Recursive Parameter Estimation for Nonlinear Continuous-Time Systems through Sensitivity-Model-Based Adaptive Filters

Dissertation submitted to the Department of Electrical Engineering and Information Sciences Ruhr-Universität Bochum for the degree of Doktor-Ingenieur

by
Christian Bohn
Hamburg

Bochum 2000
Dissertation submitted on 15 May 2000
Oral examination held on 5 July 2000

Referees:    Prof. Dr.-Ing. H. Unbehauen (dissertation supervisor)
             Prof. Dr.-Ing. habil. M. Pandit
Contents

Preface ........................................................................................................................................ v
Acknowledgements .................................................................................................................. vii
Nomenclature ............................................................................................................................ ix
List of Tables ................................................................................................................................ xvii
1 Introduction ................................................................................................................ .. 1
   1.1 Filtering and Parameter Estimation ............................................................................ 1
   1.2 Recursive Parameter Estimation for State-Space Models ....................................... 3
   1.3 Dissertation Outline .................................................................................................. 7
2 Mathematical Preliminaries ................................................................................................. 11
   2.1 Notation .................................................................................................................... 11
   2.2 Kronecker Products and Stacking Operations ......................................................... 12
   2.3 Matrix Differential Calculus ................................................................................... 13
   2.4 Gauss-Newton Algorithm for a Least-Squares Cost Functional ............................ 16
   2.5 Gauss-Newton Algorithm for a Maximum-Likelihood Cost Functional ............ 19
3 Approximate Nonlinear Filters ............................................................................................ 23
   3.1 General Filtering Algorithm ................................................................................... 24
   3.2 Rederivation of Higher-Order Terms ...................................................................... 26
4 Adaptive Filtering and Recursive Parameter Estimation .................................................... 31
   4.1 Introduction and Chapter Overview ........................................................................ 31
   4.2 State Augmentation ................................................................................................... 33
   4.3 Sensitivity-Based Approaches to Parameter Estimation ........................................... 36
   4.4 Adaptive Constant-Gain Filtering .......................................................................... 38
   4.5 Adaptive Extended Kalman Filtering ...................................................................... 42
   4.6 Adaptive Higher-Order Filters ............................................................................... 48
   4.7 Discussion of the Cost Functional and Related Approximations ...................... 53
   4.8 Relationship between State Augmentation and the Sensitivity Model ............... 54
To begin with, a warning to readers is in order: the following pages contain a lot of equations, some of which may appear dreadful and ridiculous at first sight, for example, Equation (4.123) or, maybe even worse, Equation (B.6). These equations involve Kronecker products, stacking operators, and matrix derivatives which might look confusing to readers who have not familiarized themselves with these operations. Two arguments are put forward in defense of presenting such equations.

First, writing the equations using vector and matrix derivatives instead of elementwise derivatives ensures a higher degree of interpretability, particularly for more complicated terms. A simple example illustrates this. Consider the second-order Taylor expansion of a vector-valued function \( f \) in the neighborhood of the point \( x \). This can be written as

\[
f(x + \Delta x) \equiv f(x) + \frac{df}{dx} \Delta x + b.
\]

The second-order term \( b \) can be expressed in compact notation as

\[
b = \frac{1}{2} \frac{d^2 f}{dx^T dx} \Delta x \otimes_2,
\]

where the superscript " \( \otimes_2 \) " stands for the second Kronecker power (see Appendix A), or in elementwise notation as

\[
b_i = \frac{1}{2} \sum_{j,k} \frac{d^2 f_{ij}}{dx_j dx_k} \Delta x_j \Delta x_k,
\]

or

\[
b_i = \frac{1}{2} \text{trace} \left( \frac{d^2 f}{dx dx^T} \Delta x \Delta x^T \right).
\]

Readers who agree with the author that beauty lies in the first, compact expression (which except for the symbol " \( \otimes_2 \) " replacing the standard second power and the boldface type resembles the corresponding scalar Taylor expansion) will appreciate the application of matrix differential calculus; the others have no choice but getting used to it if they want to continue reading.

Secondly, since the author claims that the algorithms presented in this dissertation are in "implementation-ready" form, these equations should be viewed accordingly. With Kronecker
products and array operations readily available in mathematical packages such as Matlab, even Equation (4.123) can be coded as one line. Furthermore, both the derivation and the implementation of these expressions are extremely “error-intolerant.” It is the author’s experience that if after several pages of pencil-and-paper computations one still ends up with matching dimensions, the computations are almost certainly correct. Since no indices are used, indexing mistakes in the coding process (such as using an “i” instead of a “j”) do not occur. Mistakes that do not lead to incorrect dimensions are extremely unlikely.

Readers who still are not convinced of these benefits can be assured of the fact that most of the derivations would look even worse if elementwise operations were applied (taking the derivative of the term $M_{ij}$ in Equation [3.5] with respect to a parameter $p$, where $x$ and $P$ are also functions of $p$ – and then implementing the resulting expression – would be an excellent exercise for these critical minds).

A variety of algorithms are presented in this dissertation. The derivation of each algorithm is included in the text. All nonstandard mathematical tools used in these derivations are reviewed in Chapter 2 and Appendix A. The equations for each algorithm, including all auxiliary terms, are usually summarized in a table; a list of tables follows after the nomenclature section. The reader who does not want to go through the details of the derivations can thus directly refer to the corresponding table for a specific algorithm. Since to the best of the author’s knowledge, all algorithms relevant to the problem addressed in this dissertation are included, it is the author’s hope that these tables and the review of the mathematical preliminaries make this dissertation a valuable reference.
Acknowledgements

This dissertation is the product of my research activities from 1994 to 2000 at the Control Engineering Laboratory, Ruhr-Universität Bochum, under the supervision and guidance of Professor Heinz Unbehauen.

I am greatly indebted to Professor Derek Atherton who first triggered my interest in control engineering during my stay as an ERASMUS scholar at the University of Sussex at Brighton in 1992/93. Working with him laid the foundations for my further research endeavors.

Further, I express my deep gratitude to Professor Heinz Unbehauen for giving me the opportunity to work with him and for providing me with continuous support and advice in scientific as well as personal matters. Without his encouragement, this work would not have materialized.

My thanks also go to Professor Pandit, Universität Kaiserslautern, who first expressed an interest in my work when we met at the Conference on Decision and Control in 1999. I appreciate his willingness to act as a referee for this dissertation.

Over the years, each one of my colleagues has contributed to this work in his or her unique way. This holds for fruitful scientific discussions as well as support in the daily matters of a researcher’s life. Particularly, I would like to thank the senior members of the Control Engineering Laboratory: Dr Johannes Dastych, Dr Rainer Pickhardt and Professor Christian Schmid. Their years of experience in control engineering were an invaluable resource. Further thanks goes to Dr Adrian Gambier, who I shared an office with for five years, Dr Thomas Junge, Dipl.-Ing. Torsten Knohl and Dr Herbert Werner for their insightful comments and criticism. Technical assistance was always readily provided by Dipl.-Ing. Udo Wieser.

Finally, I want to thank my parents Harri and Ilse Bohn and my in-laws John and Judy Schneller for their love and encouragement. Without the commitment and dedication of my loving wife Jill Schneller I would not have succeeded, and without her painstaking editing I could not have written this dissertation in English.

Bochum, July 2000

Christian Bohn
Nomenclature

Acronyms

CGEKF Constant-Gain Extended Kalman Filter
EKF Extended Kalman Filter
FBF First-Order Bias-Corrected Filter
GSF Modified Gaussian Second-Order Filter
INSPEC Information Services for the Physics and Engineering Communities (database of the Institute of Electrical Engineers)
LS Least Squares
ML Maximum Likelihood
PC Personal Computer
RPE Recursive Prediction Error Algorithm/Method
SM Sensitivity Model
TSF Truncated Second-Order Filter

General Notation

\( \alpha, \beta, \chi, ... \) Scalars
\( a, b, c, ... \) Scalars
\( A, B, C, ... \) Scalars
\( \mathbf{a}, \mathbf{b}, \mathbf{c}, ... \) Vectors
\( \mathbf{A}, \mathbf{B}, \mathbf{C}, ... \) Matrices
\( A^{(nm)} \) A matrix of dimension \( n \) by \( m \)
\( x_i \) The \( i \)th element of the vector \( \mathbf{x} \)
\( A_{kl} \) The element in the \( k,l \) position of the matrix \( \mathbf{A} \)

Constants

\( \mathbf{E}_{ij}^{(k\times d)} \) A matrix with a one in the \( i, j \) position and zeroes everywhere else
\( g \) Acceleration of gravity in Section 6.7
\( \mathbf{I}, \mathbf{I}_n \) Identity matrix (of dimension \( n \) by \( n \))
\( j \) Imaginary unit \( (\sqrt{-1}) \)
\( \mathbf{T}_i^{(n)} \) Duplication matrix (dimension \( n^2 \) by \( n(n+1)/2 \)), defined in Section 2.2
NOMENCLATURE

\( \mathbf{T}_r^{(n)} \) Reduction matrix (dimension \( n(n+1)/2 \) by \( n^2 \)), defined in Section 2.2

\( \mathbf{U}_{kl} \) Kronecker permutation matrix (dimension \( kl \) by \( kl \)), defined in Section 2.2

\( \mathbf{U}_{kl} \) Self-derivative matrix (dimension \( k^2 \) by \( P \)), defined in Rule 6 in Appendix A

\( \mathbf{0} \) Zero matrix or vector

Variables

\( A \) Prediction error covariance matrix (dimension \( m \) by \( m \)); also a general weighting matrix in Section 2.4

\( b_m \) Bias-correction term in the measurement equation (dimension \( m \) by 1)

\( b_p \) Bias-correction term in the propagation equation (dimension \( n \) by 1)

\( d \) Damper constant for the example in Section 6.7

\( D \) Viscous damping ratio for the dc motor model in Section 6.6

\( \frac{\partial \mathbf{b}_m}{\partial \mathbf{p}} \) Derivative of the bias-correction term in the output prediction equation (dimension \( m \) by \( s \))

\( \frac{\partial \mathbf{b}_m^T}{\partial \mathbf{p}^T} \) Derivative of the bias-correction term in the output prediction equation (dimension 1 by \( ms \))

\( \frac{\partial \mathbf{b}_p}{\partial \mathbf{p}^T} \) Derivative of the bias-correction term in the propagation equation (dimension \( n \) by \( s \))

\( \frac{\partial \mathbf{K}}{\partial (\text{col } \mathbf{K})^T} \) Derivative of the filter gain estimate with respect to its own column vector (dimension \( n \) by \( nm^2 \))

\( \frac{\partial \mathbf{K}}{\partial \mathbf{p}^T} \) Derivative of the filter gain with respect to the parameter vector (dimension \( n \) by \( ms \))

\( \frac{\partial \mathbf{K}}{\partial \mathbf{p}^T} \) Derivative of the filter gain estimate with respect to the parameter vector (dimension \( n \) by \( ms \))

\( \frac{\partial \mathbf{M}}{\partial \mathbf{p}^T}, \frac{\partial \mathbf{M}_{TSF}}{\partial \mathbf{p}^T}, \frac{\partial \mathbf{M}_{GSF}}{\partial \mathbf{p}^T} \) Derivatives of the correction terms for the approximate prediction error covariance matrix, for the TSF and the GSF (dimension \( m \) by \( ms \))

\( \frac{\partial \mathbf{P}_{xx}}{\partial \mathbf{p}^T} \) Sensitivity matrix of the estimation error covariance matrix (dimension \( n \) by \( ns \))

\( \frac{\partial \mathbf{c}}{\partial \mathbf{P}_{xx}} \) Sensitivity vector of the reduced column vector of the estimation error covariance matrix (dimension \( n(n+1)s/2 \) by 1)

\( \frac{\partial \hat{\mathbf{x}}}{\partial (\text{col } \mathbf{K})^T} \) Sensitivity of the state estimate with respect to the filter gain estimate (dimension \( n \) by \( nm \))
### NOMENCLATURE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{dx}{dp}$</td>
<td>Sensitivity matrix of the state estimate (dimension $n$ by $s$)</td>
</tr>
<tr>
<td>$\frac{dx}{dp}$</td>
<td>Sensitivity vector of the state estimate (dimension $ns$ by 1)</td>
</tr>
<tr>
<td>$\frac{dy^T}{d(\text{col} \ K)}$</td>
<td>Gradient of the predicted output with respect to the estimated filter gain (dimension $nm$ by $m$)</td>
</tr>
<tr>
<td>$\frac{dy^T}{dp}$</td>
<td>Gradient of the predicted output with respect to the parameter vector (dimension $s$ by $m$)</td>
</tr>
<tr>
<td>$\frac{dy^T}{\hat{p}}$</td>
<td>Gradient of the predicted output with respect to the augmented parameter vector (dimension $s+nm$ by $m$)</td>
</tr>
<tr>
<td>$\Delta \hat{p}$</td>
<td>Parameter estimate correction</td>
</tr>
<tr>
<td>$\Delta \hat{x}$</td>
<td>State estimate correction</td>
</tr>
<tr>
<td>$e$</td>
<td>Prediction error (dimension $m$ by 1)</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Term in the gradient of the maximum-likelihood cost functional in Section 2.5 (dimension $s$ by 1)</td>
</tr>
<tr>
<td>$\psi$</td>
<td>Magnetic flux for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Stochastic approximation gain in Section 2.4</td>
</tr>
<tr>
<td>$i_a, i_f$</td>
<td>Armature current and field current for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$J$</td>
<td>Moment of inertia for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$k$</td>
<td>Index for the discrete time variable $t_k$; also the gain of the linear system in Section 6.5; also a constant in the dc motor example in Section 6.6; also the spring constant for the example in Section 6.7</td>
</tr>
<tr>
<td>$K$</td>
<td>Input signal gain for the example in Section 6.7</td>
</tr>
<tr>
<td>$K, \bar{K}$</td>
<td>Filter gain matrix (dimension $n$ by $m$)</td>
</tr>
<tr>
<td>$L$</td>
<td>Parameter adaptation gain matrix (dimension $s$ by $m$)</td>
</tr>
<tr>
<td>$L^*$</td>
<td>Auxiliary gain matrix (dimension $s$ by $m$)</td>
</tr>
<tr>
<td>$L_a$</td>
<td>Armature inductivity for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Forgetting factor</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of outputs of a dynamic system; also the deviation of the mass from its steady-state value for the example in Section 6.7</td>
</tr>
<tr>
<td>$M, M_{\text{TSF}}, M_{\text{GSF}}$</td>
<td>Correction terms for the prediction error covariance matrix, for the TSF and the GSF (dimension $m$ by $m$)</td>
</tr>
<tr>
<td>$m_0$</td>
<td>Steady-state mass for the example in Section 6.7</td>
</tr>
<tr>
<td>$\dot{m}_{in}$</td>
<td>Stream of incoming material for the example in Section 6.7</td>
</tr>
</tbody>
</table>
\( \dot{m}_{\text{in,0}} \) Steady-state value of the stream of incoming material for the example in Section 6.7

\( \dot{m}_{\text{out}} \) Stream of material leaving the container for the example in Section 6.7

\( \mu \) Parameter of the Van der Pol oscillator in Section 6.3

\( \mu \) Term in the gradient of the maximum-likelihood cost functional in Section 2.5 (dimension \( s \) by 1); also the argument of the moment-generation function in Chapter 3 (dimension \( n \) by 1)

\( n \) Dimension (number of states) of a dynamic system

\( N \) Matrix of fourth moments for a Gaussian distribution in Chapter 3 (dimension \( n^2 \) by \( n^2 \))

\( \nu \) Parameter of the Van der Pol oscillator in Section 6.3

\( t \) Continuous time variable

\( t_k \) Discrete time variable

\( t_k^-, t_k^+ \) Time instants immediately before (–) and after (+) the discrete time instant \( t_k \)

\( \omega \) Parameter of the Van der Pol oscillator in Section 6.3; also angular speed for the dc motor model in Section 6.6

\( \omega_0 \) Natural frequency of the oscillatory system in Section 6.5

\( p \) Parameter vector of a dynamic system (dimension \( s \) by 1)

\( \hat{p} \) Estimate of the parameter vector of a dynamic system (dimension \( s \) by 1)

\( \tilde{p}_i \) Auxiliary parameters for a reparameterization in Sections 6.5 and 6.7

\( \tilde{p} \) Augmented parameter vector for the adaptive CGEKF (dimension \( nm s \) by 1)

\( P, \overline{P}, \overline{P}^* \) Covariance matrices

\( P_{xx}, P_{xp}, P_{pp}, \overline{P}_{xx} \) Estimation error (cross)covariance matrices (dimensions \( n \) by \( n \), \( n \) by \( s \), and \( s \) by \( s \))

\( q \) Outlet constant for the example in Section 6.7

\( q_1, q_2 \) Flows into tank 1 and 2 for the three-tank system

\( q_{13}, q_{32} \) Flow from tank 1 to tank 3 and flow from tank 3 to tank 2 for the three-tank system

\( q_{20} \) Combined flow through the outlets of tank 2 for the three-tank system

\( Q \) Process noise covariance matrix (dimension \( n \) by \( n \))

\( Q_{pp} \) Parameter pseudonoise covariance matrix (dimension \( s \) by \( s \))

\( r \) Number of inputs of a dynamic system
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>Measurement noise covariance matrix (dimension $m \times m$); also the normalized information matrix in Sections 2.4 and 2.5</td>
</tr>
<tr>
<td>$R_a$</td>
<td>Armature resistance for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$s$</td>
<td>Number of system parameters</td>
</tr>
<tr>
<td>$S$, $\bar{S}$</td>
<td>Weighting matrices for the parameter adaptation gain (dimension $m \times m$)</td>
</tr>
<tr>
<td>$S^*$</td>
<td>Auxiliary weighting term</td>
</tr>
<tr>
<td>$S_x$</td>
<td>Approximate state sensitivity in the state-augmented EKF (dimension $n \times s$)</td>
</tr>
<tr>
<td>$S_y$</td>
<td>Approximate transposed gradient of the predicted output in the state-augmented EKF (dimension $m \times s$)</td>
</tr>
<tr>
<td>$T$</td>
<td>Sampling time</td>
</tr>
<tr>
<td>$u$</td>
<td>Input signal of a dynamic system (dimension $r \times 1$)</td>
</tr>
<tr>
<td>$\bar{u}$</td>
<td>Auxiliary input signal for the example in Section 6.7</td>
</tr>
<tr>
<td>$v$</td>
<td>Velocity of the incoming material for the example in Section 6.7</td>
</tr>
<tr>
<td>$\tilde{v}$</td>
<td>Augmented process noise for the state-augmented EKF (dimension $n+s \times 1$)</td>
</tr>
<tr>
<td>$v_a$</td>
<td>Armature voltage for the dc motor model in Section 6.6</td>
</tr>
<tr>
<td>$v_p$</td>
<td>Parameter pseudonoise (dimension $s \times 1$)</td>
</tr>
<tr>
<td>$w$</td>
<td>Measurement noise (dimension $m \times 1$)</td>
</tr>
<tr>
<td>$x$</td>
<td>State vector of a dynamic system (dimension $n \times 1$)</td>
</tr>
<tr>
<td>$\hat{x}$</td>
<td>Estimate of the state vector of a dynamic system (dimension $n \times 1$)</td>
</tr>
<tr>
<td>$\tilde{x}$</td>
<td>Augmented state vector for the state-augmented EKF (dimension $n+s \times 1$)</td>
</tr>
<tr>
<td>$\xi$</td>
<td>Damping coefficient of the oscillatory system in Section 6.5</td>
</tr>
<tr>
<td>$y$</td>
<td>Output signal of a dynamic system (dimension $m \times 1$)</td>
</tr>
<tr>
<td>$\hat{y}$</td>
<td>Predicted output of dynamic system (dimension $m \times 1$)</td>
</tr>
</tbody>
</table>
Functions and Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adj $A$</td>
<td>Adjoint of the matrix $A$</td>
</tr>
<tr>
<td>$c$</td>
<td>Reduced column operator, defined in Section 2.2</td>
</tr>
<tr>
<td>$\text{col}$</td>
<td>Column operator, defined in Section 2.2</td>
</tr>
<tr>
<td>$\det A$</td>
<td>Determinant of the matrix $A$</td>
</tr>
<tr>
<td>$\frac{d}{dM} A$</td>
<td>Derivative of the matrix function $A$ with respect to the matrix $M$, defined in Equation (2.11)</td>
</tr>
<tr>
<td>$\frac{\partial}{\partial M} A$</td>
<td>Partial derivative of the matrix function $A$ with respect to its argument $M$, defined in Equation (2.13)</td>
</tr>
<tr>
<td>$\delta(t)$</td>
<td>Dirac impulse function</td>
</tr>
<tr>
<td>$\delta_{ij}$</td>
<td>Kronecker delta function</td>
</tr>
<tr>
<td>$\mathbb{E}$</td>
<td>Expectation operator</td>
</tr>
<tr>
<td>$\exp$</td>
<td>Exponential function</td>
</tr>
<tr>
<td>$f$</td>
<td>State transition function of a dynamic system (dimension $n$ by 1)</td>
</tr>
<tr>
<td>$\tilde{f}$</td>
<td>Augmented state transition function of a dynamic system (dimension $n+s$ by 1)</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Moment-generating function</td>
</tr>
<tr>
<td>$g$</td>
<td>Static input nonlinearity</td>
</tr>
<tr>
<td>$h$</td>
<td>Measurement function of a dynamic system (dimension $m$ by 1)</td>
</tr>
<tr>
<td>$\tilde{h}$</td>
<td>Measurement function of a dynamic system expressed in $\tilde{x}$ instead of $x$ and $p$ (dimension $m$ by 1)</td>
</tr>
<tr>
<td>$J$</td>
<td>Cost functional</td>
</tr>
<tr>
<td>$J_{\text{LS}}$</td>
<td>Least-squares cost functional</td>
</tr>
<tr>
<td>$J_{\text{ML}}$</td>
<td>Maximum-likelihood cost functional</td>
</tr>
<tr>
<td>$\text{lim}$</td>
<td>Limit operator</td>
</tr>
<tr>
<td>$\ln$</td>
<td>Natural logarithm</td>
</tr>
<tr>
<td>$\text{mat}_{\text{row}}$</td>
<td>Matrix operator, defined in Section 2.2</td>
</tr>
<tr>
<td>$\text{row}$</td>
<td>Row operator, defined in Section 2.2</td>
</tr>
<tr>
<td>$\hat{\sigma}^2(x)$</td>
<td>Sample variance of $x$</td>
</tr>
<tr>
<td>$\tanh$</td>
<td>Hyperbolic tangent</td>
</tr>
<tr>
<td>$\tilde{T}_i$</td>
<td>Chebyshev polynomials with the absolute terms removed, defined in Section 6.5</td>
</tr>
<tr>
<td>$T_i'$</td>
<td>Shifted Chebyshev polynomials with the absolute terms removed, defined in Section 6.6</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>trace $A$</td>
<td>Trace of the matrix $A$</td>
</tr>
<tr>
<td>$A^T$, $x^T$</td>
<td>Transpose of the matrix $A$ (the vector $x$)</td>
</tr>
<tr>
<td>$\dot{x}$</td>
<td>Time derivative of $x$</td>
</tr>
<tr>
<td>$A^{-1}$</td>
<td>Inverse of the matrix $A$</td>
</tr>
<tr>
<td>$\otimes$</td>
<td>Kronecker product, defined in Section 2.2</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>Kronecker sum, defined in Section 2.2</td>
</tr>
<tr>
<td>$p^{\otimes,i}$</td>
<td>The $i$th Kronecker power of the vector $p$, defined in Rule 5 in Appendix A</td>
</tr>
<tr>
<td>$||$</td>
<td>(Any) matrix or vector norm</td>
</tr>
</tbody>
</table>
List of Tables

Table 1.1 Approaches to recursive parameter estimation for stochastic state-space models ... 6
Table 2.1 Recursive Gauss-Newton algorithm for a least-squares cost functional .......... 19
Table 2.2 Recursive Gauss-Newton algorithm for a maximum-likelihood cost functional... 21
Table 3.1 General filtering algorithm ..................................................................................... 25
Table 3.2 Additional terms for the higher-order filters .......................................................... 29
Table 4.1 State-augmented Extended Kalman Filter .............................................................. 35
Table 4.2 Adaptive Constant-Gain Extended Kalman Filter.................................................. 41
Table 4.3 Adaptive Extended Kalman Filter .......................................................................... 46
Table 4.4 Modifications for higher-order filters..................................................................... 51
Table 4.5 EKF – RPE comparison .......................................................................................... 58
Table 4.6 The state-augmented Extended Kalman Filter as an adaptive filter ................. 58
Table 6.1 Performance of EKFs with different values for $Q$ ............................................... 70
Table 6.2 Performance of the fully adapted filters ................................................................. 73
Table 8.1 Total number of states for the different filters ....................................................... 112
Table D.1 Parameters of the model for the three-tank system and their values ............... 137
Einführung


Für die Adaption der Filterparameter wird ein Gütefunktional formuliert, in welches die Vorhersagefehler und deren Kovarianzmatrix eingehen. Dieses Gütefunktional wird im Echtzeitbetrieb rekursiv, sonst iterativ minimiert. Dabei kommen oft gradientenbasierte Optimie-
EINFÜHRUNG (INTRODUCTION IN GERMAN)


In der vorliegenden Arbeit wird ein allgemeiner, adaptiver, kontinuierlich-diskreter Filteralgorithmus vorgeschlagen und untersucht. Als Filter können in den Algorithmus das EKF, das bias-korrigierte Filter erster Ordnung (first-order bias-corrected filter, FBF), das Minimal-Varianz-Filter zweiter Ordnung (truncated second-order filter, TSF), oder das modifizierte Gaußsche Filter (modified Gaussian second-order filter, GSF) eingesetzt werden. Für diese Filter werden vollständige Empfindlichkeitsmodelle entwickelt. Die mathematische Grundlage für die Herleitung der Empfindlichkeitsmodelle bildet die Matrizendifferentialrechnung, welche auf Kroneckerprodukte und -summen sowie auf Stapeloperationen für Matrizen zurückgreift.

