
end
bb=[bb1;bb2];

% xx
sizexx=[(mem+1)*stsize + (stsize+ysize)*(stsize+usize+2),1];
xx=zeros(sizexx);
% ff
ff=zeros(size(xx));

for i=0:stsize+ysize-1
    ff(length(ff)-i)=1; % ones on the place of rx, ry
end

% boundaries for LP
boundstlow= lbx;              % lower boundary on states
boundstup = ubx;              % upper boundary on states
lbAt=lbA';boundAlow = lbAt(:);% lower boundaries on parameters
lbBt=lbB';boundBlow = lbBt(:);
lbFt=lbF';boundFlow = lbFt(:);
lbCt=lbC';boundClow = lbCt(:);
lbDt=lbD';boundDlow = lbDt(:);
lbGt=lbG';boundGlow = lbGt(:);
ubAt=ubA';boundAup  = ubAt(:); % upper boundaries on parameters
ubBt=ubB';boundBup  = ubBt(:);
ubFt=ubF';boundFup  = ubFt(:);
ubCt=ubC';boundCup  = ubCt(:);
ubDt=ubD';boundDup  = ubDt(:);
ubGt=ubG';boundGup  = ubGt(:);
boundrxlow = zeros(stsize,1);  % lower boundary on state noise
boundrxup = ubrx;              % upper boundary on state noise
boundrylow = zeros(ysize,1);   % lower boundary on output noise
boundryup =  ubry;             % upper boundary on output noise

boundxxlow = [boundstlow; boundAlow; boundBlow;  boundFlow; ...
```

```
      boundClow; boundDlow;  boundGlow; boundrxlow; boundrylow];
boundxxup  = [boundstup;  boundAup;  boundBup;   boundFup; ...
       boundCup;  boundDup;   boundGup;  boundrxup;  boundryup];
exitflag = -1;
con=1.5;
% for unsuccessful LP repeating with increased noise boundaries
while(exitflag<0)
    options = optimset('Display','off');
    [xx,fval,exitflag]=...
           linprog(ff,AA,bb,[],[],boundxxlow, boundxxup,[],options);
    EF=[EF, [exitflag; TIME]];
    ubrx=con*ubrx; ubry=con*ubry;
    boundxxup  = [boundstup;  boundAup;  boundBup;  boundFup; ...
                    boundCup;  boundDup;  boundGup;  ubrx; ubry];
end
% decomposition of xx into particular est. states and parameters
bound1=(mem+1)*stsize;       % boundary for x_t, ... , x_(t-mem)
bound2=bound1+stsize*stsize; % boundary for A
bound3=bound2+stsize*usize;  % boundary for B
bound4=bound3+stsize;        % boundary for F
bound5=bound4+ysize*stsize;  % boundary for C
bound6=bound5+ysize*usize;   % boundary for D
bound7=bound6+ysize;         % boundary for G
bound8=bound7+stsize;        % boundary for rx
if bound8+ysize~=length(xx),error('Error in xx decomposition'),end
% states estimates
xest=[];
% change of the time course:
% xx(1:bound1): ((mem+1)*stize,1) - decreasing time
% => xest: (stsize,mem+1) - increasing time
for i=1:stsize:bound1
    xestcol=xx(i:i+stsize-1,1);
    xest=[xestcol,xest];
end
% param. A estimate
Aestcol=xx(bound1+1:bound2);
for i=1:stsize
    Aest(i,:)=Aestcol((i-1)*stsize+1:i*stsize);
end;
% param. B estimate
Bestcol =xx(bound2+1:bound3);
for i=1:stsize
    Best(i,:)=Bestcol((i-1)*usize+1:i*usize);
end;
```

```
% param. F estimate
Fest =xx(bound3+1:bound4);     % param. F estimate
% param. C estimate
Cestcol=xx(bound4+1:bound5);
for i=1:ysize
    Cest(i,:)=Cestcol((i-1)*stsize+1:i*stsize);
end;
% param. D estimate
Destcol =xx(bound5+1:bound6);
for i=1:ysize
    Dest(i,:)=Destcol((i-1)*usize+1:i*usize);
end;
% param. G estimate
Gest =xx(bound6+1:bound7);     % param. G estimate

rxest=xx(bound7+1:bound8);     % rx estimate
ryest=xx(bound8+1:length(xx)); % ry estimate

% update of FacMem
for i=1:mem+1                  % save the old state
    FacMem{i}.state_old=FacMem{i}.state;
end
for i=mem+1:-1:2               % shift of factor values
    FacMem{i}.state=  FacMem{i-1}.state;
    FacMem{i}.xr    = FacMem{i-1}.xr;
    FacMem{i}.yr    = FacMem{i-1}.yr;
    FacMem{i}.Ae    = FacMem{i-1}.Ae;
    FacMem{i}.Be    = FacMem{i-1}.Be;
    FacMem{i}.Fe    = FacMem{i-1}.Fe;
    FacMem{i}.Ce    = FacMem{i-1}.Ce;
    FacMem{i}.De    = FacMem{i-1}.De;
    FacMem{i}.Ge    = FacMem{i-1}.Ge;
end
% update of newest factor
FacMem{1}.state = xx(1:stsize);
FacMem{1}.xr    = rxest;
FacMem{1}.yr    = ryest;
FacMem{1}.Ae    = Aest;
FacMem{1}.Be    = Best;
FacMem{1}.Fe    = Fest;
FacMem{1}.Ce    = Cest;
FacMem{1}.De    = Dest;
FacMem{1}.Ge    = Gest;
end
```

```
function M = rcol(M,col)
% extension of the (a,b) matrix M
% by the (a,col) zero matrix Z (from the right)
% M -> [M,Z]
%
% M = rcol(M,col)
%
% M      : matrix of the size (a,b)
% col    : number of columns of the extension matrix Z
%
% M      : extended matrix [M,Z]
%
% Design  : LP
% Updated : July, 2006
% Project : DAR

% global DATA TIME
if nargin < 2
    error('This function requires two input arguments')
end

if col < 0
    error('Number of the columns must be positive')
end

rowM=size(M,1);
M=horzcat(M,zeros(rowM,col));
```

```
function M = lcol(M,col)
% extension of the (a,b) matrix M
% by the (a,col) zero matrix Z (from the left)
% M -> [Z,M]
%
% M = lcol(M,col)
%
% M      : matrix of the size (a,b)
% col    : number of columns of the extension matrix Z
%
%
% M      : extended matrix [Z,M]
%
% Design  : LP
% Updated : July, 2006
% Project : DAR

% global DATA TIME
if nargin < 2
    error('This function requires two input arguments')
end

if col < 0
    error('Number of the columns must be positive')
end

rowM=size(M,1);
M=horzcat(zeros(rowM,col),M);
```

Index