Die Gliederung der Arbeit orientiert sich weitgehend an der Vorgehensweise bei der Herleitung und Untersuchung des allgemeinen Filteralgorithmus: Nach einer Motivation der Problemstellung und einem Überblick über bisherige Ansätze (Kapitel 1) und mathematische Grundlagen (Kapitel 2) werden im dritten Kapitel die in den Filtern höherer Ordnung auftretende Terme unter Einsatz von Matrizendifferentialrechnung in kompakter Schreibweise hergeleitet. Dies ist unabhängig von der Entwicklung der adaptiven Filter von Interesse, da diese kompakte Schreibweise eine wesentliche einfachere Implementierung der Filter erlaubt; darüber hinaus ist die Form der Herleitung sehr einfach und anschaulich.


Im fünften Kapitel werden die Vorhersagedifferentialgleichungen (propagation equations) in eine einfach zu implementierende Form überführt.
Ergebnisse aus Simulationsstudien und Experimenten an einer Laboranlage werden im sechsten und siebten Kapitel dargestellt, wobei auch die Vorteile des vorgestellten Filters gegenüber bisherigen Ansätzen veranschaulicht werden.

Die Zusammenfassung und eine Beurteilung der Ergebnisse erfolgen im achten Kapitel. Zusätzliches Material wird in vier Anhängen behandelt: Die verwendeten Regeln der Matrizendifferentialrechnung sind im Anhang A dargestellt; Anhang B enthält Herleitungen für das dritte Kapitel; eine alternative Formulierung des zustandserweiterten EKFs wird im Anhang C entwickelt; und im Anhang D wird das verwendete Dreitanksystem beschrieben und modelliert.
This dissertation addresses two important problems that arise in various scientific disciplines such as control engineering, communications, econometrics, and biology: (1) the problem of filtering (or state estimation for stochastic systems) and (2) the problem of (recursive) parameter estimation for state-space models. In this introduction, the significance of filtering and parameter estimation as well as their inherent connection is highlighted (Section 1.1). Furthermore, the importance of so-called gray-box state-space models is emphasized, and approaches to the parameter estimation of state-space models are surveyed. It is argued that nonlinear state-space models are underrepresented in system identification. The major obstacles responsible for this underrepresentation are identified in Section 1.2. An outline of this dissertation follows in Section 1.3.

1.1 Filtering and Parameter Estimation

The importance of filtering is stressed by Jazwinski (1970, p. 142):

> The problem of estimating the state of a stochastic dynamical system from noisy observations taken on the state is of central importance in engineering. Interest in this problem dates back . . . to the work of Gauss. Gauss was interested in determining the orbital elements of the celestial body from (many) observations . . . .

Other areas of application include reentry trajectory estimation (Jazwinski 1970, pp. 297-301), position estimation for space vehicles, radar target tracking, and ship navigation (Schrick 1977, Sections 5.3, 5.4 and 5.5). Furthermore, state estimators can be used for state-feedback control (for example, Gelb 1974, Section 9.5).

The need for parameter estimation arises in the field of system identification (see, for example, Ljung 1999). The goal of system identification is to derive a mathematical model that describes the behavior of a dynamical system. Generally, both the structure and the parameters of the model have to be determined. Once the structure is determined, parameter estimation algorithms can be used to find the unknown values of the system parameters from measurements of the system’s input and output signals. Often, it is desired that these parame-
Parameters are estimated recursively in real time during normal operation of the system. This occurs, for example, if the parameters are time-varying and need to be monitored online. A general approach that subsumes a variety of recursive parameter estimation algorithms is the recursive prediction error method developed by Ljung and Söderström (1983). This method, which is applicable to a wide class of models, starts by formulating a predictor for a given model. This predictor is then adjusted such that a cost functional of the prediction errors is minimized. Since the unknown system parameters are present in the predictor, their values can be extracted from this optimal predictor.

The connection between filtering and parameter estimation becomes evident if the parameters of a state-space model have to be estimated. This can occur when one is interested in the values of the system parameters of the state-space model (system identification) or when more accurate state estimates should be obtained by adjusting the system parameters (adaptive filtering), even if their values are not of independent interest. For this purpose, two general approaches can be pursued. In the first approach, the unknown system parameters are included in the state vector. If a model of the parameter time variations is available (for example, the parameters can be modeled as constant or as Brownian motion), an augmented dynamical model can be constructed for the new state vector. This approach, which is called state augmentation, was suggested as early as 1964 by Cox; it reduces the parameter estimation problem to a filtering problem. Since a filter is a recursive algorithm, a method for recursive parameter estimation is automatically obtained. Usually, the state-augmented Extended Kalman Filter is used. In the second approach, the recursive prediction error method is applied to a state-space model. The need for filtering arises here since the predictor for a stochastic state-space model is a filter. Thus, the application of the recursive prediction error method to a state-space model is equivalent to adaptive filtering. For linear systems, the Kalman Filter is employed as the predictor; for nonlinear systems, nonlinear filters must be used.

The relationship between parameter estimation for state-space models and filtering also manifests itself in the (offline or batch-recursive) maximum-likelihood approach (Mehra and Tyler 1973; Maybeck 1982, Chapter 10; Young 1981). Based on the assumption that the prediction errors are normally distributed (which approximately holds for fast sampling, see Mehra and Tyler [1973]), the negative log-likelihood function is given as

$$J_{ML}(t_k, p) = \frac{1}{2} \sum_{i=1}^{k} \left( e^T(t_i, p) A^{-1}(t_i, p) e(t_i, p) + \ln \det A(t_i, p) \right),$$
where $e$ and $A$ are the prediction error and its covariance matrix, respectively, and $p$ is the vector of unknown parameters. Theoretically, $e$ and $A$ have to be obtained from an optimal nonlinear filter. In practice, an approximate nonlinear filter has to be used since an optimal nonlinear filter is infinite dimensional. Thus, the minimization of the negative log-likelihood function can be seen as an (offline or batch-recursive) adaptive filtering approach: the approximate nonlinear filter is tuned to minimize a cost functional of the prediction errors.

**1.2 Recursive Parameter Estimation for State-Space Models**

In this dissertation, recursive parameter estimation/adaptive filtering for nonlinear continuous-discrete state-space models is investigated. Such models frequently arise if a system description is derived from first principles such as the laws of physics or chemistry. Balchen (1999) strongly argues for such modeling based on first principles (and against black-box modeling):

> Process engineers who spend a major part of their university training making mathematical models of processes based on first principles in physics, thermodynamics, fluid dynamics and chemistry seem to abandon all this kind of knowledge when it comes to formulating a model for the purpose of process control.

Since most industrial processes show pronounced nonlinear behavior, the most logical way of establishing a dynamic model is to utilize “first principles” in thermodynamics, fluid dynamics, reaction kinetics etc. to form the basic anatomy or skeleton of the model.

Thus there seems to be a paradox that process engineers tend to be misled into abandoning their basic process theory because somebody years ago stated that “as is well known process modeling from ‘first principles’ is very time consuming”.

If the resulting model contains unknown parameters, it is called a gray-box model (Ljung 1999, p. 13). The importance of online estimation for such models is stressed by Ljungquist and Balchen (1994):

> Modeling from fundamental physical laws and principles is of growing importance to the process industry, and the established knowledge has reduced the time needed to develop a “first principles” model to be used for control purposes. Online estimation in the resulting model, usually a nonlinear state-space model, is needed when the model is used in an advanced instrumentation system or when the model is used online in the controller.

Contrary to linear models, nonlinear models can only be approximately discretized (even if sample-and-hold devices on the input are used). Therefore, the most natural choice is a continuous-discrete model, that is, a model with continuous-time dynamics and a discrete-time measurement equation.
In the literature dealing with (recursive) system identification, nonlinear continuous-time (or continuous-discrete) state-space models are largely underrepresented. Figure 1.1 shows the results of an INSPEC search that has been narrowed down from general system identification and parameter estimation to recursive system identification for nonlinear continuous-time state-space models.

![Bar chart](image)

**Figure 1.1** Results of an INSPEC search for publications on system identification (for the period 1969 to 2000, week 13)

This underrepresentation can be attributed to some of the obstacles encountered in these approaches:

- The state-augmented Extended Kalman Filter requires the specification of the process noise and the measurement noise covariance matrices. If these values, which are usually unknown, are incorrectly specified, biased estimates can result (Ljung 1979; Ljungquist and Balchen 1994).

- The recursive prediction error method relies on the gradient of the predicted output to adjust the filter parameters. Unless the gradient is approximated (for example, by finite differences), the computation of this gradient requires a full sensitivity model of the predictor, including the derivatives of the state estimate and its error covariance matrix with respect to the parameter vector. The derivation of such a sensitivity model is extremely complicated. For linear models, this complication can be avoided through using an innovations model: the innovations model corresponds to the asymptotic Kalman Filter, in which the gain has reached its constant steady-state value. This gain can then be
1.2 Recursive Parameter Estimation for State-Space Models

Independently parameterized and estimated (Ljung and Söderström 1983, Section 3.8.2). For nonlinear models, the filter gain remains time-varying; thus, an innovations-model approach is theoretically unjustifiable, although it has been suggested in the literature (Ahmed 1994; Ljungquist and Balchen 1993, 1994; Zhou and Blanke, 1986, 1989).

- If the noise statistics are unknown (which is usually the case), they also have to be parameterized and estimated; otherwise, biased parameters can result. This increases the dimension of the resulting filter.

Furthermore, the similarity of apparently different algorithms is often hidden in nonuniform notation and terminology (Ljungquist and Balchen 1994). Since a comparison between algorithms for nonlinear system identification is often based on heuristic and intuitive insight rather than on mathematical rigor, a formulation that permits an easy interpretation of the differences between various algorithms is desirable.

Table 1.1 lists some approaches to the recursive parameter estimation for stochastic state-space models. In the middle column of this table, the entry “constant filter gain” refers to an innovations model. For nonlinear systems, only the approach of Jakoby and Pandit (1987) uses a time-varying filter gain (in an Extended Kalman Filter). With the exception of Zhou and Blanke’s (1986, 1989) work, higher-order filters have not been used.

A method that is not included in Table 1.1 is the bootstrap method for linear, discrete-time systems. This method was suggested for input-output models by Pandya (1974) and was extended to state-space models by Prasad and Sinha (1977) and El-Sherief and Sinha (1979). In the original form it consists of two decoupled estimators for the states and the parameters that mutually exchange their estimates; the gain of the state estimator is chosen without reference to optimal filtering. Ahmed and Sait (1989) modified this method to include the estimation of a constant filter gain. Since it requires a special parameterization of the state-space model (observability canonical form/row companion form) and is only applicable to discrete-time systems, it has no relevancy to the work in this dissertation. For linear, (multivariable) discrete-time systems, it is, however, an attractive alternative to the recursive prediction error method, particularly, since it only includes linear least-squares estimation (Gambier 1996).
<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>state-augmented Extended Kalman Filter</td>
<td>for example, Gelb (1974)</td>
</tr>
<tr>
<td>Linear, discrete</td>
<td>adaptive Kalman Filter, constant/time-varying filter gain</td>
<td>Ljung (1979); Ljung and Söderström (1983)</td>
</tr>
<tr>
<td></td>
<td>(batch-)recursive maximum-likelihood method (adaptive Kalman Filter),</td>
<td>Maybeck (1982)</td>
</tr>
<tr>
<td></td>
<td>time-varying filter gain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>adaptive Kalman Filter, time-varying filter gain</td>
<td>El-Fattah (1983)</td>
</tr>
<tr>
<td></td>
<td>third-order moment estimator (3-OM), state augmentation</td>
<td>Wiberg et al. (1997)</td>
</tr>
<tr>
<td>Linear, continuous-discrete</td>
<td>adaptive Kalman Filter, constant filter gain</td>
<td>Gavel and Azevedo (1982)</td>
</tr>
<tr>
<td>Nonlinear, discrete</td>
<td>adaptive First-Order Bias-Corrected Filter, constant filter gain</td>
<td>Zhou and Blanke (1986, 1989)</td>
</tr>
<tr>
<td></td>
<td>adaptive Extended Kalman Filter, time-varying filter gain, known noise</td>
<td>Jakoby and Pandit (1987)</td>
</tr>
<tr>
<td></td>
<td>covariance matrices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>adaptive Extended Kalman Filter, constant filter gain</td>
<td>Ljungquist and Balchen (1993, 1994)</td>
</tr>
<tr>
<td></td>
<td>adaptive Extended Kalman Filter, constant filter gain</td>
<td>Ahmed (1994)</td>
</tr>
<tr>
<td></td>
<td>adaptive (modified) Extended Kalman Filter, time-varying filter gain,</td>
<td>Chu et al. (1996)</td>
</tr>
<tr>
<td></td>
<td>no sensitivity model</td>
<td></td>
</tr>
<tr>
<td></td>
<td>adaptive Extended Kalman Filter, First-Order Bias-Corrected Filter,</td>
<td>this dissertation</td>
</tr>
<tr>
<td></td>
<td>Truncated Second-Order Filter, or Modified Gaussian Second-Order Filter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>time-varying filter gain</td>
<td></td>
</tr>
</tbody>
</table>
Also excluded in Table 1.1 are methods for linear systems that use models without explicit parameterizations such as subspace algorithms. For a general overview on state-space identification methods for discrete, multivariable, linear systems, see Gambier (1996, Section 4.2).

1.3 Dissertation Outline

To overcome the obstacles mentioned above and to provide a better understanding of the adaptive filtering algorithms, the following items are addressed in this dissertation:

- Derivation of a compact sensitivity model for the continuous-discrete versions of the four common approximate nonlinear filters: Extended Kalman Filter, First-Order Bias-Corrected Filter, Truncated Second-Order Filter, Modified Gaussian Second-Order Filter. With this sensitivity model, a general adaptive filtering algorithm is formulated.

- Investigation of the relationship between state augmentation and adaptive filtering. This is done through an exact reformulation of the state-augmented Extended Kalman Filter as a sensitivity-based adaptive filter. Ljung (1979), Ljung and Söderström (1983, Appendix 3.C), and Ljungquist and Balchen (1993, 1994) used approximations and asymptotic arguments for their investigation, but further insight, particularly an interpretation of the differences between both approaches, is obtained from the exact reformulation used here.

- Transformation of the algorithm to a form that permits an easy implementation. This includes the introduction of mechanisms for the elimination of redundant states from the filters and for the vectorization of the matrix differential equations.

This dissertation is organized as follows. In Chapter 2, mathematical preliminaries used in subsequent chapters are reviewed. This chapter includes material on Kronecker products and stacking operations as well as on matrix differential calculus. These mathematical operations offer powerful tools for the derivation of a general sensitivity model. The higher-order filters mentioned above are reformulated in Chapter 3. Although this is, strictly speaking, only a rederivation of results that are available in the literature, the new formulation, which is also reported in Bohn and Unbehauen (2000), is extremely attractive since it makes developing sensitivity models, and thus, adaptive versions of these filters straightforward. In Chapter 4, the parameter estimation/adaptive filtering problem is addressed. The well-known state-augmentation approach and the innovations-model-based approach are reviewed and their shortcomings are discussed (Sections 4.2 and 4.4, respectively). By applying matrix differential calculus and utilizing the new filter formulations of Chapter 3, a general adaptive filtering algorithm is developed (this algorithm is first derived for an adaptive Extended Kalman Filter in Section 4.5; the terms that need to be included for higher-order filters are
obtained in Section 4.6). The relationship between the least-squares cost-functional and the log-likelihood function is highlighted (Section 4.7); it is shown that the algorithm can be readily extended to a maximum-likelihood cost functional. The connection between the state-augmented Extended Kalman Filter and the sensitivity model is analyzed (Section 4.8). In contrast to earlier comparisons (Ljung 1979; Ljung and Söderström 1983; Ljungquist and Balchen 1994), this analysis is based on an exact reformulation of the Extended Kalman Filter (derived in Appendix C). A suitable form for the propagation equations of the adaptive filter is developed in Chapter 5. Results of simulation studies are given in Chapter 6, and a real-time case study (parameter estimation and tracking for a three-tank system) is presented in Chapter 7. A summary and conclusions follow in Chapter 8. Supplementary material is covered in the appendices: Appendix A contains results of matrix differential calculus, Appendix B gives auxiliary derivations for Chapter 3, the state-augmented Extended Kalman Filter is rewritten as a sensitivity-based algorithm in Appendix C, and Appendix D contains a description of the three-tank system used for the real-time identification experiment.

Once again the underlying assumption of this dissertation needs to be stressed – the necessity of keeping a continuous-discrete description of the model. Such a model is useful for a variety of reasons: For example, obtaining or monitoring system parameters that have a distinct physical meaning might be required; as opposed to a linear system, a nonlinear system can generally only be approximately discretized, even if sample-and-hold devices on the input are used. Thus, the relationship between the parameters of the continuous-time model and the parameters of the discrete-time model may be lost. Or, the behavior of the system between sampling instants might be of interest, and these data are to be reconstructed from the model. Furthermore, if a model structure and some parameters are known, only a few additional parameters might have to be estimated. This would be an advantage over the so-called black-box approaches such as neural networks, which usually do not utilize any prior knowledge of the system and require a large number of parameters. Furthermore, if only one parameter changes due to a modification of the system, only that specific parameter has to be reestimated, whereas in a black-box model, the whole set of parameters would have to be redetermined.

If, instead, models based on first principles are not available and only the input-output behavior of the system at the sampling instants is of interest, black-box models might offer a better alternative (particularly when considering the complexity of the adaptive filtering approach). It is, however, argued here along the lines of Balchen’s (1999) reasoning that insight from first principles should be included and should also be kept if the model is
(approximately) discretised. For a further discussion regarding the importance of identifying continuous-time systems, consult Chapter 1 of Unbehauen and Rao (1987) and the related survey articles (Unbehauen and Rao 1990, 1998; Young 1981). Black-box modeling is treated in detail in Sjöberg et al. (1995) and Juditsky et al. (1995); the structure detection for black-box models is discussed in Haber and Unbehauen (1990).
Chapter 2

Mathematical Preliminaries

In this chapter, notation is introduced; furthermore, mathematical tools that are used in following chapters are reviewed. In Section 2.1, notation regarding scalars, vectors, and matrices is presented. Since Kronecker products and stacking operations are needed to vectorize matrix expressions and to handle matrix derivatives, they are defined in Section 2.2. For the derivation of various algorithms in later chapters, it is often necessary to perform differentiation operations on vector and matrix functions with respect to vector and matrix arguments. Matrix differential calculus is a major tool for these derivations and is subsequently introduced in Section 2.3. Further rules of matrix differential calculus are given in Appendix A. In Sections 2.4 and 2.5, two recursive algorithms of stochastic-approximation type that are often employed for recursive parameter estimation are reviewed.

2.1 Notation

Whenever possible, lowercase Greek letters are used for scalars, boldface lowercase Latin letters for vectors, and boldface capital Latin letters for matrices. Exceptions include indices \((i, j, k, \text{ etc.})\), the time variable \(t\) and parameters with a physical meaning (damper constant \(d\), spring constant \(k\)), elements of matrices and vectors, and terms that are vectors or matrices in the general case (the signal \(y\), the variance \(R\)). Variables are set in italics, whereas constant quantities (such as the identity matrix) are set in roman. A subscript following a letter set in roman corresponds to the entry of a matrix or vector, for example, \(A_{ij}\) identifies the element in the \(k,l\) position of the matrix \(A\). Exceptions are explanatory indices, for example, \(P_{xx}\) is the covariance matrix of the estimation error in \(x\). For additional clarity, the dimensions of a vector or a matrix are occasionally given by a superscript, for example, \(A_{\text{nn}}\) indicates that the matrix \(A\) is of dimension \(n\) by \(m\). The inverse of a matrix \(A\) is denoted by \(A^{-1}\). The superscript \(\text{"T"}\) is used for the transpose of a matrix or a vector. The identity matrix of dimension \(n\) is denoted by \(I_n\), and a matrix or vector with all zero elements is written as \(0\). The expression \(E_{ij}^{(kl)}\) stands for a \(k\) by \(l\) matrix with a one in the \(i,j\) position and zeroes everywhere else. The
symbol “E” refers to the expectation. The Dirac impulse is denoted by $\delta(t)$; $\delta_{ij}$ is the Kronecker delta function. The latter is one if $i$ equals $j$ and zero for all other cases.

### 2.2 Kronecker Products and Stacking Operations

In this section, definitions and facts related to Kronecker products and stacking operations are given. Only relations that are used in subsequent chapters are stated here. Further material on Kronecker products can be found in Weinmann (1991, Chapter 4) and Magnus and Neudecker (1999, Chapter 2).

The Kronecker product is defined as (Weinmann 1991, p. 73)

$$A \otimes B := \begin{bmatrix} A_{11}B & \cdots & A_{1m}B \\
\vdots & \ddots & \vdots \\
A_{n1}B & \cdots & A_{nm}B \end{bmatrix}. \quad (2.1)$$

The Kronecker permutation matrix is (Weinmann 1991, p. 73)

$$U_{kl} := \sum_{i=1}^{k} \sum_{j=1}^{l} E_{ij}^{(k,l)} \otimes E_{ji}^{(k,l)}. \quad (2.2)$$

Two stacking operations for matrices are introduced: the “col” operator stacks a matrix columnwise into a column vector, and the “row” operator stacks a matrix rowwise into a row vector (Weinmann 1991, pp. 76-77). The “col” operator is sometimes denoted as the “vec” operator in the literature (Magnus and Neudecker 1999, p. 30). The inverse of the “col” operator is the “mat” operator, which stacks a vector columnwise into a matrix. Thus, the relation

$$A = \text{mat}_{\text{vec}} \text{ col } A^{(\text{vec})} \quad (2.3)$$

holds.

A fact that is often used to vectorize matrix expressions is the relation (Weinmann 1991, p. 76; Magnus and Neudecker, Chapter 2, Theorem 2)

$$\text{col } ABC = (C^T \otimes A) \text{ col } B. \quad (2.4)$$

The result of the column operator applied to the transpose of a matrix can be expressed in terms of the nontransposed matrix as (Weinmann 1991, p. 77)

$$\text{col } A^{(\text{vec})T} = U_{nm} \text{ col } A. \quad (2.5)$$

The Kronecker sum of two square matrices is defined as (Weinmann 1991, p. 75)

$$B^{(\text{vec})} \oplus C^{(\text{vec})} := I_m \otimes B + C \otimes I_n. \quad (2.6)$$
This Kronecker sum arises if an expression of the form \( BA + AC^\top \) is vectorized as
\[
\col(B^{(n\times n)}A^{(n\times n)} + A^{(n\times n)}C^{(n\times n)^\top}) = (I_n \otimes B + C \otimes I_n) \col A.
\] (2.7)

If a symmetric matrix is stacked into a vector, this vector contains redundant entries. It is sometimes convenient to eliminate these redundant entries. This can be done by premultiplying the vector by a specific matrix, which is here denoted as the reduction matrix. For this purpose, the “reduced column operator” \( c \) is introduced as
\[
c A^{(n\times n)} := T_t^{(n)} \col A; \quad A = A^\top,
\] (2.8)
where \( T_t^{(n)} \) is an \( n(n+1)/2 \) by \( n^2 \) matrix that eliminates the redundant entries from \( \col A \). This reduction matrix is the (right) pseudoinverse of the duplication matrix (Magnus and Neudecker 1999, pp. 48-49) which can be used to recover the full column vector from the reduced vector, that is,
\[
\col A^{(n\times n)} = T_t^{(n)} c A,
\] (2.9)
where \( T_t^{(n)} \) is an \( n^2 \) by \( n(n+1)/2 \) matrix. Thus, the symmetric matrix \( A \) can be recovered from its reduced column vector as
\[
A = \mat_{n\times n} T_t^{(n)} c A^{(n\times n)}.
\] (2.10)

\section*{2.3 Matrix Differential Calculus}

Several definitions for the derivative of a vector or matrix function with respect to a vector or matrix exist in the literature (Magnus and Neudecker 1999; Weinmann 1991; Bentler and Lee 1978; Vetter 1973). If carelessly defined, they suffer from being inconsistent in the sense that the higher-dimensional derivatives (such as the derivative of a matrix with respect to a matrix) do not collapse to the corresponding lower-dimensional derivatives when the arguments change their dimensions. For example, if the gradient of a scalar \( \alpha \) with respect to a vector \( x \) of dimension \( n \) by 1 is defined as a column vector, that is,
\[
\frac{d\alpha}{dx} := \begin{bmatrix} \frac{d\alpha}{dx_1} \\ \vdots \\ \frac{d\alpha}{dx_n} \end{bmatrix},
\]
and the derivative of a vector function \( f \) of dimension \( m \) by 1 with respect to the vector \( x \) (the Jacobi matrix) is defined as
then it is obvious that when $f$ reduces to a scalar both definitions are inconsistent.

To the best of the author’s knowledge, the only notation that fully avoids these inconsistencies is the one used by Vetter (1973) and Weinmann (1991). Magnus and Neudecker (1999) call all notation but theirs “bad notation,” but even their “good notation” is slightly inconsistent: the derivative of a scalar with respect to a matrix and the derivative of a matrix with respect to a scalar do not fit into their scheme (where they would be vectors) and require separate definitions (Magnus and Neudecker 1999, pp. 173-74). Their aim is not to provide a general definition for a matrix derivative but rather to define generalizations of the Jacobi and the Hessian matrices. Therefore, the notation of Vetter (1973) and Weinmann (1991) is used here. In this notation, the $nr$ by $ms$ derivative of the $n$ by $m$ matrix function $A(M)$ with respect to its $r$ by $s$ matrix argument $M$ can be seen as the Kronecker product of the $r$ by $s$ “matrix derivative operator” and the matrix function $A$, that is,

$$\frac{dA(M)}{dM} = \frac{d}{dM} \otimes A(M) = \begin{bmatrix} \frac{d}{dM_{11}} & \ldots & \frac{d}{dM_{1s}} \\ \vdots & \ddots & \vdots \\ \frac{d}{dM_{r1}} & \ldots & \frac{d}{dM_{rs}} \end{bmatrix} \otimes A(M) = \begin{bmatrix} \frac{dA(M)}{dM_{11}} & \ldots & \frac{dA(M)}{dM_{1s}} \\ \vdots & \ddots & \vdots \\ \frac{dA(M)}{dM_{r1}} & \ldots & \frac{dA(M)}{dM_{rs}} \end{bmatrix}. \quad (2.11)$$

From this definition, it follows that $\frac{df}{dx^T}$, where $f$ and $x$ are vectors, is the usual Jacobi matrix, and its transpose $\frac{df^T}{dx}$ is the gradient (matrix). Further, $\frac{d}{dx}$ and $\frac{d}{dx^T}$ denote two different derivatives (applied to a scalar, the former would be a column vector, whereas the latter would be a row vector). Higher derivatives are defined accordingly, where the derivative operations have to be carried out from right to left. For example, the Hessian matrix of a scalar is $\frac{d^2\alpha}{dx^Tdx}$. Since different higher derivatives such as $\frac{d^2}{dx^Tdx}$, $\frac{d^2}{dx^Tdx^T}$ exist, the notation $\frac{d^2}{dx^2}$ commonly used for Hessian matrices becomes ambiguous and can no longer be used.

Standard rules for matrix differential calculus (product rule, chain rule, Kronecker product rule) and the Taylor expansion of matrix-valued functions are given by Weinmann (1991, Chapter 5). For easy reference, they are also listed in Appendix A.
For functions with multiple arguments, the generalized chain rule can be derived: If the arguments \( B_1, \ldots, B_q \) of a matrix function \( A(B_1, \ldots, B_q, M) \) are functions of the matrix \( M \), the derivative of \( A \) with respect to \( M \) is given as

\[
\frac{dA(B_1, \ldots, B_q, M)}{dM} = \sum_{k=1}^{q} \left( \frac{dA(B_1(M), \ldots, B_{k-1}(M), B_k(M^\star), B_{k+1}(M), \ldots, B_q(M), M)}{dM^\star} \right)_{M^\star=M}
\]

where \( M^\star \) is a dummy variable used for differentiation. To simplify this notation, the usual concept of a partial derivative is introduced as

\[
\frac{\partial A(B_1, \ldots, B_q, M)}{\partial M} := \frac{dA(B_1(M), \ldots, B_q(M), M^\star)}{dM^\star} \Bigg|_{M^\star=M},
\]

that is, the dependency of \( B_1, \ldots, B_q \) on \( M \) is not taken into account when the partial derivative is evaluated. Decomposing the first term on the right-hand side of Equation (2.12) using the matrix chain rule gives the generalized matrix chain rule

\[
\frac{dA(B_1, \ldots, B_q, M)}{dM} = \sum_{k=1}^{q} \left( \frac{dA(B_1(M), \ldots, B_q(M), M^\star)}{dM^\star} \right) \left( \frac{\partial \text{col}(B_k^T)}{\partial M} \otimes I_m \right)
\]

For the special case of a matrix function \( A \) of vector-valued arguments \( b \) and \( m \), the generalized matrix chain rule becomes

\[
\frac{dA^{(n\times n)}(b_1, \ldots, b_q, M)}{dm^{(r\times 1)}} = \sum_{k=1}^{q} \left( \frac{dA(b_1, \ldots, b_q, m)}{db_k^T} \otimes I_m \right) + \frac{dA(b_1, \ldots, b_q, m)}{dm^T}.
\]

Applying the Kronecker product rule, the derivative of the Kronecker sum is obtained as

\[
\frac{d}{dM^{(n\times n)}} (B^{(n\times n)} \otimes C^{(n\times n)}) = (I_r \otimes U_{mn}) \left( \frac{dB}{dM} \otimes I_n \right) \left( I_s \otimes U_{nm} \right) + \frac{dC}{dM} \otimes I_n.
\]

A useful relation that does not seem to be reported in closed form in the literature is the derivative of the logarithm of the determinant of a matrix (a term that arises in the log-likelihood function for a Gaussian distribution):
Applying the chain rule to an individual element of this expression gives
\[
\frac{d \ln \det A(p)}{dp_i} = \frac{1}{\det A} \frac{d \det A(p)}{dp_i},
\]
and using the relation (Weinmann 1991, pp. 79-80)
\[
\frac{d \det A(p)}{dp_i} = \text{trace}(\text{adj} A \frac{dA(p)}{dp_i}),
\]
leads to the well-known relation (Maybeck 1982, p. 79)
\[
\frac{d \ln \det A(p)}{dp_i} = \text{trace}(A^{-1} \frac{dA(p)}{dp_i}).
\]
An expression for the vector derivative can be obtained with
\[
\text{trace}(A^{-1} \frac{dA(p)}{dp_i}) = \text{row}(A^{-1}) \col \frac{dA(p)}{dp_i};
\]
this yields the desired result
\[
\frac{d \ln \det A(p)}{dp^\text{vec}_i} = \begin{bmatrix}
\text{row}(A^{-1}) \col \frac{dA(p)}{dp_1} \\
\vdots \\
\text{row}(A^{-1}) \col \frac{dA(p)}{dp_s}
\end{bmatrix} = (I_s \otimes \text{row}(A^{-1})) \col \frac{dA(p)}{dp^\text{vec}_i}.
\]

### 2.4 Gauss-Newton Algorithm for a Least-Squares Cost Functional

In this section, the well-known stochastic Gauss-Newton algorithm that is often used for recursive parameter estimation (Ljung and Söderström 1983; Zypkin 1987) is reviewed. The starting point for this algorithm is the cost functional
\[
J(t_k, p) = E \frac{1}{2} e^T(t_k, p) A^{-1}(t_k) e(t_k, p),
\]
where \( e \) is an error vector that depends on the unknown parameter vector \( p \), and \( A \) is a sequence of weighting matrices. Alternatively, the empirical cost functional
\[
J(t_k, p) = \frac{1}{2} \sum_{i=1}^{k} e^T(t_i, p) A^{-1}(t_i) e(t_i, p)
\]
2.4 GAUSS-NEWTON ALGORITHM FOR A LEAST-SQUARES COST FUNCTIONAL

can be used.

Observations of the error $e$ are available at discrete time instants $t_k$. Using arguments from stochastic approximation theory (Saridis 1974) and the Gauss-Newton optimization method, an algorithm that processes new observations as they become available to give improved parameter estimates can be derived as

$$
\hat{p}(t_{k+1}) = \hat{p}(t_k) - \frac{1}{k+1} R^{-1}(t_{k+1}) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} A^{-1}(t_{k+1}) e(t_{k+1}),
$$

(2.25)

where the matrix $R$ is the normalized information matrix, and the last three terms on the right-hand side correspond to the gradient of the cost functional. Alternatively, $R$ can be seen as the Gauss-Newton approximation of the Hessian matrix of the cost functional in Equation (2.23). The matrix $R$ is given as

$$
R(t_{k+1}) = \frac{1}{k+1} \sum_{i=1}^{k+1} \frac{d e^T(t_i, \hat{p}(t_i))}{d \hat{p}(t_i)} A^{-1}(t_i) \frac{d e(t_i, \hat{p}(t_i))}{d \hat{p}^T(t_i)},
$$

(2.26)

and can be recursively computed as

$$
R(t_{k+1}) = R(t_k) + \frac{1}{k+1} \left( \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} A^{-1}(t_{k+1}) \frac{d e(t_{k+1}, \hat{p}(t_k))}{d \hat{p}^T(t_k)} - R(t_k) \right).
$$

(2.27)

The above algorithm assigns equal weight to all observations. This becomes evident in Equation (2.26), which corresponds to the computation of a standard average. Often an algorithm that puts more weight on recent observations and discards observations from the past is desired. This occurs, for example, when time-varying parameters have to be estimated. Such an algorithm results if a technique called exponential age-weighting or exponential forgetting is used (Maybeck 1982, pp. 28-30; Jazwinski 1970, pp. 307-8). Formally, at every time instant, the expectation in the cost functional is taken with a different sequence of weighting matrices: the matrices $A$ are replaced by the following “aged” versions

$$
A_{aged}(t_i, t_k) = \lambda^{i-k} A(t_i),
$$

where $\lambda$ is a positive real number smaller than one. The update equations then change to

$$
\hat{p}(t_{k+1}) = \hat{p}(t_k) - \gamma(k+1) R^{-1}(t_{k+1}) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} A^{-1}(t_{k+1}) e(t_{k+1}),
$$

(2.28)

and

$$
R(t_{k+1}) = R(t_k) + \gamma(k+1) \left( \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} A^{-1}(t_{k+1}) \frac{d e(t_{k+1}, \hat{p}(t_k))}{d \hat{p}^T(t_k)} - R(t_k) \right),
$$

(2.29)

with
\[ \gamma(k+1) = \frac{\gamma(k)}{\lambda + \gamma(k)} = \frac{1 - \lambda}{1 - \lambda^2}. \]  

(2.30)

If \( \lambda \) is set to one, Equations (2.28) and (2.29) become identical to Equations (2.25) and (2.27).

The algorithm given above is usually implemented in a different form to avoid the inversion of the matrix \( R \). For this, a matrix \( P \) is defined as

\[ P(t_{k+1}) := \gamma(k+1) R^{-1} (t_{k+1}). \]

The matrix \( P \) is usually called the covariance matrix (it would be the covariance matrix of the parameter estimation error if \( p \) were a stochastic variable with a Gaussian distribution). The algorithm can then be rewritten as

\[ S(t_{k+1}) = \lambda A(t_{k+1}) + \frac{d e(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} P(t_k) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)}, \]  

(2.31)

\[ L(t_{k+1}) = P(t_k) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} S^{-1}(t_{k+1}), \]  

(2.32)

\[ \hat{p}(t_{k+1}) = \hat{p}(t_k) - L(t_{k+1}) e(t_{k+1}), \]  

(2.33)

and

\[ P(t_{k+1}) = \frac{1}{\lambda} (P(t) - L(t_{k+1}) S(t_{k+1}) L^T(t_{k+1})). \]  

(2.34)

Alternatively, the covariance matrix could be updated first as

\[ P(t_{k+1}) = \frac{1}{\lambda} (P(t) - P(t_k) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} S^{-1}(t_{k+1}) \frac{d e(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} P(t_k)), \]  

(2.35)

and the gain matrix can be computed using the updated covariance matrix as

\[ L(t_{k+1}) = P(t_{k+1}) \frac{d e^T(t_{k+1}, \hat{p}(t_k))}{d \hat{p}(t_k)} A^{-1}(t_{k+1}). \]  

(2.36)

Yet another equivalent form is obtained by defining

\[ \bar{P}(t_{k+1}) := \frac{1}{\lambda} P(t_{k+1}) \]  

(2.37)

and

\[ \bar{S}(t_{k+1}) := \frac{1}{\lambda} S(t_{k+1}). \]  

(2.38)

Thus, the algorithm becomes
\[ \bar{S}(t_{k+1}) = A(t_{k+1}) + \frac{d}{dp^T(t_k)} \left( \frac{de(t_{k+1}, \hat{p}(t_k))}{dp}(t_k) \right) \frac{de^T(t_{k+1}, \hat{p}(t_k))}{dp^T(t_k)}, \quad (2.39) \]

\[ L(t_{k+1}) = \bar{P}(t_k) \frac{de^T(t_{k+1}, \hat{p}(t_k))}{dp^T(t_k)} \bar{S}^{-1}(t_{k+1}) \bar{e}(t_{k+1}), \quad (2.40) \]

\[ \hat{p}(t_{k+1}) = \hat{p}(t_k) - L(t_{k+1}) \bar{e}(t_{k+1}), \quad (2.41) \]

and

\[ \bar{P}(t_{k+1}) = \frac{1}{\lambda}(\bar{P}(t_{k+1}) - L(t_{k+1}) \bar{S}(t_{k+1}) L^T(t_{k+1})). \quad (2.42) \]

The gain matrix \( L \) remains unaffected by this transformation, and both algorithms are equal if their initial values are replaced according to \( \bar{P}(t_0) = P(t_0)/\lambda \). Table 2.1 summarizes the recursive Gauss-Newton algorithm.

<table>
<thead>
<tr>
<th>Table 2.1 Recursive Gauss-Newton algorithm for a least-squares cost functional</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weighting matrix (possibility I)</strong></td>
</tr>
<tr>
<td><strong>Weighting matrix (possibility II)</strong></td>
</tr>
<tr>
<td><strong>Gain matrix</strong></td>
</tr>
<tr>
<td><strong>Parameter update</strong></td>
</tr>
<tr>
<td><strong>Covariance update</strong></td>
</tr>
</tbody>
</table>

### 2.5 Gauss-Newton Algorithm for a Maximum-Likelihood Cost Functional

A recursive Gauss-Newton algorithm for minimizing the maximum-likelihood cost functional can also be constructed. This algorithm is included here for reference. The maximum-likelihood cost functional for nonlinear regression with normally distributed errors is given as (for example, Magnus and Neudecker 1999, p. 325; Chu et al. 1996)

\[ J_{\text{ML}}(t_k, p) = \frac{1}{2} \sum_{i=1}^{k} (e^T(t_i, p) A^{-1}(t_i, p) e(t_i, p) + \ln \det A(t_i, p)), \quad (2.48) \]
where $e$ is the prediction error, and $A$ is the prediction error covariance matrix. Partially following the notation of Chu et al. (1996), the gradient for this cost functional is given as

$$
\frac{dJ_{MB}(t_i)}{dp} = \sum_{i=1}^{k} \left( \frac{de^T(t_i)}{dp} A^{-1}(t_i) e(t_i) + \frac{1}{2} (\eta(t_i) - \mu(t_i)) \right),
$$

(2.49)

where the argument $p$ has been omitted (all gradients are, as usual, understood as being computed with the latest estimate of $p$). The terms $\eta$ and $\mu$ are given as

$$
\eta(t_i) = \begin{bmatrix}
\text{trace} (A^{-1}(t_i) \frac{dA(t_i)}{dp_1}) \\
\vdots \\
\text{trace} (A^{-1}(t_i) \frac{dA(t_i)}{dp_s})
\end{bmatrix},
$$

(2.50)

and

$$
\mu(t_i) = \begin{bmatrix}
\left( e^T(t_i) A^{-1}(t_i) \frac{dA(t_i)}{dp_1} A^{-1}(t_i) e(t_i) \right) \\
\vdots \\
\left( e^T(t_i) A^{-1}(t_i) \frac{dA(t_i)}{dp_s} A^{-1}(t_i) e(t_i) \right)
\end{bmatrix},
$$

(2.51)

see Equation (12) of Chu et al. (1996). An alternative formulation is given by Maybeck (1982), Equation (10-29a), as

$$
\eta(t_i) - \mu(t_i) = \begin{bmatrix}
\text{trace} \left( [A^{-1}(t_i) - A^{-1}(t_i) e(t_i) e^T(t_i) A^{-1}(t_i)] \frac{dA(t_i)}{dp_1} \right) \\
\vdots \\
\text{trace} \left( [A^{-1}(t_i) - A^{-1}(t_i) e(t_i) e^T(t_i) A^{-1}(t_i)] \frac{dA(t_i)}{dp_s} \right)
\end{bmatrix}.
$$

(2.52)

Using the derivative of the inverse of a matrix (Rule 7 in Appendix A) and Equation (2.22), more compact expressions can be derived as

$$
\eta(t_i) = \left[ I_s \otimes \text{row} (A^{-1}(t_i)) \right] \text{col} \frac{dA(t_i)}{dp^T},
$$

(2.53)

and

$$
\mu(t_i) = \left[ I_s \otimes e^T(t_i) A^{-1}(t_i) \right] \frac{dA(t_i, p)}{dp} A^{-1}(t_i) e(t_i).
$$

(2.54)
An equivalent expression that looks more like Equation (2.52) can be obtained as
\[ \eta(t_i) - \mu(t_i) = \left[ I_x \otimes \text{row} \left( A^{-1}(t_i) - A^{-1}(t_i) e(t_i) e^T(t_i) A^{-1}(t_i) \right) \right] \text{col} \frac{dA(t_i)}{dp^T}. \] (2.55)

The normalized information matrix is given as
\[ R(t_k) = \frac{1}{\lambda} \sum_{i=1}^{k} \left( \frac{de^T(t_i)}{dp} \right) A^{-1}(t_i) \frac{de(t_i)}{dp^T} + \frac{1}{2} \eta(t_i) \eta^T(t_i). \] (2.56)

The recursive Gauss-Newton algorithm of Chu et al. (1996) is, in a slightly different notation and with a forgetting factor \( \lambda \), given in Table 2.2. For an error covariance matrix \( A \) that is independent of \( p \), it follows that \( \eta = \mu = 0 \), and the algorithm collapses to the corresponding algorithm for the least-squares cost functional.

**Table 2.2** Recursive Gauss-Newton algorithm for a maximum-likelihood cost functional

<table>
<thead>
<tr>
<th>Gradient terms</th>
<th>( \eta(t_{k+1}) = \left[ I_x \otimes \text{row} \left( A^{-1}(t_{k+1}) \right) \right] \text{col} \frac{dA(t_{k+1})}{dp} )</th>
<th>(2.57)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \mu(t_{k+1}) = \left[ I_x \otimes e^T(t_{k+1}) A^{-1}(t_{k+1}) \right] \frac{dA(t_{k+1})}{dp} A^{-1}(t_{k+1}) e(t_{k+1}) )</td>
<td>(2.58)</td>
</tr>
<tr>
<td>Weighting matrix</td>
<td>( S(t_{k+1}) = \lambda A(t_{k+1}) + \frac{de(t_{k+1})}{dp^T} P(t_k) \frac{de^T(t_{k+1})}{dp} )</td>
<td>(2.59)</td>
</tr>
<tr>
<td>Gain matrix</td>
<td>( L(t_{k+1}) = P(t_k) \frac{de^T(t_{k+1})}{dp} S^{-1}(t_{k+1}) )</td>
<td>(2.60)</td>
</tr>
<tr>
<td>Auxiliary covariance update</td>
<td>( P^*(t_{k+1}) = \frac{1}{\lambda} \left( P(t_k) - L(t_{k+1}) S(t_{k+1}) L^T(t_{k+1}) \right) )</td>
<td>(2.61)</td>
</tr>
<tr>
<td>Auxiliary weighting factor</td>
<td>( S^<em>(t_{k+1}) = 2 + \eta^T(t_{k+1}) P^</em>(t_{k+1}) \eta(t_{k+1}) )</td>
<td>(2.62)</td>
</tr>
<tr>
<td>Auxiliary gain matrix</td>
<td>( L^<em>(t_{k+1}) = \frac{P^</em>(t_{k+1}) \eta(t_{k+1})}{S^*(t_{k+1})} )</td>
<td>(2.63)</td>
</tr>
<tr>
<td>Covariance update</td>
<td>( P(t_{k+1}) = P^<em>(t_{k+1}) - L^</em>(t_{k+1}) S^<em>(t_{k+1}) L^{T^</em>}(t_{k+1}) )</td>
<td>(2.64)</td>
</tr>
<tr>
<td>Parameter update</td>
<td>( \hat{p}(t_{k+1}) = \hat{p}(t_k) - L(t_{k+1}) e(t_{k+1}) - \frac{1}{2} P(t_{k+1}) (\eta(t_{k+1}) - \mu(t_{k+1})) )</td>
<td>(2.65)</td>
</tr>
</tbody>
</table>
Chapter 3

Approximate Nonlinear Filters

This chapter discusses state estimation for nonlinear continuous-discrete systems (continuous-time systems with discrete output measurements) through approximate nonlinear filters. It therefore lays the foundation for the following chapters where it is shown how adaptive versions of these filters can be constructed and used as parameter estimators. Furthermore, the results of this chapter are of independent interest, since they can be advantageously applied toward the implementation of the nonlinear filters.

For nonlinear systems, state estimation is often done by means of approximate nonlinear filters. Standard filters that can be employed for this task are the Extended Kalman Filter (EKF), the First-Order Bias-Corrected Filter (FBF), the Truncated Second-Order Filter (TSF), and the Modified Gaussian Second-Order Filter (GSF) (Gelb 1974; Jazwinski 1970; Maybeck 1982). The higher-order filters (FBF, TSF, GSF) differ from the EKF in so far that additional terms are present in the equations for the propagation of the state estimate, the output prediction, and the prediction error covariance matrix. Expressions for these terms are given in the literature, but they are rather cumbersome in the sense that the vector and matrix terms have to be computed element by element and that summations over up to four indices are necessary. Alternative expressions for these higher-order terms are derived in this chapter using the rules of matrix differential calculus. Since matrix differential calculus is employed, there is a clear correspondence between the derivation for the vector case and for the scalar case (Jazwinski 1970, pp. 336-46; Maybeck 1982; pp. 215-27). The simplified expressions are useful for an easy, straightforward implementation (in the sense of prototyping, for example, in Matlab) of these filters, and the derivation shows that matrix differential calculus serves as a powerful tool for these types of problems. Furthermore, it is shown in Chapter 4 that adaptive filters can easily be constructed from the filter formulation given here. The results of this chapter are also reported in Bohn and Unbehauen (2000).

This chapter is organized as follows. In Section 3.1, a general algorithm containing the four filters considered here is introduced. The higher-order terms present in the equations of these filters are rederived in Section 3.2.
3.1 General Filtering Algorithm

In this dissertation, only continuous-discrete filters (for continuous-time systems with discrete-time measurements) are considered. The underlying system is thus modeled as

\[ \dot{x}(t) = f(x(t)) + v(t), \]  
\[ y(t_k) = h(x(t_k)) + w(t_k), \]

where, with a slight abuse of notation in Equation (3.1), \( v(t) \) is continuous-time Gaussian white noise with covariance matrix \( \mathbb{E}\{v(t)v^T(t')\} = Q\delta(t-t') \), and \( w(t_k) \) is discrete-time Gaussian white noise with covariance matrix \( \mathbb{E}\{w(t_j)w^T(t_j)\} = R\delta_{jj} \). The system state vector \( x \) has the dimension \( n \) by 1, and the measurement vector has the dimension \( m \) by 1. To be able to use the higher-order filters considered here, it must be assumed that both \( f \) and \( h \) are twice differentiable in \( x \). Generally, the functions \( f \) and \( h \) can depend on other variables as well (an input signal \( u \), for example) and can be time-varying. For notational simplicity, these other dependencies are not explicitly stated.

All filters considered here are special cases of the general filtering algorithm in Table 3.1. In this table and in the remainder of this dissertation, all derivatives are understood as being computed with the latest available estimate inserted for the state vector, for example,

\[ \frac{\partial f}{\partial x^T} \text{ stands for } \left. \frac{\partial f}{\partial x^T} \right|_{x=x^T}. \]

The superscripts “−” and “+” indicate the time instants just before and immediately after a measurement is taken.

Equations for the additional terms \( b_p \), \( b_m \), and \( M \) in Equations (3.6), (3.8), and (3.10), respectively, are available in Gelb (1974), Jazwinski (1970), and Maybeck (1982). The terms \( b_p \) and \( b_m \) are called bias-correction terms, with the subscripts \( p \) and \( m \) referring to the propagation and the measurement equation, respectively.

For the various filters, these terms are given as

\[ b_{p,i} = \begin{cases} 0 \\ \frac{1}{2} \text{trace}\left( \frac{\partial^2 f_i}{\partial x \partial x^T} P_{xx} \right) = \frac{1}{2} \sum_{j,k=1}^{n} P_{x_{i,j}x_{i,k}} \frac{\partial^2 f_i}{\partial x_j \partial x_k} \end{cases} \]

for \( i = 1,2,...,n \), see Maybeck (1982), Equation (12-51) and Jazwinski (1970), Equations (9A.1), (9A.5), and (9A.7);

\[ b_{m,i} = \begin{cases} 0 \\ \frac{1}{2} \text{trace}\left( \frac{\partial^2 h_i}{\partial x \partial x^T} P_{xx} \right) = \frac{1}{2} \sum_{j,k=1}^{n} P_{x_{i,j}x_{i,k}} \frac{\partial^2 h_i}{\partial x_j \partial x_k} \end{cases} \]

for \( i = 1,2,...,n \), see Maybeck (1982), Equation (12-51) and Jazwinski (1970), Equations (9A.1), (9A.5), and (9A.7);
for $i = 1, 2, \ldots, m$, see Maybeck (1982), Equation (12-47) and Jazwinski (1970), Equations (9B.15), (9B.16), (9A.7), and (9A.5); and

$$
M = -b_m b_m^T
$$

(3.5)

$$
M_y = \frac{1}{2} \text{trace} \left( \frac{\partial^2 h_i}{\partial x^T} P_{xx} \frac{\partial^2 h_j}{\partial x^T} P_{xx} \right) = \sum_{k,i,p,q=1}^m \frac{\partial^2 h_i}{\partial x_p \partial x_q} P_{xx,ip} P_{xx,kq} \frac{\partial^2 h_j}{\partial x_p \partial x_q}
$$

GSF, $i, j = 1, \ldots, m$,

see Maybeck (1982), Equation (12-59) and Jazwinski (1970), Equation (9B.12).

Since these expressions are quite cumbersome, closed forms that do not require element-by-element computations are desirable. In the following section, it is shown how such closed, much simpler forms can be derived.

**Table 3.1** General filtering algorithm

**Approximate propagation of the state expectation and the error covariance from $t^*_k$ to $t^*_{k+1}$:**

$$
\hat{x}(t) = f(\hat{x}(t)) + b_p (\hat{x}(t), P_{xx}(t))
$$

(3.6)

$$
\hat{P}_{xx}(t) = \frac{\partial f}{\partial x^T} P_{xx}(t) + P_{xx}(t) \frac{\partial f^T}{\partial x} + Q
$$

(3.7)

**Predicted output and prediction error:**

$$
\hat{y}(t_{k+1}) = h(\hat{x}(t_{k+1})) + b_m (\hat{x}(t_{k+1}), P_{xx}(t_{k+1}))
$$

(3.8)

$$
e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1})
$$

(3.9)

**Approximate prediction error covariance matrix:**

$$
A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} + M(\hat{x}(t_{k+1}), P_{xx}(t_{k+1}))
$$

(3.10)

**Filter gain matrix:**

$$
K(t_{k+1}) = P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1})
$$

(3.11)

**State update:**

$$
\hat{x}(t^*_{k+1}) = \hat{x}(t^*_{k+1}) + K(t_{k+1}) e(t_{k+1})
$$

(3.12)

**Covariance update:**

$$
P(t^*_{k+1}) = (I_n - K(t_{k+1}) \frac{\partial h}{\partial x^T}) P_{xx}(t_{k+1})
$$

(3.13)
### 3.2 Rederivation of Higher-Order Terms

The starting point for the rederivation is the fact that the bias-correction term $b_p$ for the FBF, TSF, and GSF follows from expanding $f$ into a Taylor series around the expected value (the most recent estimate) and keeping terms up to second order, that is,

$$f(x) \equiv f(\hat{x}) + \frac{\partial f}{\partial x^T}(x - \hat{x}) + \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x^T}[(x - \hat{x}) \otimes (x - \hat{x})],$$

(3.14)

(see Rule 5 in Appendix A for the Taylor expansion of matrix functions). Then, taking the expectation and noting that

$$E(x - \hat{x}) = 0$$

(3.15)

and

$$E[(x - \hat{x}) \otimes (x - \hat{x})] = E[(x_1 - \hat{x}_1)(x_n - \hat{x}_n)] = \text{col } P_{xx}$$

(3.16)

results in

$$E f(x) \equiv f(\hat{x}) + \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x^T} E[(x - \hat{x}) \otimes (x - \hat{x})] = f(\hat{x}) + \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x^T} \text{col } P_{xx}.$$  

(3.17)

The measurement function $h$ can be expanded the same way. The two bias-correction terms $b_p$ and $b_m$ are thus given as

$$b_p = \begin{cases} 0 & \text{EKF}, \\ \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x^T} \text{col } P_{xx} & \text{FBF, TSF, GSF}, \end{cases}$$

(3.18)

and

$$b_m = \begin{cases} 0 & \text{EKF}, \\ \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \text{col } P_{xx} & \text{FBF, TSF, GSF}. \end{cases}$$

(3.19)

These expressions are clearly more compact than the corresponding terms in Equations (3.3) and (3.4).

The derivation of the term $M$ is slightly more complicated. First, note that the prediction error covariance matrix $A$ in Equation (3.10) should approximate $E \{(y - E y)(y - E y)^T\}$. 
Expanding $h$ into a Taylor series around the expectation of $x$ and keeping terms up to second order, as done for $f$ in Equation (3.14), leads to the expression

$$h - E[h] = \frac{\partial h}{\partial x^T}(x - \hat{x}) + \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T}[(x - \hat{x}) \otimes (x - \hat{x})] - \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \text{col } P_{xx}. \quad (3.20)$$

With this, the quadratic form $(h - E[h])(h - E[h])^T$ can be computed (a lengthy, but straightforward expression which is given in Appendix B). Then, taking the expectation and noting that all first and third moments are zero gives

$$E\{(h - E[h])(h - E[h])^T\} = \frac{\partial h}{\partial x^T} E\{(x - \hat{x})(x - \hat{x})^T\} \frac{\partial h^T}{\partial x}$$

$$- \frac{1}{4} \frac{\partial^2 h}{\partial x^T \partial x^T} \text{col } P_{xx} \text{col } P_{xx}^T \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T$$

$$+ \frac{1}{4} \frac{\partial^2 h}{\partial x^T \partial x^T} E\{[(x - \hat{x}) \otimes (x - \hat{x})][(x - \hat{x}) \otimes (x - \hat{x})]^T\} \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T. \quad (3.21)$$

Using Equation (3.19) to simplify the second term on the right-hand side of Equation (3.21),

$$E\{(h - E[h])(h - E[h])^T\} = \frac{\partial h}{\partial x^T} P_{xx} \frac{\partial h^T}{\partial x} - b_m b_m^T$$

$$+ \frac{1}{4} \frac{\partial^2 h}{\partial x^T \partial x^T} E\{[(x - \hat{x}) \otimes (x - \hat{x})][(x - \hat{x}) \otimes (x - \hat{x})]^T\} \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T \quad (3.22)$$

is obtained. The last term on the right-hand side of Equation (3.22) contains the matrix of fourth moments of the estimation error. For the TSF, this term is neglected; therefore, $M$ is given by the second term on the right-hand side of Equation (3.22), which is in accordance with Equation (3.5). For the GSF, the matrix of fourth moments is included.

To obtain an expression for $M$ for the GSF, the task now is to derive an expression for the matrix of fourth moments for a Gaussian distribution in terms of the covariance matrix. An elementwise expression for the fourth moments is given by Jazwinski (1970, p. 336). Here, the aim is to derive a compact expression that requires no elementwise computations. This can be done in a straightforward, but lengthy manner, namely, by taking the fourth (vector) derivative of the moment-generating function for a multivariable Gaussian distribution and evaluating the result at the origin (this also illustrates the applicability of matrix differential calculus to statistical problems). The moment-generating function for a zero-mean Gaussian distribution is given as (Maybeck 1979, p. 104)
\[ \Phi(\mu) = \exp\left(-\frac{1}{2} \mu^T P_{xx} \mu \right) , \quad (3.23) \]

and the matrix of fourth moments (denoted here by \( N \)) can be derived from it as

\[ N := \mathbb{E}\{ [(x - \hat{x}) \otimes (x - \hat{x})][(x - \hat{x}) \otimes (x - \hat{x})]^T \} = \frac{1}{j^4} \left. \frac{\partial^4 \Phi(\mu)}{\partial \mu^T \partial \mu \partial^2 \mu} \right|_{\mu = 0} . \quad (3.24) \]

The fourth derivative in Equation (3.24) can be computed by repeatedly applying the matrix product rule and the Kronecker product rule (for the full derivation, see Appendix B). This leads to

\[ N = (I_n \otimes U_m)(P_{xx} \otimes P_{xx}) + \text{col}(P_{xx}) (\text{col}(P_{xx})^T \right) . \quad (3.25) \]

If this expression is inserted into Equation (3.22), the terms containing \( \text{col}(P_{xx}) \) cancel, and an approximate expression for the prediction error covariance matrix for the GSF can be obtained with

\[ \mathbb{E} \{ (h - E\hat{h})(h - E\hat{h})^T \} \approx \frac{\partial \hat{h}}{\partial x^T} P_{xx} \frac{\partial \hat{h}^T}{\partial x} + \frac{1}{2} \frac{\partial^2 \hat{h}}{(I_n \otimes U_m)(P_{xx} \otimes P_{xx})} \left( \frac{\partial^2 \hat{h}}{\partial x^T \partial x^T} \right)^T . \quad (3.26) \]

Thus, from Equations (3.22) and (3.26), the term \( M \) is given as

\[ M = \begin{cases} 0 & \text{EKF, FBF,} \\ -b_m b_m^T & \text{TSF,} \\ \frac{1}{2} \frac{\partial^2 \hat{h}}{\partial x^T \partial x^T} \left( \frac{\partial^2 \hat{h}}{\partial x^T \partial x^T} \right)^T & \text{GSF}. \end{cases} \quad (3.27) \]

Evaluating this term for the GSF in an element-by-element fashion confirms that it is equivalent to the expression given by Equation (3.5). As for the bias-correction terms, this expression is more compact and elegant than the expression in Equation (3.5); furthermore, it does not have to be computed element by element. Table 3.2 summarizes the expressions for the higher-order terms.

Whether the new formulation of the higher-order terms is computationally superior to the formulation reported in the literature largely depends on the way the equations are finally implemented. For prototyping in Matlab, Kronecker products together with sparse matrix operations outperform standard for-loops. In actual implementation, for example, in a real-time application, the equations are usually “customized” in the sense that multiplication-by-zero operations are removed (that is, information about the structure of the large derivative
matrices is utilized) and redundant computations are avoided (for example, the lower triangular part of the covariance matrix contains redundant states and can be discarded). The effort that needs to be spent on this depends on how much time is available for the actual computation, which is determined by the sampling time. The expressions derived here are efficient in the sense that less customization is necessary if fast array computation routines and an efficient implementation of the Kronecker product are available.

### Table 3.2 Additional terms for the higher-order filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>$b_p$</th>
<th>$b_m$</th>
<th>$M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKF</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FBF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSF</td>
<td>$\frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x^T} \text{col } P_{xx}$</td>
<td>$\frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \text{col } P_{xx}$</td>
<td>$-b_m b_m^T$</td>
</tr>
<tr>
<td>GSF</td>
<td>$\frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} (\text{I}<em>{n_x} + U</em>{nn}) (P_{xx} \otimes P_{xx}) \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 4

Adaptive Filtering and Recursive Parameter Estimation

In this chapter, adaptive versions of the filters presented in the preceding chapter are developed. The motivation for this is twofold. Adaptation can be used (1) to improve the accuracy of the state estimates and (2) to determine unknown system parameters. In both cases, the filter is used as an adaptive predictor, and the filter parameters (which can include noise parameters and system parameters) are tuned such that a cost functional in the prediction errors is minimized. Thus, adaptive filters automatically give recursive parameter estimation algorithms, which is the main reason to consider them here.

4.1 Introduction and Chapter Overview

Both for linear and nonlinear filters, it is well known that the filter performance is largely affected by the choice of the covariance matrices $Q$ and $R$. For a linear system, the optimal filter results if $Q$ and $R$ are set to their true values, or, in some cases, if their ratio is correct (Alspach 1972); all filters with different settings are suboptimal. For nonlinear systems, however, this does not hold: Even if the disturbances were Gaussian and white, and the covariances $Q$ and $R$ were known (a rather unrealistic assumption), their true values would not necessarily give the best filter performance. “Pseudonoise,” that is, an increased process noise covariance matrix, is often used as it can improve filter performance (Maybeck 1982, p. 24). This occurs because the propagation and update equations of the filters are only approximations, whereas they are exact for linear systems. An increased value of the covariance (usually of the process noise) can reduce the effects of these approximations.

For linear systems, if $Q$ and $R$ are not known, several approaches to adaptive filtering (or so-called adaptive noise estimation) can be employed (Gelb 1974, pp. 317-20; Jazwinski 1970, pp. 311-18; Maybeck 1982, pp. 120-44; Mehra 1970, 1971, 1972; Sage and Hulsa 1969; Shellenbarger 1966, 1967; see also Mohamed and Schwarz [1999] for an application). The adaptation of the filter can be carried out by tuning the assumed noise characteristics (the covariance matrices $Q$ and $R$); however, unique estimates are only available when the number
of unknowns in $Q$ is less than $nm$ (this holds as different values of $Q$ and $R$ can lead to the same filter gain $K$, see Alspach [1972]). Alternatively, utilizing the fact that for linear time-invariant systems and stationary noise signals the gain of the Kalman Filter converges to a steady-state value, this steady-state filter gain $K$ can be estimated rather than the covariance matrices $Q$ and $R$. This corresponds to identifying the suboptimal, stationary Kalman Filter and provides, for example, the rationale behind the innovations-model-based variant of the recursive prediction error algorithm for linear state-space models (Ljung and Söderström 1983, pp. 122-29).

As mentioned above, for nonlinear systems, the problem of finding “optimal” noise covariance matrices remains even if the true values of these matrices were known. It is therefore desirable to have an adaptation algorithm that automatically determines a suitable setting for $Q$ and $R$. Except for the recursive prediction error method, the approaches to adaptive noise estimation for linear systems cannot be generalized to nonlinear systems. Furthermore, the filter gain for the nonlinear filters will remain time-varying even for time-invariant nonlinear systems and stationary noise signals because the state estimate and possibly an additional input signal enter the covariance propagation differential equation. Thus, an approach with an estimated constant gain is unjustifiable from a theoretical standpoint, and it can intuitively be expected that keeping the filter gain time-varying should give better performance (this will be illustrated using a simulation example in Chapter 6). Goodwin and Sin (1984) also indicate this:

\[ \ldots \text{some provision for parameter time variations is almost certainly necessary, since no single value of } \theta_i \text{ [the filter gain parameter vector] will give good performance under all conditions due to the restricted complexity nature of the predictor} \ldots. \]

The situation is further complicated since the filter gain may vary in time as fast as the states. Nevertheless, such a constant-gain approach has been proposed by Zhou and Blanke (1989), Ljungquist and Balchen (1993, 1994), and Ahmed (1994).

In this dissertation, the more general setting – where both the system states and additional unknown system parameters have to be estimated – is considered. The approaches in this chapter can treat both the estimation of “optimal” noise covariance matrices (or the optimal fixed filter gain) and system parameters in the same framework. This holds with the exception of the state-augmentation approach discussed in Section 4.2. Although this approach can only handle unknown system parameters, it is nevertheless included for reference.

This chapter is organized as follows. The concept of state augmentation is discussed in Section 4.2, where the algorithm for a state-augmented EKF is given. The other approaches to
4.2 State Augmentation

Adaptive filtering are based on sensitivity models, and the foundations of these approaches are outlined in Section 4.3. The sensitivity-based approach for adaptive constant-gain filters is presented in Section 4.4. As a new approach, an adaptive EKF is developed in Section 4.5; furthermore, it is argued that this approach avoids some of the shortcomings of the two previous approaches. Adaptive higher-order filters are presented in Section 4.6. The relationship between the least-squares cost functional used for developing the algorithms and the maximum-likelihood cost functional is explored in Section 4.7. It is shown that the algorithms can readily be extended to a maximum-likelihood method. Although there is different reasoning behind the sensitivity-based approaches and the state-augmentation concept, the latter can be interpreted as a sensitivity-based approach with a simplified sensitivity model. This interpretation was suggested by Ljung and Söderström (1983, Appendix 3.C) and is examined in greater detail in Section 4.8. The results of this discussion are interesting as they show that both algorithms do not differ much and that the difference has a straightforward explanation. This weakens possible claims that one algorithm is better than the other, as put forward, for example, by Chu et al. (1996). Finally, the relationship to other approaches is explored in Section 4.9. This chapter is summarized in Section 4.10.

4.2 State Augmentation

If either the state transition function $f$ or the measurement function $h$ or both depend on additional quantities such as system parameters, disturbances, or unknown input signals, it is possible to build a filter that also gives estimates of these unknowns by implementing an approach called state augmentation (Gelb 1974, p. 349; Grewal and Andrews 1993, Example 5.3). In this approach, these unknown quantities are usually modeled either as constant or as Brownian motion (integrated white noise), although more general time variations are possible.

Then, a new state vector is formed as

$$\tilde{x} = \begin{bmatrix} x \\ p \end{bmatrix},$$

where all additional unknown quantities are collected in the parameter vector $p$ of dimension $s$ by 1, and the system equations are rewritten for the augmented state vector as

$$\tilde{x} = \begin{bmatrix} f(x, p, u) \\ 0 \end{bmatrix} + \begin{bmatrix} v \\ v_p \end{bmatrix} = \tilde{f}(\tilde{x}, u) + \tilde{v}$$

and

$$y(t_k) = h(x(t_k), p(t_k), u(t_k)) + w(t_k) = \tilde{h}(\tilde{x}(t_k), u(t_k)) + w(t_k),$$

where $x(t_k)$ and $p(t_k)$ are the state and parameter vectors at time $t_k$, respectively.
where the dependence on an input signal $u$ has been included (the function $\tilde{h}$ differs from $h$ only formally through a change in the argument). This concept was suggested by Cox (1964) for linear and nonlinear systems:

\[ \text{... one can easily include as state variables unknown parameters, either constant or randomly varying. The estimation of such parameters constitutes a form of the identification problem, a central problem in the theory of adaptive control. Parameter estimation will be treated as a special case of estimation of state variables.} \]

Since then, the state-augmented EKF has become a popular parameter estimation method for linear systems (Nelson and Stear 1976; Yoshimura et al. 1981; Young 1981). Its convergence properties were analyzed by Ljung (1979), who suggested the recursive prediction error method to overcome the convergence problems associated with the state-augmentation approach. State augmentation is also used in nonlinear observers together with Lyapunov theory (Friedland 1997; Haessig and Friedland 1997; Lyashevskiy and Chen 1996) or in the so-called Extended Luenberger Observer (Du et al. 1995).

Assuming a covariance matrix \( E\{v_p(t)\delta(t-t')\} = Q_{pp} \delta(t-t') \) for the "pseudonoise" \( v_p \) acting on the time derivative of the parameter vector gives a covariance matrix for the augmented process noise $\tilde{v}$ as

\[
E\{	ilde{v}(t)\tilde{v}^T(t')\} = \begin{bmatrix} Q & 0 \\ 0 & Q_{pp} \end{bmatrix} \delta(t-t').
\]  

Then, any of the filters discussed in the previous chapter can be implemented for this augmented system. The EKF is the most common choice. In principle, it is possible to use higher-order filters as well, but this has a serious drawback: Usually, the covariance matrix of the pseudonoise is used as a design parameter that is tuned to give fast convergence and good tracking capabilities by preventing the covariance matrix of the estimation error in $p$ from becoming too small. Often, additional means of "covariance management" (Wellstead and Zarrop 1991) are also employed. Thus, the covariance matrix of the estimation error in $p$ will be kept larger than its "theoretical" value, and, consequently, the bias correction terms given in Equations (3.18) and (3.19) will become unrealistically large. The filter thereby overcompensates for the nonlinear effects of the uncertainty in the estimates and as a result gives bad state estimates or becomes unstable (particularly for large sampling times).

The EKF for the system described by Equations (4.2) and (4.3) is given in Table 4.1. In all filtering algorithms considered here, derivatives are understood as being computed with the latest available state and parameter estimates.
### Table 4.1 State-augmented Extended Kalman Filter

<table>
<thead>
<tr>
<th>State estimate propagation</th>
<th>( \dot{x} = f(\dot{x}, u, \dot{p}(t_k)) ) (4.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariance propagation</td>
<td>( P_{xx} = \frac{\partial f}{\partial x} P_{xx} + \frac{\partial f^T}{\partial x} P_{xp} + \frac{\partial f}{\partial p} P_{xp} + \frac{\partial f^T}{\partial p} P_{pp} + Q ) (4.6)</td>
</tr>
<tr>
<td></td>
<td>( \dot{P}<em>{xp} = \frac{\partial f}{\partial x} P</em>{xp} + \frac{\partial f}{\partial p} P_{pp} ) (4.7)</td>
</tr>
<tr>
<td></td>
<td>( \dot{P}<em>{pp} = Q</em>{pp} ) (4.8)</td>
</tr>
<tr>
<td>Predicted output</td>
<td>( \hat{y}(t_{k+1}) = h(\hat{x}(t_{k+1}^-), u(t_{k+1}^-), \hat{p}(t_k)) ) (4.9)</td>
</tr>
<tr>
<td>Prediction error</td>
<td>( e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1}) ) (4.10)</td>
</tr>
<tr>
<td>Approximate prediction</td>
<td>( A(t_{k+1}) = R + \frac{\partial h}{\partial x} P_{xx} (t_{k+1}^-) \frac{\partial h^T}{\partial x} + \frac{\partial h}{\partial p} P_{xp} (t_{k+1}^-) \frac{\partial h^T}{\partial p} ) (4.11)</td>
</tr>
<tr>
<td>error covariance matrix</td>
<td>( K(t_{k+1}) = (P_{xx}(t_{k+1}^-) \frac{\partial h^T}{\partial x} + P_{xp}(t_{k+1}^-) \frac{\partial h^T}{\partial p})A^{-1}(t_{k+1}) ) (4.12)</td>
</tr>
<tr>
<td>Filter gain matrices</td>
<td>( L(t_{k+1}) = (P_{xp}(t_{k+1}^-) \frac{\partial h^T}{\partial x} + P_{pp}(t_{k+1}^-) \frac{\partial h^T}{\partial p})A^{-1}(t_{k+1}) ) (4.13)</td>
</tr>
<tr>
<td>State update</td>
<td>( \Delta \hat{x}(t_{k+1}) = K(t_{k+1})e(t_{k+1}) ) (4.14)</td>
</tr>
<tr>
<td></td>
<td>( \hat{x}(t_{k+1}^+) = \hat{x}(t_{k+1}^-) + \Delta \hat{x}(t_{k+1}) ) (4.15)</td>
</tr>
<tr>
<td>Parameter update</td>
<td>( \Delta \hat{p}(t_{k+1}) = L(t_{k+1})e(t_{k+1}) ) (4.16)</td>
</tr>
<tr>
<td></td>
<td>( \hat{p}(t_{k+1}) = \hat{p}(t_k) + \Delta \hat{p}(t_{k+1}) ) (4.17)</td>
</tr>
<tr>
<td>Covariance updates</td>
<td>( P_{xx}(t_{k+1}^+) = P_{xx}(t_{k+1}^-) - K(t_{k+1})A(t_{k+1})K^T(t_{k+1}) ) (4.18)</td>
</tr>
<tr>
<td></td>
<td>( P_{xp}(t_{k+1}^+) = P_{xp}(t_{k+1}^-) - K(t_{k+1})A(t_{k+1})L^T(t_{k+1}) ) (4.19)</td>
</tr>
<tr>
<td></td>
<td>( P_{pp}(t_{k+1}^+) = P_{pp}(t_{k+1}^-) - L(t_{k+1})A(t_{k+1})L^T(t_{k+1}) ) (4.20)</td>
</tr>
</tbody>
</table>
A drawback of the state-augmented EKF as a parameter estimator for linear systems was pointed out by Ljung (1979), who showed that this algorithm lacks desirable converge properties and concluded:

It is . . . unrealistic to assume that the noise structure of the system is known, while the dynamics are unknown. Therefore, if the noise characteristics of the model is \[sic\] chosen ad hoc (as, apparently, is usually done) then the system parameter estimates will in general be biased. We have consequently arrived at the perhaps trivial observation that the cause of the bias does not lie in the EKF-method itself, but comes from incorrect noise assumptions associated with the model.

Since unknown parameters in the noise covariance matrices cannot be interpreted as states, they cannot be estimated through state augmentation. Thus, parameter estimates will always depend on the noise covariance matrices. Simulation studies can be used to analyze this dependence if a realistic or typical set of values for the unknown system parameters is available.

Furthermore, as was also pointed out by Ljung (1979), the EKF can be interpreted as a gradient approach to the estimation of the unknown parameters, where an approximation is used to compute the gradient. This is discussed in further detail in Section 4.8. It turns out that this simplified gradient computation, which, by itself, does not seem to have too much of an effect on the parameter estimates, has to be replaced by a “corrected” gradient computation in order to allow covariance parameters to be estimated as well.

State augmentation can thus only be recommended in two cases: (1) if auxiliary parameters, disturbances, or “dummy state variables” that are introduced to give improved estimates of the true system states, but are not of too much independent interest, have to be estimated or (2) if simulation studies indicate that the noise dependence is not too severe and noise settings have been determined that give (almost) unbiased estimates for a range of typical parameter values. Since the EKF is the least complex algorithm among the ones discussed in this chapter, it could also be used for a first analysis and for obtaining starting values for the more complex algorithms treated in the following sections.

4.3 Sensitivity-Based Approaches to Parameter Estimation

The state-augmented filters of the previous section treat the parameters as stochastic variables and estimate them as auxiliary states. Another possible approach is the so-called prediction error method, which makes use of adaptive predictors: a parameterized predictor generates predictions of the system’s output at the next sampling instants. The predicted system output is then compared to the actual measurement, and the parameters of the predictor are corrected
such that the prediction errors are minimized (usually in the sense of a quadratic criterion). As pointed out above, this constitutes an adaptive filtering approach.

The parameters are commonly adjusted using a stochastic-approximation-type algorithm; this requires the gradient of the system output, which can be computed from the output prediction equation. If an EKF is used, taking the gradient of the output prediction equation (3.8) with respect to the parameter vector results in

$$
\frac{d\hat{y}^T(t_{k+1})}{dp} = \left( \frac{\partial h}{\partial x^T} \frac{d\hat{x}(t_{k+1})}{dp^T} + \frac{\partial h}{\partial p^T} \right)^T.
$$

To evaluate this expression, the so-called sensitivity $d\hat{x}(t_{k+1})/dp^T$ is necessary. Although an exact expression for this sensitivity is usually unavailable, its propagation and update equations can be derived. Applying the derivative operator $d/\hat{p}^T$ to the propagation equation (4.5) and changing the order of the differentiation, which, strictly speaking, is only allowed for constant parameters (see Eykhoff 1974, p. 348), yields

$$
\frac{d\hat{x}}{d\hat{p}^T} = \frac{d}{dx^T} \frac{d\hat{x}}{d\hat{p}^T} + \frac{d}{dp^T}.
$$

Given a starting value, the sensitivity can thus be propagated between sampling instants. More complications arise for the measurement update, Equation (4.15): Applying the derivative operator results in

$$
\frac{d\hat{x}(t_{k+1})}{dp^T} = \frac{d\hat{x}(t_{k+1})}{dp^T} + \frac{dK(t_{k+1})}{dp^T} e(t_{k+1}),
$$

which can be decomposed using the matrix product rule as

$$
\frac{d\hat{x}(t_{k+1})}{dp^T} = \frac{d\hat{x}(t_{k+1})}{dp^T} + K(t_{k+1}) \frac{de(t_{k+1})}{dp^T} + \frac{dK(t_{k+1})}{dp^T} (I_x \otimes e(t_{k+1})).
$$

Noting that the gradient of the prediction error is the negative gradient of the prediction yields

$$
\frac{d\hat{x}(t_{k+1})}{dp^T} = \frac{d\hat{x}(t_{k+1})}{dp^T} - K(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{dp^T} + \frac{dK(t_{k+1})}{dp^T} (I_x \otimes e(t_{k+1})).
$$

The term $dK(t_{k+1})/\hat{p}^T$ on the right-hand side of Equation (4.25) causes the algorithm to become quite complex: From the general expression for the filter gain

$$
K(t_{k+1}) = P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1}) ,
$$
it is evident that the evaluation of $dK(t_{k+1})/d\hat{p}^T$ requires the sensitivity of the estimation error covariance matrix, that is, $dP_{kk}/d\hat{p}^T$. To obtain this sensitivity, a propagation equation and an update equation have to be developed. The result, a recursive prediction error algorithm for a general nonlinear state-space model, is devised in Section 4.5. Before this algorithm is presented, a method to avoid this increase in complexity is investigated in the next section.

It is interesting that if the term $dK(t_{k+1})/d\hat{p}^T$ is neglected (along with some other, minor modifications), the algorithm resembles the state-augmented EKF; further elaboration of this topic follows in Section 4.8.

4.4 Adaptive Constant-Gain Filtering

Evaluating the term $dK(t_{k+1})/d\hat{p}^T$ by taking the derivative of Equation (4.26) greatly complicates the algorithm. Therefore, even for the linear case, this step is usually avoided. Exceptions are Maybeck (1982, Chapter 10), who carried out all the derivations in detail to develop his recursive maximum-likelihood online identification method for linear discrete-time systems, and El-Fattah (1983), who suggested an adaptive Kalman Filter with a least-squares criterion. Mehra and Tyler (1973) derived a sensitivity model for offline maximum-likelihood identification for nonlinear continuous-discrete systems. In both cases, the necessary derivatives were computed elementwise. Jakoby and Pandit (1987) outlined a similar approach for nonlinear, discrete-time, single-input-single-output systems. Again, the derivatives were computed elementwise. The noise statistics were assumed to be known. For the example considered, however, the derivative of the covariance matrix was not used.

It has been pointed out above that for linear systems the filter gain matrix will converge to a steady-state value. As a result, instead of the covariance matrices $Q$ and $R$, this steady-state filter gain can be independently parameterized and estimated. This approach is the well-known recursive prediction error method applied to an innovations model (Ljung and Söderström 1983, Section 3.8.2). An extension of this to nonlinear systems was suggested by Zhou and Blanke (1986, 1989), Ljungquist and Balchen (1993, 1994), and Ahmed (1994). In the latter two references, an EKF was applied; therefore, these approaches correspond to making the so-called Constant-Gain Extended Kalman Filter (CGEKF, Safonov and Athans 1978) adaptive. Zhou and Blanke (1989) made use of a First-Order Bias-Corrected Filter (with a constant filter gain). It is surprising that in their approach the filter gain was assumed constant since the algorithm would have provided all the ingredients for a time-varying filter gain. They also only demonstrated their algorithm in an extremely simple simulation example
(one state, two unknown parameters, and no process noise). Furthermore, they did not give
details for the computation of the required derivatives.

As previously mentioned, the filter gain for the nonlinear filters discussed in the previous
chapter generally remains time-varying unless the system is in an equilibrium state (Maybeck
1982, p. 57); however, such a case would not be of too much interest in a system identifi-
cation setting. Thus, it is to be expected that a filter with a time-varying gain exhibits better
performance than a filter with a gain that is restricted to being constant. Furthermore, the
performance difference between an optimal constant-gain filter and a suboptimal, detuned
filter with a time-varying gain may be small, or the detuned filter might even perform better
than the constant-gain filter if the former is not too severely detuned.

In this section, only the CGEKF is treated. The extension to a higher-order filter is
straightforward but requires further derivatives. It turns out though that the computational
simplification that results from assuming a constant filter gain, namely, the fact that the error
covariance and its sensitivity are no longer needed, is lost if a higher-order filter is applied.
This loss occurs because the error covariance matrix is needed in the computation of the bias-
correction terms, and, consequently, the covariance sensitivity is necessary for the derivatives
of these terms. If the covariance matrix and its sensitivity are computed anyway to evaluate
the bias-correction terms and their derivatives, it does not make sense not to use these quanti-
ties to compute a time-varying filter gain. A possible and consequent extension would be to
assume the bias-correction terms as constant, which was suggested by Bohn and Unbehauen
(1999a) for a discrete-time filter, but this is not pursued here.

In the adaptive CGEKF, both the unknown system parameter vector \( \hat{p} \) and the elements of
the gain matrix \( \hat{K} \) have to be estimated. The algorithm is formulated such that the corre-
sponding column vector \( \text{col} \hat{K} \) is estimated. The sensitivity update equation (4.25) can then be
rewritten separately for \( \hat{p} \) and \( \text{col} \hat{K} \) as

\[
\frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} = \frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} - \hat{K}(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{d\hat{p}^T} + \frac{d\hat{K}(t_{k+1})}{d\hat{p}^T} (I_m \otimes e(t_{k+1})) \tag{4.27}
\]

and

\[
\frac{d\hat{x}(t_{k+1})}{d (\text{col} \hat{K})^T} = \frac{d\hat{x}(t_{k+1})}{d (\text{col} \hat{K})^T} - \hat{K}(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{d (\text{col} \hat{K})^T} + \frac{d\hat{K}}{d (\text{col} \hat{K})^T} (I_m \otimes e(t_{k+1})) \tag{4.28}
\]

Since in Equation (4.27) \( \hat{K} \) has been replaced by \( \hat{K} \), which is independent of \( \hat{p} \), the term
\( d\hat{K}/d\hat{p}^T \) vanishes, and Equation (4.27) becomes
\[ \frac{d\hat{x}(t_{k+1}^+)}{d\hat{p}^T} = \frac{d\hat{x}(t_{k+1}^-)}{d\hat{p}^T} - \hat{K}(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{d\hat{p}^T}. \]  

(4.29)

In Equation (4.28), the derivative of the filter gain matrix with respect to its own column vector is given as

\[ \frac{\partial \hat{K}}{\partial (\text{col } \hat{K})^T} = \begin{bmatrix} E_{11}^{(\text{norn})} & \ldots & E_{n1}^{(\text{norn})} & \ldots & E_{m1}^{(\text{norn})} \end{bmatrix}. \]  

(4.30)

With this expression, the last term on the right-hand side of Equation (4.28) can be simplified to

\[ \frac{d\hat{K}(t_{k+1})}{d (\text{col } \hat{K})^T} (I_{mn} \otimes e(t_{k+1})) = e^T(t_{k+1}) \otimes I_n, \]  

(4.31)

so that Equation (4.28) becomes

\[ \frac{d\hat{x}(t_{k+1}^+)}{d (\text{col } \hat{K})^T} = \frac{d\hat{x}(t_{k+1}^-)}{d (\text{col } \hat{K})^T} - \hat{K}(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{d (\text{col } \hat{K})^T} + e^T(t_{k+1}) \otimes I_n. \]  

(4.32)

Since the filter gain matrix \( \hat{K} \) is estimated rather than computed, the covariance matrix of the state estimation error is no longer needed, and both its propagation and update equation can be discarded. With the introduction of an augmented parameter vector as

\[ \tilde{p} = \begin{bmatrix} p \\ \text{col } K \end{bmatrix}, \]

the algorithm is given in Table 4.2. For notational convenience, the augmented parameter vector has been used for some of the update equations, whereas the other equations have been written out separately for the system and the gain parameters. In the first column, it is indicated whether the corresponding equations belong to the Constant-Gain Extended Kalman Filter (CGEKF), the sensitivity model (SM), or the parameter estimation algorithm (RPE).
### Table 4.2 Adaptive Constant-Gain Extended Kalman Filter

<table>
<thead>
<tr>
<th>C</th>
<th>G</th>
<th>E</th>
<th>K</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>State estimate propagation</td>
<td>$\hat{x} = f(\hat{x}, u, \hat{p}(t_k))$</td>
<td>(4.33)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensitivity propagation</td>
<td>$\frac{d\hat{x}}{d\hat{p}^T} = \frac{\partial f}{\partial x^T} \frac{d\hat{x}}{d\hat{p}^T} + \frac{\partial f}{\partial p^T}$</td>
<td>(4.34)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predicted output</td>
<td>$\hat{y}(t_{k+1}) = h(\hat{x}(t_{k+1}), u(t_{k+1}), \hat{p}(t_k))$</td>
<td>(4.36)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prediction error</td>
<td>$e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1})$</td>
<td>(4.37)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradients of prediction</td>
<td>$\frac{d\hat{y}^T(t_{k+1})}{d\hat{p}} = \left( \frac{\partial h}{\partial x^T} \frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} + \frac{\partial h}{\partial p^T} \frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} \right)^T$</td>
<td>(4.38)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{d\hat{y}^T(t_{k+1})}{d\text{col } \hat{K}} = \left( \frac{\partial h}{\partial x^T} \frac{d\hat{x}(t_{k+1})}{d\text{col } \hat{K}} \right)^T$</td>
<td>(4.39)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{d\hat{y}^T(t_{k+1})}{d\hat{\hat{p}}} = \left[ \begin{array}{c} \frac{d\hat{y}^T(t_{k+1})}{d\hat{p}} \ \frac{d\hat{y}^T(t_{k+1})}{d\text{col } \hat{K}} \end{array} \right]$</td>
<td>(4.40)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter adaptation gain</td>
<td>$S(t_{k+1}) = \lambda A(t_{k+1}) + \frac{d\hat{y}^T(t_{k+1})}{d\hat{p}} P_{pp}(t_k) \frac{d\hat{y}^T(t_{k+1})}{d\hat{p}}$</td>
<td>(4.41)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$L(t_{k+1}) = P_{pp}(t_k) \frac{d\hat{y}^T(t_{k+1})}{d\hat{p}} S^{-1}(t_{k+1})$</td>
<td>(4.42)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter update</td>
<td>$\Delta \hat{p}(t_{k+1}) = L(t_{k+1}) e(t_{k+1})$</td>
<td>(4.43)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\hat{p}(t_{k+1}) = \hat{p}(t_k) + \Delta \hat{p}(t_{k+1})$,</td>
<td>(4.44)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with $\hat{\hat{p}} = \begin{bmatrix} \hat{p} \ \text{col } \hat{K} \end{bmatrix}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In contrast to the state-augmented EKF of Section 4.2, the state and the parameter estimates in the CGEKF are no longer updated simultaneously. Instead, the new estimate of the filter gain obtained from Equation (4.44) is used to compute the state update in Equation (4.49). The matrix $A$ in Equation (4.41) should be the covariance matrix of the true prediction errors; however, this matrix is usually unknown. In the formulation used here, $A$ is interpreted as a weighting matrix. Alternatively, $A$ could be estimated from the data, see Equation (3.63) in Ljung and Söderström (1983) and Equation (29) in Ahmed (1994). Ljungquist and Balchen (1994), however, reported that a fixed matrix should be used since “simulation experience indicates that the robustness is reduced considerably without a corresponding improvement of the transient response when the estimate . . . is used.” A forgetting factor $\lambda$ can be used in Equations (4.41) and (4.45) to “manage the covariance matrix.” The algorithm can be seen as the counterpart to Gavel and Azevedo’s (1982) algorithm for a linear state-space model applied to a nonlinear model.

### 4.5 Adaptive Extended Kalman Filtering

In the last section, a simplification for the derivation of an adaptive filtering algorithm has been discussed. The resulting algorithm does not have to take the covariance sensitivity into account as it estimates a constant (or, if forgetting is employed, a slowly time-varying) filter gain. Consequently, it also does not require the covariance matrix. It has been argued above that such an approach is theoretically justified only for linear systems. Therefore, a new algo-
rithm is developed in this section by rigorously deriving the full sensitivity model that is needed to make the EKF adaptive. The main reason why this has not been done in this general form seems to be the complexity that arises from computing the necessary sensitivity equations. By using matrix differential calculus, however, this complication can be alleviated. In this manner, the algorithm can be developed in a closed, “implementation-ready” formulation. No elementwise derivatives have to be computed, and no bookkeeping over indices is necessary.

The starting point for this derivation is a standard EKF as given in Chapter 3. The sensitivity model for this filter has already been developed in Section 4.3 up to the point where the term \( \frac{dK(t_{k+1})}{dp^T} \) arises. This derivation is now continued.

The derivative of the filter gain in Equation (4.26) can be computed with the help of the matrix product rule as

\[
\frac{dK(t_{k+1})}{dp^T} = \frac{dP_{xx}(t_{k+1})}{dp^T} \left( I_x \otimes \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1}) \right) + P_{xx}(t_{k+1}) \frac{d^2 h^T}{dp^T dx} \left( I_x \otimes A^{-1}(t_{k+1}) \right) + P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} \frac{dA^{-1}(t_{k+1})}{dp^T}.
\]

The derivative in the second term on the right-hand side of Equation (4.50) has to be evaluated using the matrix chain rule, since \( h \) depends on \( x \), which in turn depends on \( p \). Some other second derivatives of this type are needed below; therefore, the derivatives

\[
\frac{d^2 f}{dp^T dx^T} = \frac{d}{dp^T} \frac{df}{dx^T} = \frac{\partial^2 f}{\partial x^T} \left( \frac{dx}{dp^T} \otimes I_n \right) + \frac{\partial^2 f}{\partial p^T dx^T},
\]

\[
\frac{d^2 f^T}{dp^T dx^T} = \frac{d}{dp^T} \frac{df^T}{dx^T} = \frac{\partial^2 f^T}{\partial x^T} \left( \frac{dx}{dp^T} \otimes I_n \right) + \frac{\partial^2 f^T}{\partial p^T dx^T},
\]

\[
\frac{d^2 h}{dp^T dx^T} = \frac{d}{dp^T} \frac{dh}{dx^T} = \frac{\partial^2 h}{\partial x^T} \left( \frac{dx}{dp^T} \otimes I_n \right) + \frac{\partial^2 h}{\partial p^T dx^T},
\]

and

\[
\frac{d^2 h^T}{dp^T dx^T} = \frac{d}{dp^T} \frac{dh^T}{dx^T} = \frac{\partial^2 h^T}{\partial x^T} \left( \frac{dx}{dp^T} \otimes I_m \right)^T + \frac{\partial^2 h^T}{\partial p^T dx^T}.
\]

are introduced. All the terms on the far left-hand side of Equations (4.51) to (4.54) are not analytical derivatives. Rather, they are quantities that will be numerically evaluated in the
algorithm. Only the partial derivatives on the far right-hand side need to be provided in analytical form.

To compute $\frac{dK(t_{k+1})}{dp^T}$ from Equation (4.50), the term $\frac{dA^{-1}(t_{k+1})}{dp^T}$ is needed. From the derivative of the inverse of a matrix (see Rule 7 in Appendix A), this term follows as

$$
\frac{dA^{-1}(t_{k+1})}{dp^T} = -A^{-1}(t_{k+1}) \frac{dA(t_{k+1})}{dp^T} \left( I_s \otimes A^{-1}(t_{k+1}) \right).
$$

(4.55)

With $A(t_{k+1})$ given as

$$
A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} P_{xx} (t_{k+1}^-) \frac{\partial h^T}{\partial x},
$$

(4.56)

its derivative can be computed as

$$
\frac{dA(t_{k+1})}{dp^T} = \frac{\partial R}{\partial p^T} + \frac{d^2 h}{dp^T dx^T} \left( I_s \otimes P_{xx} (t_{k+1}^-) \frac{\partial h^T}{\partial x} \right) + \frac{\partial h}{\partial x^T} \frac{dP_{xx}(t_{k+1}^-)}{dp^T} \left( I_s \otimes \frac{\partial h^T}{\partial x} \right)
$$

$$
+ \frac{\partial h}{\partial x^T} P_{xx}(t_{k+1}^-) \frac{d^2 h^T}{dp^T dx}.
$$

(4.57)

In Equations (4.50) and (4.57), the covariance sensitivity matrix $\frac{dP_{xx}}{dp^T}$ is required. Analogously to the state sensitivity, a propagation and an update equation for this sensitivity have to be derived by taking the derivative of the propagation and the update equation, respectively. Taking the derivative of the covariance propagation equation

$$
\dot{P}_{xx} = \frac{df}{dx^T} P_{xx} + P_{xx} \frac{df^T}{dx} + Q
$$

(4.58)

yields the covariance sensitivity propagation equation

$$
\frac{d\dot{P}_{xx}}{dp^T} = \frac{d^2 f}{dp^T dx^T} (I_s \otimes P_{xx}) + \frac{df}{dx^T} \frac{dP_{xx}}{dp^T} + \frac{dP_{xx}}{dp^T} \left( I_s \otimes \frac{df^T}{dx} \right) + P_{xx} \frac{d^2 f^T}{dp^T dx} + \frac{dQ}{dp^T}.
$$

(4.59)

Taking the derivative of the covariance update equation

$$
P_{xx}(t_{k+1}^+) = \left( I_n - K(t_{k+1}) \frac{\partial h}{\partial x^T} \right) P_{xx}(t_{k+1}^-)
$$

(4.60)

gives the covariance sensitivity update equation

$$
\frac{dP_{xx}(t_{k+1}^+)}{dp^T} = \left( I_n - K(t_{k+1}) \frac{\partial h}{\partial x^T} \right) \frac{dP_{xx}(t_{k+1}^-)}{dp^T} - K(t_{k+1}) \frac{d^2 h}{dp^T dx^T} \left( I_s \otimes P_{xx}(t_{k+1}^-) \right)
$$

$$
- \frac{dK(t_{k+1})}{dp^T} \left( I_s \otimes \frac{\partial h}{\partial x^T} P_{xx}(t_{k+1}^-) \right),
$$

(4.61)
which concludes the derivation of the algorithm.

An approximate version that is computationally much simpler is obtained if the covariance sensitivity is neglected in Equations (4.50) and (4.57). Then, the covariance sensitivity propagation and update equations are not needed. This would result in the algorithm that Jakoby and Pandit (1987) applied to their example. Neglecting the covariance sensitivity, however, means that parameters in $Q$ can no longer be estimated since the gradient of $Q$ only enters in the covariance sensitivity propagation equation. Thus, a bias resulting from wrongly assumed noise statistics could not be removed with this simplified algorithm, and it is unclear whether this algorithm offers any general advantages over state augmentation. Jakoby and Pandit (1987) achieved a reduction in computational requirements over state augmentation but still reported a higher accuracy.

The adaptive EKF is summarized in Table 4.3. The equations of the algorithm are ordered in the way that is most suitable for implementation: all terms can be computed with the latest available estimates, and terms do not have to be recomputed. Again, in the first column of Table 4.3, it is identified to which part of the algorithm the equations belong.
<table>
<thead>
<tr>
<th>State estimate propagation</th>
<th>$\dot{x} = f(\hat{x}, u, \hat{p}(t_k))$ (4.62)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Covariance propagation</td>
<td>$\dot{P}<em>{xx} = \frac{\partial f}{\partial x^T} P</em>{xx} + P_{xx} \frac{\partial f^T}{\partial x} + Q(\hat{p}(t_k))$ (4.63)</td>
</tr>
<tr>
<td>Sensitivity propagation</td>
<td>$\frac{\partial \dot{x}}{\partial p} = \frac{\partial f}{\partial x^T} \frac{\partial \dot{x}}{\partial p} + \frac{\partial f}{\partial p^T}$ (4.64)</td>
</tr>
<tr>
<td></td>
<td>$\frac{\partial \dot{P}<em>{xx}}{\partial p} = \frac{\partial^2 f}{\partial p^T \partial x^T} (I \otimes P</em>{xx}) + \frac{\partial f}{\partial x^T} \frac{\partial P_{xx}}{\partial p} + \frac{\partial P_{xx}}{\partial p} \left( I \otimes \frac{\partial f^T}{\partial x} \right)$ (4.65)</td>
</tr>
<tr>
<td></td>
<td>$+ P_{xx} \frac{\partial^2 f^T}{\partial p^T \partial x} + \frac{\partial Q}{\partial p^T}$</td>
</tr>
<tr>
<td>Predicted output</td>
<td>$\hat{y}(t_{k+1}) = h(\hat{x}(t_{k-1}), u(t_{k-1}), \hat{p}(t_k))$ (4.68)</td>
</tr>
<tr>
<td>Prediction error</td>
<td>$e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1})$ (4.69)</td>
</tr>
<tr>
<td>Approximate prediction error covariance matrix</td>
<td>$A(t_{k+1}) = R(\hat{p}(t_k)) + \frac{\partial h}{\partial x^T} P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x}$ (4.70)</td>
</tr>
<tr>
<td>Filter gain</td>
<td>$K(t_{k+1}) = P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1})$ (4.71)</td>
</tr>
<tr>
<td>Gradient of prediction</td>
<td>$\frac{\partial \hat{y}^T(t_{k+1})}{\partial \hat{p}} = \left( \frac{\partial h}{\partial x^T} \frac{\partial \hat{x}(t_{k+1})}{\partial p} + \frac{\partial h}{\partial p^T} \right)^T$ (4.72)</td>
</tr>
</tbody>
</table>
### Table 4.3 continued

<table>
<thead>
<tr>
<th>Derivative of prediction error covariance matrix</th>
<th></th>
</tr>
</thead>
</table>
| \[
\frac{dA(t_{k+1})}{dp^\top} = \frac{\partial R}{\partial p^\top} + \frac{d^2 h}{dp^\top} dx^\top \left( I_n \otimes P_{xx}(t_{k+1}) \frac{\partial h^\top}{\partial x} \right) + \frac{\partial h}{\partial x^\top} P_{xx}(t_{k+1}) \frac{d^2 h^\top}{dp^\top} dx + P_{xx}(t_{k+1}) \frac{\partial h^\top}{\partial x} \frac{d^2 h^\top}{dp^\top} dx
\] (4.73) |
| with                                                                 |
|                                                                 |
| \[
\frac{d^2 h}{dp^\top} dx^\top = \frac{\partial ^2 h}{\partial x^\top \partial x^\top} \left( \frac{dx}{dp} \otimes I_n \right) + \frac{\partial ^2 h}{\partial p \partial x^\top} (4.74)
\] |
|                                                                 |
| \[
\frac{d^2 h^\top}{dp^\top} dx = \frac{\partial ^2 h^\top}{\partial x^\top \partial x} \left( \frac{dx}{dp} \otimes I_m \right) + \frac{\partial ^2 h^\top}{\partial p \partial x} (4.75)
\] |

<table>
<thead>
<tr>
<th>Derivative of inverse of prediction error covariance matrix</th>
</tr>
</thead>
</table>
| \[
\frac{dA^{-1}(t_{k+1})}{dp^\top} = -A^{-1}(t_{k+1}) \frac{dA(t_{k+1})}{dp^\top} \left( I_n \otimes A^{-1}(t_{k+1}) \right)
\] (4.76) |

<table>
<thead>
<tr>
<th>Derivative of filter gain</th>
</tr>
</thead>
</table>
| \[
\frac{dK(t_{k+1})}{dp^\top} = \frac{dP_{xx}(t_{k+1})}{dp^\top} \left( I_n \otimes \frac{\partial h^\top}{\partial x} A^{-1}(t_{k+1}) \right) + P_{xx}(t_{k+1}) \frac{d^2 h^\top}{dp^\top} dx \left( I_n \otimes A^{-1}(t_{k+1}) \right)
\] (4.77) |
| + \[
P_{xx}(t_{k+1}) \frac{\partial h^\top}{\partial x} \frac{dA^{-1}(t_{k+1})}{dp^\top}
\] |

<table>
<thead>
<tr>
<th>Parameter adaptation gain</th>
</tr>
</thead>
</table>
| \[
S(t_{k+1}) = \lambda A(t_{k+1}) + \frac{d\hat{y}(t_{k+1})}{dp^\top} P_{pp}(t_k) \frac{d\hat{y}^\top(t_{k+1})}{dp}
\] (4.78) |

<table>
<thead>
<tr>
<th>Parameter covariance update</th>
</tr>
</thead>
</table>
| \[
P_{pp}(t_{k+1}^+) = \left( P_{pp}(t_{k+1}^-) - L(t_{k+1}) S(t_{k+1}) L^\top(t_{k+1}) \right) / \lambda
\] (4.79) |
Table 4.3 continued

| Sensitivity updates | \[
\frac{d\hat{x}(t_{k+1}^+)}{dp^T} = \frac{d\hat{x}(t_{k+1})}{dp^T} - K(t_{k+1}) \frac{d\hat{y}(t_{k+1})}{dp^T} + \frac{dK(t_{k+1})}{dp^T} (I_s \otimes e(t_{k+1})) \]

| Covariance update | \[
P_{xx}(t_{k+1}^-) = \left( I_n - K(t_{k+1}) \frac{\partial h}{\partial x^T} \right) P_{xx}(t_{k+1}^-) \]

| State update | \[
\Delta \hat{x}(t_{k+1}) = K(t_{k+1}) e(t_{k+1}) \]
\[
\hat{x}(t_{k+1}^+) = \hat{x}(t_{k+1}) + \Delta \hat{x}(t_{k+1}) \]

| Parameter update | \[
\Delta p(t_{k+1}) = L(t_{k+1}) e(t_{k+1}) \]
\[
\hat{p}(t_{k+1}) = \hat{p}(t_{k}) + \Delta \hat{p}(t_{k+1}) \]

### 4.6 Adaptive Higher-Order Filters

In the adaptive EKF algorithm of Section 4.5, the predictor is an EKF. This algorithm is now modified such that higher-order filters are used as predictors. For this purpose, the derivatives of the bias-correction terms in the state estimate propagation equation, the output prediction equation, and the extra term in the expression for the approximate prediction error covariance matrix have to be computed. The propagation equation for the higher-order filters of Chapter 3 is

\[
\hat{x}(t) = f(\hat{x}(t)) + b_p(\hat{x}(t), P_{xx}(t)), \quad (4.88)
\]

where the bias correction-term is given as

\[
b_p = \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x} \text{col } P_{xx}. \quad (4.89)
\]

The sensitivity propagation equation for the state estimate then changes to
\[
\frac{\partial \hat{x}}{\partial p^T} = \frac{\partial f}{\partial x^T} \frac{\partial \hat{x}}{\partial p^T} + \frac{\partial f}{\partial p^T} + \frac{\partial b_p}{\partial \hat{x}^T} .
\] (4.90)

With the third derivative
\[
\frac{d^3 f}{dp^T dx^T dx^T} = \frac{d}{dp^T} \frac{\partial^2 f}{\partial x^T \partial x^T} = \frac{\partial^3 f}{\partial x^T \partial x^T \partial x^T} (\frac{dx}{dp^T} \otimes I_{\alpha^2}) + \frac{\partial^3 f}{\partial p^T \partial x^T \partial x^T} ,
\] (4.91)
the derivative of the bias-correction term can be computed as
\[
\frac{\partial b_p}{\partial \hat{x}^T} = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} (I_{\alpha} \otimes \text{col } P_{\alpha \alpha}) + \frac{\partial^2 f}{\partial x^T \partial x^T} \frac{d \text{col } P_{\alpha \alpha}}{dp^T} .
\] (4.92)

The last term on the right-hand side of Equation (4.92) can be computed from the covariance sensitivity by using elementary stacking operations.

The output prediction equation for the higher-order filters is
\[
\hat{y}(t_{k+1}) = h(\hat{x}(t_{k+1})) + b_m (\hat{x}(t_{k+1}), P_{\alpha \alpha}(t_{k+1})) ,
\] (4.93)
with the bias-correction term given as
\[
b_m = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \text{col } P_{\alpha \alpha} .
\] (4.94)

The gradient of the output prediction can then be written as
\[
\frac{d \hat{y}^T (t_{k+1})}{dp} = \left( \frac{\partial h}{\partial x^T} \frac{\partial \hat{x}(t_{k+1})}{\partial p^T} + \frac{\partial h}{\partial p^T} + \frac{\partial b_m (t_{k+1})}{\partial p^T} \right)^T .
\] (4.95)

Analogously to the derivative of \( b_p \), the derivative of \( b_m \) follows as
\[
\frac{\partial b_m}{\partial \hat{x}^T} = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} (I_{\alpha} \otimes \text{col } P_{\alpha \alpha}) + \frac{\partial^2 h}{\partial x^T \partial x^T} \frac{d \text{col } P_{\alpha \alpha}}{dp^T} ,
\] (4.96)
with the third derivative
\[
\frac{d^3 h}{dp^T dx^T dx^T} = \frac{d}{dp^T} \frac{\partial^2 h}{\partial x^T \partial x^T} = \frac{\partial^3 h}{\partial x^T \partial x^T \partial x^T} (\frac{dx}{dp^T} \otimes I_{\alpha^2}) + \frac{\partial^3 h}{\partial p^T \partial x^T \partial x^T} .
\] (4.97)

For the TSF and the GSF, the approximate prediction error covariance matrix is computed according to
\[
A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} P_{\alpha \alpha}(t_{k+1}) \frac{\partial h^T}{\partial x} + M (\hat{x}(t_{k+1}), P_{\alpha \alpha}(t_{k+1})) .
\] (4.98)

The derivative of the matrix \( A \) is therefore computed as
\[
\frac{dA(t_{k+1})}{d\hat{p}^T_{k+1}} = \left. \frac{dA(t_{k+1})}{d\hat{p}^T_{k+1}} \right|_{\text{EKF}} + \frac{dM(t_{k+1})}{d\hat{p}^T_{k+1}},
\]

where the first term on the right-hand side is the derivative of \( A \) for the EKF in Equation (4.57). For the TSF, the covariance correction term \( M \) is given as

\[
M = -b_n^T b_n
\]

The derivative of this expression can be straightforwardly computed by applying the matrix product rule and follows as

\[
\frac{dM}{d\hat{p}^T} = -\frac{db_m^T}{d\hat{p}^T} (I_x \otimes b_n^T) - b_m^T \frac{db_m^T}{d\hat{p}^T}.
\]

The first derivative on the right-hand side of Equation (4.101) is available from Equation (4.96); the derivative in the second term can be computed in a similar manner and is given as

\[
\frac{db_m^T}{d\hat{p}^T} = \frac{1}{2} \left( \frac{d \left( \text{col} \ P_m^T \right)^T}{d\hat{p}^T} (I_x \otimes \frac{\partial^2 h}{\partial x^T \partial x^T}) + \left( \text{col} \ P_x^T \right)^T \frac{d^3 h^T}{d\hat{p}^T dx dx} \right),
\]

with the third derivative

\[
\frac{d^3 h^T}{d\hat{p}^T dx dx} = \frac{d}{d\hat{p}^T} \frac{\partial^2 h^T}{\partial x^T \partial x} = \frac{\partial^3 h^T}{\partial x^T \partial x^T} \left( \frac{dx}{d\hat{p}^T} \otimes I_m \right) + \frac{\partial^3 h^T}{\partial x^T \partial x}.
\]

For the GSF, the derivation is more involved. The derivative operator \( d/d\hat{p}^T \) has to be applied to the expression

\[
M = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \left( I_n + U_m \right) \left( P_{xx} \otimes P_{xx} \right) \frac{\partial^2 h^T}{\partial x \partial x}.
\]

Application of the matrix product rule results in

\[
\frac{dM}{d\hat{p}^T} = \frac{1}{4} \left\{ \frac{d^3 h}{d\hat{p}^T dx dx^T} \left[ I_x \otimes \left( (I_n + U_m)(P_{xx} \otimes P_{xx}) \frac{\partial^2 h^T}{\partial x \partial x} \right) \right] \right\}
\]

\[
+ \frac{\partial^3 h}{\partial x^T \partial x^T} \left( I_n + U_m \right) \frac{d P_{xx} \otimes P_{xx}}{d\hat{p}^T} \left( I_x \otimes \frac{\partial^2 h^T}{\partial x^T \partial x} \right)
\]

\[
+ \frac{\partial^3 h}{\partial x^T \partial x^T} \left( I_n + U_m \right) \left( P_{xx} \otimes P_{xx} \right) \frac{d^3 h^T}{d\hat{p}^T dx dx} \right\}.
\]

Finally, the derivative of the Kronecker product \( P_{xx} \otimes P_{xx} \) follows from the Kronecker product rule (Rule 3 in Appendix A) as

\[
\frac{d P_{xx} \otimes P_{xx}}{d\hat{p}^T} = \frac{d P_{xx}}{d\hat{p}^T} \otimes P_{xx} + U_m \left( \frac{d P_{xx}}{d\hat{p}^T} \otimes P_{xx} \right) \left( I_x \otimes U_m \right).
\]
With these terms, all expressions required to devise adaptive higher-order filters are provided. Table 4.4 lists the modifications that are needed to change the adaptive EKF algorithm (Table 4.3) to an adaptive higher-order filter.

**Table 4.4** Modifications for higher-order filters

| State estimate propagation | $t^*_k \leq t \leq t^*_{k+1}$ | $\hat{x}(t) = f(\hat{x}) + b_p(\hat{x}, P_{xx})$ (4.107) with $b_p = \frac{1}{2} \frac{\partial^2 f}{\partial x^T \partial x} \text{col} P_{xx}$ (4.108) |
| Sensitivity propagation | $t^*_k \leq t \leq t^*_{k+1}$ | $\frac{\partial \hat{x}}{\partial p^T} = \frac{\partial f}{\partial x^T} \frac{\partial \hat{x}}{\partial p^T} + \frac{\partial f}{\partial p^T} + \frac{\partial b_p}{\partial p^T}$ (4.109) with $\frac{\partial b_p}{\partial p^T} = \frac{1}{2} \left( \frac{d^3 f}{dp^T dx^T dx^T} (I_{xx} \otimes \text{col} P_{xx}) + \frac{\partial^2 f}{\partial x^T \partial x^T} \text{d \col} P_{xx} \right)$ (4.110) and $\frac{d^3 f}{dp^T dx^T dx^T} = \frac{\partial^3 f}{\partial x^T \partial x^T \partial x^T} (\frac{\partial x}{\partial p^T} \otimes I_n) + \frac{\partial^3 f}{\partial p^T \partial x^T \partial x^T}$ (4.111) |
| Predicted output | $\hat{y}(t_{k+1}) = h(\hat{x}(t^*_{k+1})) + b_m(\hat{x}(t^*_{k+1}), P_{xx}(t^*_{k+1}))$ (4.112) with $b_m = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x} \text{col} P_{xx}$ (4.113) |
| Approximate prediction error covariance matrix | $A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} P_{xx}(t^*_{k+1}) \frac{\partial h}{\partial x}^T + M_{\text{TSF/GSF}}(\hat{x}(t^*_{k+1}), P_{xx}(t^*_{k+1}))$ (4.114) with $M_{\text{TSF}} = -b_m b_m^T$ (4.115) and $M_{\text{GSF}} = \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} \left( I_{xx} + U_{xx} \right)(P_{xx} \otimes P_{xx}) \frac{\partial^2 h}{\partial x \partial x}^T$ (4.116) |
Table 4.4 continued

Gradient of prediction  
\[
\frac{d\hat{y}^T(t_{k+1})}{d\hat{\phi}} = \left( \frac{\partial h}{\partial x^T} \frac{d\hat{x}(t_{k+1})}{d\hat{\phi}} + \frac{\partial h}{\partial \hat{\phi}^T} + \frac{\partial b_m(t_{k+1})}{d\hat{\phi}^T} \right)^T 
\]  
(4.117)

with  
\[
\frac{\partial b_m}{\partial \hat{\phi}^T} = \frac{1}{2} \left( \frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} (I_s \otimes \text{col } P_{xx}) + \frac{\partial^2 h}{\partial x^T dx^T} \frac{d\text{col } P_{xx}}{d\hat{\phi}^T} \right) 
\]  
(4.118)

and  
\[
\frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} = \frac{\partial^3 h}{\partial x^T dx^T dx^T} (\frac{dx}{d\hat{\phi}^T} \otimes I_n^T) + \frac{\partial^3 h}{\partial \hat{\phi}^T dx^T dx^T} 
\]  
(4.119)

Derivative of prediction error covariance matrix  
\[
\frac{dA(t_{k+1})}{d\hat{\phi}^T} = \left. \frac{dA(t_{k+1})}{d\hat{\phi}^T} \right|_{\text{EKF}} + \frac{dM_{\text{TSF/GSF}}(t_{k+1})}{d\hat{\phi}^T} 
\]  
(4.120)

with  
\[
\frac{dM_{\text{TSF}}}{d\hat{\phi}^T} = - \frac{db_m}{d\hat{\phi}^T} (I_s \otimes b_m^T) - b_m \frac{db_m}{d\hat{\phi}^T} 
\]  
(4.121)

\[
\frac{db_m}{d\hat{\phi}^T} = \frac{1}{2} \left( \frac{d(\text{col } P_{xx})^T}{d\hat{\phi}^T} (I_s \otimes \frac{\partial^2 h}{\partial x^T dx^T}) + (\text{col } P_{xx})^T \frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} \right) 
\]  
(4.122)

\[
\frac{dM_{\text{GSF}}}{d\hat{\phi}^T} = \frac{1}{4} \left( \frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} [I_s \otimes (I_n^T + U_m)(P_{xx} \otimes P_{xx}) \frac{\partial^2 h}{\partial x^T dx^T}] \right)
\]

\[
+ \frac{\partial^2 h}{\partial x^T dx^T} (I_n^T + U_m) \frac{dP_{xx} \otimes P_{xx}}{d\hat{\phi}^T} (I_s \otimes \frac{\partial^2 h}{\partial x^T dx^T}) + \frac{\partial^2 h}{\partial x^T dx^T} (I_n^T + U_m)(P_{xx} \otimes P_{xx}) \frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} \right) 
\]  
(4.123)

\[
\frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} = \frac{dP_{xx} \otimes P_{xx}}{d\hat{\phi}^T} \otimes P_{xx} + U_m \frac{dP_{xx} \otimes P_{xx}}{d\hat{\phi}^T} (I_s \otimes U_m) \]  
(4.124)

and  
\[
\frac{d^3 h}{d\hat{\phi}^T dx^T dx^T} = \frac{dx}{d\hat{\phi}^T} \otimes I_m + \frac{\partial^3 h}{\partial \hat{\phi}^T dx^T dx^T} \]  
(4.125)
4.7 Discussion of the Cost Functional and Related Approximations

The algorithm discussed in the two previous sections aims at minimizing a quadratic (least-squares) cost functional of the form

\[ J_{LS}(t_k, p) = \frac{1}{2} \sum_{i=1}^{k} e^T(t_i, p) A^{-1}(t_i, p) e(t_i, p), \] (4.126)

where the prediction errors and their covariance matrix are generated by an EKF (or higher-order filter). The only difference between the cost functional in Equation (4.126) and the one in Section 2.4 is that the weighting matrix \( A \) depends on the parameter vector \( p \). The gradient of this cost functional therefore is

\[ \frac{d J_{LS}(t_k, p)}{dp} = \sum_{i=1}^{k} \frac{d e^T(t_i, p) A^{-1}(t_i, p) e(t_i, p)}{dp} \]

\[ -\frac{1}{2} \sum_{i=1}^{k} (I \otimes e^T(t_i, p) A^{-1}(t_i, p)) \frac{dA(t_i, p)}{dp} A^{-1}(t_i, p) e(t_i, p). \] (4.127)

For recursive algorithms, the gradient at time \( t_k \) is usually computed under the assumption that the gradient at time \( t_{k-1} \) is zero (that is, the previous parameter vector was optimal). Moreover, the dependence of \( A \) on \( p \) is commonly neglected in the recursive prediction error method, so that the last term on the right-hand side of Equation (4.127) is omitted. The gradient is thus approximated as

\[ \frac{d J_{LS}(t_k, p)}{dp} \approx \frac{d e^T(t_k, p) A^{-1}(t_k, p) e(t_k, p)}{dp}. \] (4.128)

With the formulation of the adaptive filters given here, however, this approximation is not necessary, and the second term on the right-hand side of Equation (4.127) could easily be included since the expression \( dA/dp \) is already available from the sensitivity model of the filter.

The cost functional in Equation (4.126) can also be seen as an approximation of the negative log-likelihood function

\[ J_{ML}(t_k, p) = \frac{1}{2} \sum_{i=1}^{k} (e^T(t_i, p) A^{-1}(t_i, p) e(t_i, p) + \ln \det A(t_i, p)), \] (4.129)

which is

\[ J_{ML}(t_k, p) = J_{LS}(t_k, p) + \frac{1}{2} \sum_{i=1}^{k} \ln \det A(t_i, p). \] (4.130)
The gradient of this cost functional is (see Section 2.5)

$$\frac{dJ_{ML}(t_k,P)}{dp} = \frac{dJ_{LS}(t_k,P)}{dp} + \frac{1}{2} \sum_{i=1}^{k} (I_i \otimes \text{row}(A^{-1}(t_i,P))) \text{col} \frac{dA(t_i,P)}{dp}.$$  \hspace{1cm} (4.131)

All terms required to compute this gradient are readily available from the adaptive filtering algorithm. The algorithm could therefore without any difficulty be extended to a maximum-likelihood method by simply substituting the recursive maximum-likelihood update equations (Table 2.2) for the least-squares update equations (Table 2.1) – only the “RPE” part of the algorithm needs to be changed. It is not clear, however, whether the maximum-likelihood cost functional offers any advantages over the least-squares cost functional since for nonlinear systems (with process noise) the prediction errors are not Gaussian distributed, anyway.

The above discussion does not hold for adaptive constant-gain filters because a constant weighting matrix $A$ is used. Thus, the adaptive constant-gain filter cannot be interpreted as a maximum-likelihood approach unless the prediction error covariance matrix was known (an unrealistic assumption).

### 4.8 Relationship between State Augmentation and the Sensitivity Model

It should be clear that the reasoning behind the two different approaches (state-augmentation and sensitivity-based methods) is inherently different. In state augmentation, the additional parameters are interpreted as stochastic variables (states), and nonlinear filtering is employed to obtain estimates. In the sensitivity-based approach, the parameters do not need to be considered as stochastic variables. Rather, they are seen as tuning parameters that are adjusted such that good filter performance results. Although these approaches are different, Ljung (1979) and Ljung and Söderström (1983) already pointed out that there are similarities between the two approaches: the EKF can be interpreted as a recursive prediction error algorithm with a simplified gradient computation. Ljung and Söderström (1983) showed this by developing an alternative recursive prediction error algorithm that is asymptotically equivalent to the original one (Appendix 3.A) and comparing this algorithm to the EKF (Appendix 3.C). Ljungquist and Balchen (1993, 1994) also presented a comparison between the state-augmented EKF and the RPE algorithm. However, their comparison is based on “rough, and asymptotically . . . incorrect” approximations and leads to an RPE formulation that is “almost identical” to the state-augmented EKF (Ljungquist and Balchen 1994). Here, the relationship is explored in a different way: the EKF is (exactly, not asymptotically) rewritten in sensitivity-model-based notation. With this notation, the simplifications that lead from the
4.8 RELATIONSHIP BETWEEN STATE AUGMENTATION AND THE SENSITIVITY MODEL

adaptive EKF to the state-augmented EKF become clearly visible. The actual derivation of the reformulation is given in Appendix C. As shown there, the state-augmented EKF can be written in the following form (note that the terms $P_{ss}$ and $K$ here correspond to $\bar{P}_{ss}$ and $\bar{K}$ in Appendix C; for uniformity with the other algorithms presented in this chapter, the former notation is used).

\[
\hat{x} = f(\hat{x}, \hat{u}, \hat{p}(t_k)) \tag{4.132}
\]

\[
\dot{P}_{ss} = \frac{\partial f}{\partial x} P_{ss} + P_{ss} \frac{\partial f^T}{\partial x} + Q \tag{4.133}
\]

\[
\dot{S}_x = \frac{\partial f}{\partial x} S_x + \frac{\partial f}{\partial p} \tag{4.134}
\]

\[
\dot{y}(t_{k+1}) = h(\hat{x}(t_{k+1}), u(t_{k+1}), \hat{p}(t_k)) \tag{4.135}
\]

\[
e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1}) \tag{4.136}
\]

\[
S_y(t_{k+1}) = \frac{\partial h}{\partial x} S_x(t_{k+1}) + \frac{\partial h}{\partial p} \tag{4.137}
\]

\[
A(t_{k+1}) = R + \frac{\partial h}{\partial x} P_{ss}(t_{k+1}) \frac{\partial h^T}{\partial x} + S_y(t_{k+1}) P_{pp}(t_k) S_y^T(t_{k+1}) \tag{4.138}
\]

\[
K(t_{k+1}) = P_{ss}(t_{k+1}) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1}) \tag{4.139}
\]

\[
L(t_{k+1}) = P_{pp}(t_k) (\frac{\partial h}{\partial x} S_x(t_{k+1}) + \frac{\partial h}{\partial p})^T A^{-1}(t_{k+1}) \tag{4.140}
\]

\[
\Delta \hat{p}(t_{k+1}) = L(t_{k+1}) e(t_{k+1}) \tag{4.141}
\]

\[
\hat{p}(t_{k+1}) = \hat{p}(t_k) + \Delta \hat{p}(t_{k+1}) \tag{4.142}
\]

\[
\Delta \hat{v}(t_{k+1}) = K(t_{k+1}) e(t_{k+1}) + S_x(t_{k+1}) \Delta \hat{p}(t_{k+1}) \tag{4.143}
\]

\[
\hat{v}(t_{k+1}^+) = \hat{v}(t_{k+1}) + \Delta \hat{v}(t_{k+1}) \tag{4.144}
\]

\[
P_{pp}(t_{k+1}) = P_{pp}(t_k) - L(t_{k+1}) A(t_{k+1}) L^T(t_{k+1}) \tag{4.145}
\]

\[
P_{ss}(t_{k+1}^+) = P_{ss}(t_{k+1}^-) - K(t_{k+1}) A(t_{k+1}) K^T(t_{k+1}) - K(t_{k+1}) A(t_{k+1}) L^T(t_{k+1}) P_{pp}^{-1}(t_{k+1}) L(t_{k+1}) A(t_{k+1}) K^T(t_{k+1}) \tag{4.146}
\]
The above formulation of the state-augmented EKF gives rise to some remarkable observations. First, it follows from Equations (4.134) and (4.147) that the term $S_x$ closely resembles the sensitivity $\frac{d\hat{x}}{d\hat{p}^T}$: Their propagation equations are identical, and the difference in their update equations has an interesting interpretation. If the parameter covariance matrix $P_{pp}$ were constant, the term $P_{pp}(t_k)P_{pp}^{-1}(t_{k+1})$ could be dropped, and Equation (4.147) would become

$$S_x(t_{k+1}^+)=S_x(t_{k+1}^-)-K(t_{k+1})S_y(t_{k+1}).$$  \hspace{1cm} (4.148)

With the assumption $S_x = \frac{d\hat{x}}{d\hat{p}^T}$, the right-hand side of Equation (4.137) becomes identical to the transposed gradient $d\hat{y}/d\hat{p}^T$ in Equation (4.72), and it follows that $S_y = d\hat{y}/d\hat{p}^T$. Equation (4.148) then becomes

$$S_x(t_{k+1}^+)=S_x(t_{k+1}^-)-K(t_{k+1})\frac{dy(t_{k+1})}{d\hat{p}^T}.$$  \hspace{1cm} (4.149)

This should be compared to the update equation for the sensitivity, Equation (4.25). The difference is that the state-augmented EKF neglects the dependence between the filter gain and the system parameters; thus, the state-augmented EKF uses the approximate sensitivity update

$$\frac{d\hat{x}(t_{k+1}^+)}{d\hat{p}^T} = \frac{d\hat{x}(t_{k+1}^-)}{d\hat{p}^T} - K(t_{k+1})\frac{dy(t_{k+1})}{d\hat{p}^T}.$$  \hspace{1cm} (4.150)

This approximation eliminates the term $dK/d\hat{p}^T$, which is the major source of complexity in the adaptive EKF. Because of this simplification, the sensitivity of the covariance matrix is no longer needed, but it also becomes impossible to estimate parameters in the noise covariance matrices. This difference allows the state-augmented EKF to be interpreted as an approximate recursive prediction error algorithm with a simplified gradient (or, more precisely, sensitivity) computation.

A second difference lies in the computation of the filter gains. For the state-augmented EKF, both filter gains are computed using the same prediction error covariance matrix $A$. This makes sense because both the state and the parameter vector are treated as random variables and estimated by the same filter. This filter only “sees” one prediction error covariance matrix, whereas in the adaptive EKF, the predictor only “sees” the prediction error covariance matrix.
matrix that is influenced by the variability of the state vector. Only the adaptation algorithm also “sees” the influence of the parameter variability on the prediction error covariance.

A third difference is the way the state update is computed in Equation (4.143), where the additional term \( S_x(t_{k+1})\Delta \hat{p}(t_{k+1}) \) is present. Using the above argumentation, this term corresponds to \( \frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} \Delta \hat{p}(t_{k+1}) \), and the expression for the state update can then be written as

\[
\hat{x}(t_{k+1}) = \hat{x}(t_{k+1}) + K(t_{k+1}) e(t_{k+1}) + \frac{d\hat{x}(t_{k+1})}{d\hat{p}^T} \Delta \hat{p}(t_{k+1}).
\] (4.151)

The last term on the right-hand side of this update equation corresponds to a first-order expansion of \( \hat{x}(t_{k+1}) \) which would correct for the effect of a small parameter change \( \Delta \hat{p}(t_{k+1}) \).

Finally, a fourth difference is given by the last term on the right-hand side of Equation (4.146). This term follows from the derivation of the alternative notation and does not seem to have an obvious interpretation.

The relationship between the terms in the RPE formulation and the terms in the conventional formulation is given in Table 4.5. The RPE formulation of Ljungström and Balchen (1994, Table 1) is also included. They used the approximation that “the state covariance matrix [of the EKF] corresponds to the state sensitivity matrix times the parameter covariance matrix times the transposed state sensitivity matrix” and concluded that this “is asymptotically . . . incorrect since the parameter covariance matrix tends to zero while the state covariance matrix does not.” This approximation would, in fact, refer to the case where all uncertainty in the states is caused by the parameter variability. The exact RPE formulation in Table 4.5 avoids this approximation: here the state covariance matrix of the EKF corresponds to the state covariance matrix of the RPE algorithm plus the product term of the state sensitivity and the parameter covariance.

For reference, the formulation of the state-augmented EKF as an adaptive filter is given in Table 4.6. The additional term in the covariance update equation (4.146) and the term \( P_{pp}(t_k)P_{pp}^{-1}(t_{k+1}) \) in the sensitivity update equation (4.147) are omitted.

The differences between the state-augmented EKF and the adaptive EKF could be used to design some modifications for the adaptive EKF. For example, an additional first-order correction step for the state estimate as in Equation (4.151) could be implemented, or the same prediction error covariance could be applied in the gain computations. For the examples studied in this dissertation, however, none of these modifications significantly changed the behavior of the algorithm.
Table 4.5 EKF – RPE comparison

<table>
<thead>
<tr>
<th></th>
<th>Conventional EKF</th>
<th>Exact RPE</th>
<th>Approximate RPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{xx}$</td>
<td>$P_{xx}$</td>
<td>$P_{xx} + \frac{d\hat{x}}{d\hat{p}} P_{pp} \frac{d\hat{x}^T}{d\hat{p}}$</td>
<td>$\frac{d\hat{x}}{d\hat{p}} P_{pp} \frac{d\hat{x}^T}{d\hat{p}}$</td>
</tr>
<tr>
<td>$P_{xp}$</td>
<td>$\frac{d\hat{x}}{d\hat{p}} P_{pp}$</td>
<td>$\frac{d\hat{x}}{d\hat{p}} P_{pp}$</td>
<td></td>
</tr>
<tr>
<td>$P_{pp}$</td>
<td>$P_{pp}$</td>
<td>$P_{pp}$</td>
<td></td>
</tr>
<tr>
<td>$K$</td>
<td>$K + \frac{d\hat{x}}{d\hat{p}} L$</td>
<td>$K$ (for $P_{xx} = 0$)</td>
<td></td>
</tr>
<tr>
<td>$L$</td>
<td>$L$</td>
<td>$L$</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6 The state-augmented Extended Kalman Filter as an adaptive filter

<table>
<thead>
<tr>
<th></th>
<th>State estimate propagation $t_1^+ \leq t \leq t_{k+1}$</th>
<th>( \hat{x} = f(\hat{x}, u, \hat{p}(t_k)) ) (4.152)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Covariance propagation $t_1^+ \leq t \leq t_{k+1}$</td>
<td>( \hat{P}<em>{xx} = \frac{\partial f}{\partial \hat{x}} P</em>{xx} + P_{xx} \frac{\partial f^T}{\partial \hat{x}} + Q(\hat{p}_k) ) (4.153)</td>
</tr>
<tr>
<td></td>
<td>Sensitivity propagation $t_1^+ \leq t \leq t_{k+1}$</td>
<td>( \frac{d\hat{x}}{d\hat{p}} = \frac{\partial f}{\partial \hat{x}} \frac{d\hat{x}}{d\hat{p}} + \frac{\partial f}{\partial \hat{p}} ) (4.154)</td>
</tr>
<tr>
<td></td>
<td>Predicted output $t_{k+1}$</td>
<td>( \hat{y}(t_{k+1}) = h(\hat{x}(t_{k+1}), u(t_{k+1}), \hat{p}(t_k)) ) (4.155)</td>
</tr>
<tr>
<td></td>
<td>Prediction error $t_{k+1}$</td>
<td>( e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1}) ) (4.156)</td>
</tr>
<tr>
<td></td>
<td>Gradient of prediction $t_{k+1}$</td>
<td>( \frac{d\hat{y}^T(t_{k+1})}{d\hat{p}} = \left( \frac{\partial h}{\partial \hat{x}} \frac{d\hat{x}(t_{k+1})}{d\hat{p}} + \frac{\partial h}{\partial \hat{p}} \right)^T ) (4.157)</td>
</tr>
</tbody>
</table>
### 4.9 Relationship to Other Approaches

The methods discussed in this chapter fit into the framework of so-called system reference adaptive model (SRAM) techniques (Unbehauen and Rao 1987, Chapter 6). In this framework, an adaptive model is connected to the true system, which provides the reference behavior for this adaptive model. Unbehauen and Rao (1987) further divide this technique into the parallel model approach, the series model approach, and the series-parallel model approach. This classification can be made according to which input signals are fed into the adaptive model and which output signals the adaptive model generates:

- **In the parallel model approach** (Unbehauen and Rao 1987, Section 6.1.1), the adaptive model gets the same input signal as the true system and generates a signal that should

---

**Table 4.6 continued**

<table>
<thead>
<tr>
<th><strong>EKF</strong></th>
<th>Approximate prediction error covariance matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A(t_{k+1}) = R + \frac{\partial h}{\partial x} P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x}$ $+$ $\frac{\partial \hat{y}(t_{k+1})}{\partial \hat{p}^T} P_{pp}(t_{k}) \frac{\partial \hat{y}^T(t_{k+1})}{\partial \hat{p}}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Filter gain</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$K(t_{k+1}) = P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>RPE</strong></th>
<th>Parameter adaptation gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L(t_{k+1}) = P_{pp}(t_{k}) \frac{\partial \hat{y}^T(t_{k+1})}{\partial \hat{p}} A^{-1}(t_{k+1})$</td>
<td></td>
</tr>
</tbody>
</table>

| **Parameter update** |
|---|---|
| $\Delta \hat{p}(t_{k+1}) = L(t_{k+1}) e(t_{k+1})$ |
| $\hat{p}(t_{k+1}) = \hat{p}(t_{k}) + \Delta \hat{p}(t_{k+1})$ |

| **Parameter covariance update** |
|---|---|
| $P_{pp}(t_{k+1}) = P_{pp}(t_{k}) - L(t_{k+1}) A(t_{k+1}) L^T(t_{k+1})$ |

<table>
<thead>
<tr>
<th><strong>EKF</strong></th>
<th>State update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta \hat{x}(t_{k+1}) = K(t_{k+1}) e(t_{k+1}) + \frac{\partial \hat{x}(t_{k+1})}{\partial \hat{p}^T} \Delta \hat{p}(t_{k+1})$</td>
<td></td>
</tr>
<tr>
<td>$\hat{x}(t_{k+1}) = \hat{x}(t_{k+1}) + \Delta \hat{x}(t_{k+1})$</td>
<td></td>
</tr>
</tbody>
</table>

| **Covariance update** |
|---|---|
| $P_{xx}(t_{k+1}) = P_{xx}(t_{k+1}) - K(t_{k+1}) A(t_{k+1}) K^T(t_{k+1})$ |

<table>
<thead>
<tr>
<th><strong>SM</strong></th>
<th>Sensitivity update</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{\partial \hat{x}(t_{k+1})}{\partial \hat{p}^T} = \frac{\partial \hat{x}(t_{k+1})}{\partial \hat{p}^T} - K(t_{k+1}) \frac{\partial y(t_{k+1})}{\partial \hat{p}^T}$.</td>
<td></td>
</tr>
</tbody>
</table>

---
reproduce the true system’s output. This approach is also referred to as output error approach (Landau 1979, Chapter 7). The corresponding parameter adaptation mechanism aims at minimizing the deviation of the model’s output from the system’s output, the so-called output error. The model’s output is only computed from the system’s input. This indicates that the model cannot be an optimal predictor since information contained in past outputs is not directly used.

- In the series model approach (Unbehauen and Rao 1987, Section 6.1.2), the adaptive model receives the output of the true system as its input signal. The adaptive model corresponds to the inverse of the true system augmented with an additional dynamical system to ensure realizability. The output of the adaptive model should reproduce the signal that results if the system’s input is filtered through this additional dynamical system. This approach is also called the input error approach (Landau 1979, Chapter 7), as the adaptation mechanism minimizes the difference between the filtered system input and the model output.

- In the series-parallel model approach (Unbehauen and Rao 1987, Sections 6.1.3 and 6.3.1.2), the adaptive model gets both the true system’s input and output signals as input signals. The adaptive model either generates a prediction, from which a prediction error can be computed, or it directly generates the (possibly filtered) equation error. Generally, the terms “prediction error” and “equation error” refer to the same quantity. For the above reasons, the series-parallel model approach is also called equation error approach (Landau 1979, Chapter 7). In the filtering literature, the term “residual” is ambiguously used to denote both the (a priori) prediction error and the a posteriori error computed with the updated estimates immediately after a measurement has been processed (Maybeck 1979, p. 218). In the system identification literature, the latter meaning is usually adopted (Ljung and Söderström 1983, p. 316), and the prediction error is called “innovation.” In a truly continuous-time setting, these differences disappear: the model output at time $t$ is computed using all the information that is available up to time $t$. It thus no longer makes sense to think of the model output as a prediction. The term “equation error” is then more appropriate, unless the somewhat contradictory term “posterior prediction errors” (Ljung and Söderström 1983, p. 316) is adopted. Contrary to Unbehauen and Rao (1987) and Landau (1979), Young (1981) classifies the approaches differently. He uses the term “prediction error method” if a noise model is included. Then, a “prediction error approach within an output error or an equation error context” can be defined. This is, however, not helpful here.
This classification reveals that the adaptive filtering (or prediction error) approach proposed here corresponds to a series-parallel model (or equation error) approach, as the filter relies both on the system’s input and output signals to generate a prediction. Young (1981) defines the state-augmented EKF as an output error method, but this ignores the fact that the output error method in the original form (Young 1981, Figure 3a) does not include a state correction, whereas the EKF (Young 1981, Figure 7) does.

Some earlier approaches to the identification of nonlinear state-space models were based on a parallel model approach (Matausek and Stankovic 1980; Sidar 1976). In these approaches, the state estimates were generated through the simulation of the adjustable model, and no explicit state correction was carried out, that is, the mismatch between the true and the estimated output was not used to improve the state estimates. The adjustable model only based the “estimate” of the system’s output signal on the system’s input signal; thus, no filtering operation took place. It should be clear that when unknown initial conditions, process noise, or other disturbances are present, the filtering approach is superior to the parallel model approach: the filtering operation can correct the deviation between the true and the estimated states, whereas in the parallel model approach only the parameter adjustment can correct these differences. Figure 4.1 illustrates the difference between a parallel model approach, where (ideally) the estimated state converges to the true states without any measurement corrections, and a series-parallel model approach, where the state is corrected at every sampling instant.

The series model approach does not seem appropriate for estimating the parameters of systems with a prescribed structure since an analytical inverse would be necessary. This approach is, however, common practice in black-box system identification, where, for example, neural networks are used to learn the inverse of a system (Ng 1997).

Figure 4.1. Convergence of the estimated state to the true state for a parallel model approach (left) and a series-parallel model approach (right) in a continuous-discrete setting.
A somewhat similar approach to the adaptive EKF used here was applied by Chu et al. (1996): Instead of using an EKF as a predictor, the current parameter estimates are inserted into a state-augmented EKF that is then used to generate state estimates. The parameter update of this EKF is modified so that the system parameters and the noise parameters are still estimated by a gradient approach. Based on an example for which the first approach fails, Chu et al. (1996) claim that the latter approach is superior because the “modified approach . . . correctly accounts for the interactions between the state estimator and the parameter estimator, improving the performance of the adaptive filter.” Since it is shown in Section 4.8 that this interaction between the state and parameter estimation in the state-augmented EKF can be interpreted as a first-order correction of the state estimate in Equation (4.151) and the replacement of a weighting matrix, this statement becomes doubtful. If it were true, these modifications could easily be incorporated into the algorithm presented here. Further differences exist in the work Chu et al. (1996). First, instead of applying a sensitivity model, the sensitivities are computed from finite differences. Secondly, instead of implementing a quadratic error criterion, the maximum-likelihood cost functional is used.

4.10 Chapter Summary and Discussion

In this chapter, various algorithms for adaptive filtering and parameter estimation have been discussed. The following items have been covered:

- The concept of state augmentation has been reviewed in Section 4.2. In this section, the algorithm for a state-augmented EKF has also been given.
- The foundations of the sensitivity-based approaches have been discussed in Section 4.3. It has been shown that the derivative of the filter gain causes major complications.
- An approach that avoids these difficulties by assuming a constant filter gain has been reviewed in Section 4.4.
- A new approach that consists of an adaptive EKF with a full sensitivity model has been suggested in Section 4.5. Terms required to use higher-order filters as adaptive predictors have been derived in Section 4.6.
- The cost functional that is used to derive the algorithm has been discussed in Section 4.7, and the relationship to the maximum-likelihood method has been drawn.
- In Section 4.8, the similarities between state-augmentation and sensitivity-based approaches have been outlined. The differences between both approaches have been investigated by exactly reformulating the state-augmented EKFs as a recursive prediction error algorithm. In this formulation, the differences can be clearly interpreted.
Finally, the relationship to other approaches has been discussed in Section 4.9. In this section, it has been shown that the algorithm falls into the class of series-parallel approaches in the framework of Unbehauen and Rao (1987).

This chapter contains the main contributions of this dissertation, which can be summarized as follows:

- An adaptive EKF and adaptive higher-order filters have been developed. Over existing approaches, these algorithms offer two advantages. First, the noise covariance matrices can be parameterized as well, which is not the case for state-augmented filters. These proposed filters thus overcome the bias problem associated with incorrect noise assumptions. Secondly, the algorithms keep the filter gain time-varying, which should intuitively give better state estimates and make the algorithms applicable to a wider class of systems. This constitutes another advantage over the previously suggested constant-gain filters: constant-gain filters can only be applied to systems for which a stable constant-gain filter exists (see Section 6.4 below).

- The whole filtering algorithm and the sensitivity model have been derived by consequently applying matrix differential calculus. This results in an elegant, easily interpretable, and implementation-ready formulation. The difficulty Aström and Källström (1973) pointed out with reference to the computation of gradients for the maximum-likelihood identification of linearized ship dynamics that “it is, however, extremely tedious to compute the gradient of the likelihood function . . . .” has been alleviated.

- The algorithm uses a least-squares cost functional but is very general and can be readily modified for a maximum-likelihood cost functional by the inclusion of two additional terms in the gradient. Expressions for these terms are given in Section 4.7. The algorithm can thus be interpreted as a version of the maximum-likelihood method. In comparison to previous applications of the maximum-likelihood method (for example, Mehra and Tyler 1973; Chu et al. 1996), the method here is developed as a recursive algorithm, and a complete and compact formulation of the sensitivity model is used.

- A new approach to compare the state-augmented EKF to the adaptive EKF has been pursued. An exact reformulation has been employed instead of the previously used asymptotic and approximate comparisons. The differences between the algorithms include an approximate gradient computation, a first-order correction step in the state estimate, and a replacement of a covariance matrix that results from the interpretation of the parameters as stochastic variables.
Chapter 5

Implementation of the Propagation Equations

In this chapter, the propagation equations of the adaptive Extended Kalman Filter discussed in the previous chapter are changed to a form that is more suitable for implementation. The aim is a formulation in which all redundant entries are removed and all differential equations are vectorized. This becomes possible by applying stacking operations and by interchanging stacking operations with differentiation.

The four propagation differential equations from the last chapter provide the starting point for this derivation:

\[
\dot{x} = f(\hat{x}, u, \hat{p}) \tag{4.62}
\]

\[
\dot{P}_{xx} = \frac{\partial f}{\partial x^T} P_{xx} + P_{xx} \frac{\partial f^T}{\partial x} + Q \tag{4.63}
\]

\[
\frac{d\hat{x}}{dp^T} = \frac{\partial f}{\partial x^T} \frac{d\hat{x}}{dp^T} + \frac{\partial f}{\partial p^T} \tag{4.64}
\]

\[
\frac{dP_{xx}}{dp^T} = \frac{d^2 f}{dp^T dx^T} (I_x \otimes P_{xx}) + \frac{\partial f}{\partial x^T} \frac{dP_{xx}}{dp^T} + \frac{dP_{xx}}{dp^T} (I_x \otimes \frac{\partial f^T}{\partial x}) + P_{xx} \frac{d^2 f^T}{dp^T dx} + \frac{\partial Q}{\partial p^T}. \tag{4.65}
\]

Out of these four equations, only Equation (4.62) is ready for implementation “as is.” The other equations should not be used in the form given here. Instead, these equations should be vectorized, such that they can all be grouped into one large vector differential equation; and redundant entries should be removed, both to reduce storage and computation time requirements as well as to avoid numerical problems (loss of symmetry).

First, Equation (4.63) is vectorized using the well-known relation \( \text{col } ABC = (C^T \otimes A) \text{ col } B \). This gives

\[
\text{col } \dot{P}_{xx} = (\frac{\partial f}{\partial x^T} \otimes \frac{\partial f}{\partial x^T}) \text{ col } P_{xx} + \text{ col } Q. \tag{5.1}
\]

Only \( n(n+1)/2 \) of the \( n^2 \) individual elements of this vector differential equation are distinct since the covariance matrix is symmetric. To remove the redundant entries, the reduced column operator defined in Equation (2.8) is used, and Equation (5.1) is rewritten as
\[ c \dot{P}_{xx} = \left( T_r \left( \frac{\partial f}{\partial x} \right)^T \oplus \frac{\partial f}{\partial x} \right) T_r \right) \cdot c \, P_{xx} + c \, Q \cdot (5.2) \]

This is a vector differential equation without redundant entries; thus, it can be implemented in this form.

Next, Equation (4.64) is vectorized using the column operator to give

\[ \text{col} \left( \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} \right) = \left( I \otimes \frac{\partial f}{\partial x} \right) \cdot \text{col} \left( \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} \right) + \text{col} \left( \frac{\partial f}{\partial \hat{p}^T} \right) \cdot (5.3) \]

With the relation

\[ \text{col} \left( \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} \right) = \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} \cdot (5.4) \]

Equation (5.3) can be rewritten as

\[ \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} = \left( I \otimes \frac{\partial f}{\partial x} \right) \cdot \frac{\text{d} \hat{x}}{\text{d} \hat{p}^T} + \frac{\partial f}{\partial \hat{p}^T} \cdot (5.5) \]

The sensitivity does not contain any redundant entries; therefore, Equation (5.5) cannot be further simplified.

Finally, the propagation equation for the covariance sensitivity, Equation (4.65), is simplified. It is possible to vectorize Equation (4.65) directly. However, a different approach using Equation (5.2) as a starting point is pursued here. First, it is noted that the covariance sensitivity is given as

\[ \frac{\text{d} P_{xx}}{\text{d} \hat{p}^T} = \begin{bmatrix} \frac{\text{d} P_{xx}}{\text{d} \hat{p}_1} & \ldots & \frac{\text{d} P_{xx}}{\text{d} \hat{p}_j} \end{bmatrix}, \cdot (5.6) \]

which is a matrix composed of \( s \) symmetric \( n \) by \( n \) blocks. Therefore, only \( n(n+1)/2 \) of its \( n^2s \) entries are distinct. This matrix can be stacked into a column vector and expressed in terms of the reduced column vector \( c \, P_{xx} \) as

\[ \text{col} \left( \frac{\text{d} P_{xx}}{\text{d} \hat{p}^T} \right) = \begin{bmatrix} \frac{\text{d} T_r \cdot c \, P_{xx}}{\text{d} \hat{p}_1} \\ \vdots \\ \frac{\text{d} T_r \cdot c \, P_{xx}}{\text{d} \hat{p}_j} \end{bmatrix} = \left( I \otimes T_r \right) \cdot \frac{\text{d} c \, P_{xx}}{\text{d} \hat{p}^T} \cdot (5.7) \]

The last term on the right-hand side of this equation is the derivative of the reduced column vector of the covariance matrix with respect to the parameter vector. This term contains all
distinct entries of the covariance sensitivity and can be constructed from the covariance sensitivity as

$$\frac{d \ c \ P_{xx}}{d \ p} = (I_x \otimes T_e) \ \text{col} \ \frac{dP_{xx}}{d \ p^T}.$$  \hspace{1cm} (5.8)

Furthermore, the covariance sensitivity can be reconstructed as

$$\frac{dP_{xx}}{d \ p} = \text{mat} \ _{\text{nons}} \ (I_x \otimes T_e) \ \frac{d \ c \ P_{xx}}{d \ p}.$$  \hspace{1cm} (5.9)

The matrix propagation differential equation for $dP_{xx}/dp^T$ can thus be replaced by the vector differential propagation equation for $d \ c \ P_{xx}/d \ p$. This equation can be obtained by differentiating Equation (5.2). Then, instead of the derivatives defined in Equations (4.51) and (4.52), only

$$\frac{d^2 f}{d \ p dx^T} = \frac{d}{d \ p} \frac{df}{dx^T} = (\frac{d \hat{X}^T}{d \ p} \otimes I_n) \ \frac{\partial^2 f}{\partial x^T \partial x^T} + \frac{\partial^2 f}{\partial \ p \partial x^T},$$  \hspace{1cm} (5.10)

is required. Using Equation (2.16), the derivative of the Kronecker sum in Equation (5.2) is obtained as

$$\frac{d}{d \ p} \left( \frac{df}{dx^T} \oplus \frac{df}{dx^T} \right) = (\frac{d^2 f}{d \ p dx^T} \otimes I_n) + (I_s \otimes \Theta_{nn}) \ (\frac{d^2 f}{d \ p dx^T} \otimes I_n) \ U_{nn},$$  \hspace{1cm} (5.11)

and the differential equation for $d \ c \ P_{xx}/d \ p$ follows as

$$\frac{d \ c \ P_{xx}}{d \ p} = (I_x \otimes T_e) \ (\frac{d \ c \ P_{xx}}{d \ p}) \left( I_x \otimes \left( I_e \frac{df}{dx^T \oplus \frac{df}{dx^T} \otimes T_e} \right) \right) \ \frac{d \ c \ P_{xx}}{d \ p}$$

$$+ (I_s \otimes T_e) \ \frac{d \ c \ Q}{d \ p}.$$  \hspace{1cm} (5.12)

Thus, Equations (4.62), (5.2), (5.5), and (5.12) constitute the full set of propagation differential equations for the adaptive Extended Kalman Filter. The propagation phase of the algorithm can then be carried out in the following steps:

1. With the quantities at $t_k^+$, build $d \hat{X}/d \ p$ by stacking $d \hat{x}/d \ p^T$ into a column vector; build the reduced column vector $c \ P_{xx}$; construct the reduced column vector $d \ c \ P_{xx}/d \ p$ using Equation (5.8).
2. Integrate the propagation equations (4.62), (5.2), (5.5), and (5.12) from $t_k^+$ to $t_{k+1}$.
3. Stack $\frac{d\mathbf{x}}{d\mathbf{p}}$ into an $n$ by $s$ matrix to obtain $\frac{d\mathbf{x}}{d\mathbf{p}}^T$; build the full column vector $\text{col } P_{s_s} = T_c P_{s_s}$, and stack it into an $n$ by $n$ matrix to get $P_{s_s}$; rebuild $\frac{dP_{s_s}}{d\mathbf{p}}^T$ from $\text{d} \text{c } P_{s_s} / d\mathbf{p}$ using Equation (5.9).

In this manner, all redundant entries are eliminated from the differential equations. The formulation of the propagation equations presented here permits an easy and straightforward implementation, yet it may not be computationally efficient since Kronecker products and large sparse matrices are involved. Nevertheless, this implementation has been used to build the filters for the simulation examples in Chapter 6 and the filter for the real-time application in Chapter 7. Furthermore, the derivation of this simplified formulation shows that matrix differential calculus, Kronecker products, and the column stacking operation offer powerful tools for such situations. By applying these tools, algebraic effort and notational complexity can be reduced, element-by-element computations (that would involve rather complex summations over the products of individual matrix entries and derivatives) are avoided, and bookkeeping over individual indices becomes unnecessary.
Chapter 6

Simulation Studies

In this chapter, the adaptive EKF developed in Chapter 4 is tested in simulation studies and, as far as possible, compared to the nonadaptive/state-augmented EKF and the adaptive CGEKF. The purpose of these simulation studies is to point out some potential applications for the nonlinear system identification (and filtering) algorithms discussed in this dissertation. The first example (Section 6.1) highlights the improvement in filter performance due to adaptation and illustrates that a time-varying filter gain gives better performance than a constant filter gain. The second example (Section 6.2) demonstrates the bias problem resulting from wrongly tuned noise covariance matrices in the EKF and shows that adaptive filters overcome this problem. In the third example (Section 6.3), a seemingly “harmless” system is considered, where the “harmlessness” is exhibited by all algorithms working satisfactorily. This example is included to stress that in nonlinear system identification it is difficult to compare algorithms since such comparisons are always application-specific. A system where the constant-gain filter fails is considered in the fourth example (Section 6.4); here, a time-varying gain is required for a stable filter, yet only the state-augmented EKF and the adaptive EKF allow for such a time-varying gain. The final three examples (Sections 6.5, 6.6, and 6.7) demonstrate some further applications of the adaptive EKF: a linear dynamical system with a static output nonlinearity (a Wiener system) is identified in Section 6.5, a bilinear dynamical system (motivated by the model of a dc motor) with an additional input nonlinearity (a saturation element) is considered in Section 6.6, and a mass-spring-damper system with a time-varying mass and an input nonlinearity (a deadzone) is discussed in Section 6.7. Some general conclusions are drawn in Section 6.8.

6.1 Adaptive Filtering for a Van der Pol Oscillator

This example illustrates the performance improvement due to adaptive filtering. The system under consideration is the Van der Pol oscillator that was used by Ahmed (1994) and is described by
\[
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t)
\end{bmatrix} = \begin{bmatrix}
x_2(t) \\
-9x_1(t) + 2(1-x_1^2(t))x_2(t)
\end{bmatrix} + v(t),
\]

\[y(t_k) = \sqrt{0.05} x_1^2(t_k) + w(t_k),\]

with \( t_k = kT \) and a sampling time of \( T = 0.05 \) seconds. The only difference to the system considered by Ahmed (1994) is that instead of 0.05 the factor \( \sqrt{0.05} \) is used in the measurement equation. The factor \( \sqrt{0.05} \) was chosen to approximately scale the output to unit variance (for the noise-free case). The initial state was set to

\[x(0) = \begin{bmatrix}
-2 \\
-1
\end{bmatrix}.
\]

The noise (co)variances were chosen as

\[E v(t)v^T(t') = 0.5 I \delta(t-t')\]

for the process noise and

\[E w(t_i)w(t_j) = 0.2 \delta_{ij}\]

for the measurement noise. To demonstrate the effect of the process noise, parts of the noise-free and the noisy state trajectories are shown as phase-plane plots in Figure 6.1. The output signal is shown in Figure 6.2.

The duration of the simulation was 500 seconds. The settings for the EKF were selected as

\[\hat{x}(0) = [0 \ 0]^T,\ P_x(0) = I,\ R = 0.2,\]

and different values for \( Q \) were considered. Table 6.1 lists the empirical estimation error variances that are obtained when \( Q \) is varied. It is evident that the performance is quite sensitive with regard to \( Q \) and that both too small and too large a covariance matrix lead to reduced filter performance.

\[\begin{array}{c|c|c}
Q & \hat{\sigma}^2(e_1) & \hat{\sigma}^2(e_2) \\
\hline
0.1 I & 0.1302 & 1.1543 \\
0.5 I & 0.1093 & 0.9487 \\
I & 0.1140 & 0.9829 \\
5 I & 0.1572 & 1.4502
\end{array}\]
6.1 Adaptive Filtering for a Van der Pol Oscillator

Figure 6.1 State trajectories of the Van der Pol oscillator, noise-free (left) and noisy (right)

For the filtering experiment with the adaptive EKF, the settings

\[
\hat{x}(0) = [0 \ 0]^T, \ P_{xx}(0) = I, \ Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \hat{p}_1^2 & 0 \\ 0 & \frac{1}{2} \hat{p}_2^2 \end{bmatrix}, \ \hat{p}(0) = \begin{bmatrix} \sqrt{0.2} \\ \sqrt{0.2} \end{bmatrix}, \ \frac{d\hat{x}(0)}{d\hat{p}^T} = 0,
\]

\[
\frac{dP_{xx}(0)}{d\hat{p}^T} = 0, \ P_{pp}(0) = 10I, \ R = 0.2
\]

were chosen. Figure 6.3 shows the estimated diagonal elements of the process noise covariance matrix and the elements of the parameter covariance matrix. The elements of the parameter covariance matrix (particularly, the lower right element \(P_{pp,22}\) corresponding to \(\hat{Q}_{22}\)) decrease slowly. This indicates that the prediction errors contain little information about the optimal size of the process noise covariance matrix. Intuitively, this can be explained by the fact that the process noise covariance matrix affects the prediction errors rather indirectly: it “adjusts” the size of the state estimation error covariance matrix, which then determines the filter gain. Similar results have been observed in many identification and filtering experiments: the elements of the process noise covariance matrix tend to converge slowly.
The final value obtained for the process noise covariance matrix is

\[
Q = \begin{bmatrix}
0.4941 & 0 \\
0 & 0.8716
\end{bmatrix}.
\]

Interestingly, this value differs from the true covariance matrix \((0.5 \mathbf{I})\). The second diagonal element is slightly overestimated, which corresponds to the introduction of pseudonoise. It is seen below that the performance with this “optimal” covariance matrix is indeed better than the performance obtained with the true covariance matrix.

For the adaptive CGEKF, the settings

\[
\hat{x}(0) = [0 \quad 0]^T, \quad K(\hat{p}) = [\hat{p}_1 \quad \hat{p}_2]^T, \quad \hat{p}(0) = [0 \quad 0]^T, \quad \frac{d\hat{x}(0)}{d\hat{p}} = \mathbf{0}, \quad P_{\hat{p}}(0) = \mathbf{I}, \quad A = 1
\]

were selected. Figure 6.4 shows the estimated filter gains and the elements of the parameter covariance matrix. Here, the elements of the covariance matrix decrease faster, which indicates that there is more information about the size of the filter gain in the prediction error. This can be explained by the fact that the filter gain has direct influence on the state estimates. The final estimate of the filter gain is

\[
K = \begin{bmatrix}
0.2223 \\
0.0527
\end{bmatrix}.
\]

The true and the predicted states for an EKF and a CGEKF are shown in Figure 6.5. The performance of both filters is given in Table 6.2. For these filters, \(Q\) (for the EKF) and \(K\) (for the CGEKF) are the final values that resulted from the adaptation algorithm (in order not to take adaptation transients into account, the performance was evaluated for the fully adapted filters). With regard to the first state (which is measured), there is not much difference in filter performance; for the second state (which is not measured), the EKF gives approximately 25% lower variance, which shows the advantage of using a time-varying filter gain. This advantage is also seen if the performance of the CGEKF is compared to the performance of the EKF shown in Table 6.1. The fact that some wrongly tuned filters outperform the optimal constant-gain filter indicates that the performance loss due to wrong covariance matrices can be smaller than the performance loss due to a constant filter gain.

As mentioned above, the EKF with the optimal process noise covariance matrix performs better than the one with the true process noise covariance matrix. This confirms the well-known fact that adding pseudonoise can improve estimation accuracy (Maybeck 1982, p. 24). Interestingly, the adaptation algorithm automatically determines an adequate amount of pseudonoise.
Figure 6.3 Estimated diagonal elements of $Q$ (left) and elements of their covariance matrix $P_{pp}$ (right) for the adaptive EKF

Figure 6.4 Estimated filter gains (left) and elements of their covariance matrix $P_{pp}$ (right) for the adaptive CGEKF

<table>
<thead>
<tr>
<th>Table 6.2 Performance of the fully adapted filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Type</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>EKF with $Q = \begin{bmatrix} 0.4941 &amp; 0 \ 0 &amp; 0.8716 \end{bmatrix}$, $R = 0.2$</td>
</tr>
<tr>
<td>CGEKF with $K = \begin{bmatrix} 0.2223 \ 0.0527 \end{bmatrix}$</td>
</tr>
</tbody>
</table>
Finally, the evolution of the filter gains for the optimal EKF is shown in Figure 6.6; the optimal constant gain is also included. It is evident that the filter gains vary over quite a large range and include sign changes; this again indicates that a constant-gain filter will inevitably lead to some loss in performance.

Figure 6.5 True and predicted states; solid: true states; dashed: EKF estimates; dotted: CGEKF estimates

Figure 6.6 Filter gains of the EKF (solid) and the CGEKF (dashed)

6.2 Parameter Estimation for a Van der Pol Oscillator (I)

This example illustrates the need for adaptive noise estimation in the context of system identification. First, the well-known problem of biased estimates obtained from a state-augmented EKF due to nonoptimal noise assumptions is demonstrated. It is then shown that adaptive
filters can overcome this problem. As in Ahmed (1994), a Van der Pol oscillator with one uncertain parameter described by

\[
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t)
\end{bmatrix} = \begin{bmatrix}
x_2(t) \\
-9x_1(t) + p_1 (1 - x^2_1(t)) x_2(t)
\end{bmatrix} + v(t),
\]

(6.3)

with \( p_1 = 2 \), is used. The system dynamics are thus identical to the system considered in the previous example. The measurement equation is given by

\[ y(t_k) = x_1(t_k) + w(t_k), \]

(6.4)

with \( t_k = kT \) and a sampling time of \( T = 0.05 \) seconds. The noise (co)variances were set to

\[ \text{E} v(t) v^T(t') = 0.5 \text{I} \delta(t - t') \]

for the process noise and

\[ \text{E} w(t_i) w(t_j) = 0.5 \delta_{ij} \]

for the measurement noise. The choice of the measurement noise covariance matrix corresponds to a quite large noise-to-signal ratio of approximately 25%. Again, the initial state is

\[ x(0) = \begin{bmatrix} -2 \\ -1 \end{bmatrix}. \]

Figure 6.7 shows the output signal of this Van der Pol oscillator.

First, a state-augmented EKF was used to estimate the unknown system parameter. The settings for this filter were chosen as

\[ \hat{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T, \quad \hat{p}_i(0) = 0, \quad P_{xx}(0) = \text{I}, \quad P_{xp}(0) = 0, \quad P_{pp}(0) = 1, \quad R = 1, \]

and three different values for \( Q \) were considered:

a) \( Q = 0.1 \text{I} \),

b) \( Q = \text{I} \),

c) \( Q = 10 \text{I} \).

The parameter estimate for the three cases is shown in Figure 6.8. It becomes clear that the final estimate depends on the choice of the matrix \( Q \) and that only for the case where the ratio of \( Q \) to \( R \) corresponds to the true values (case b), an unbiased estimate is obtained. This shows that the choice of \( Q \) is indeed crucial if the EKF is used as a parameter estimator.
Figure 6.7 Output signal of the Van der Pol oscillator

Figure 6.8 Parameter estimates obtained from three differently tuned state-augmented EKFs

Ljung’s (1979) discussion suggests two possible sources for this bias problem: the wrongly tuned covariance matrices and the approximate gradient computation of the EKF. Therefore, as the next step, an adaptive EKF without adaptation of $Q$ was implemented. This corresponds to a “corrected” gradient computation, but this algorithm still uses fixed noise covariance matrices. The settings for this filter were chosen as

$$
\hat{x}(0) = [0 \ 0]^T, \ P_{xx}(0) = I, \ \hat{p}_i(0) = 0, \ \frac{d\hat{x}(0)}{d\hat{p}_1} = [0 \ 0]^T, \ \frac{dP_{xx}(0)}{d\hat{p}_1} = 0, \ P_{pp}(0) = 1, \ R = 1,
$$

and the same values for $Q$ as above were considered.

Figure 6.9 shows the resulting parameter estimate for the three cases. The situation is essentially unchanged: the parameter estimate still depends on the choice of the matrix $Q$, and only for a correct choice is an unbiased estimate obtained (case b). This clearly demonstrates that an improved gradient computation alone cannot overcome the bias problem and that it is necessary to adjust the process noise covariance matrix as well.
This was done in the next experiment, where an adaptive EKF with adaptation of $Q$ was used. The settings for this filter were selected as

$$
\begin{align*}
\dot{x}(0) &= [0 \ 0]^T, \ P_{xx}(0) = I, \ Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \hat{p}_2^2 & 0 \\ 0 & \frac{1}{2} \hat{p}_3^2 \end{bmatrix}, \ P_{pp}(0) = \begin{bmatrix} 0 & \sqrt{0.2} \\ \sqrt{0.2} & \sqrt{0.2} \end{bmatrix}^T, \ \frac{d\hat{x}(0)}{dp^T} &= 0, \\
\frac{dP_{xx}(0)}{dp^T} &= 0, \ P_{pp}(0) = I, \ R = 1.
\end{align*}
$$

The results of this experiment are shown in Figure 6.10. An accurate estimate is clearly obtained.

Finally, an adaptive CGEKF with the settings

$$
\begin{align*}
\dot{x}(0) &= [0 \ 0]^T, \ K(\hat{p}) = [\hat{p}_2 \ \hat{p}_3]^T, \ P_{pp}(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}^T, \ \frac{d\hat{x}(0)}{dp^T} &= 0, \ P_{pp}(0) = I, \ A = 1
\end{align*}
$$

was applied. The result shown in Figure 6.11 demonstrates that this adaptive filter also overcomes the problem of incorrect noise assumptions and gives an accurate parameter estimate. Overall, this example illustrates the necessity of adaptive noise estimation for system parameter estimation.
6.3 Parameter Estimation for a Van der Pol Oscillator (II)

This example was considered in Bohn (2000). The system is the Van der Pol oscillator that was used by Astolfi et al. (1996) and is described by

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\omega^2 x_1(t) + 2 \omega \nu x_2(t) - 2 \omega \nu \mu x_1^2(t) x_2(t) \end{bmatrix} + v(t),$$  \hspace{1cm} (6.5)

with $\omega = 0.5$, $\nu = 1$, and $\mu = 2$. Astolfi et al. (1996) claim that the system dynamics are nonlinear in the parameters; this is correct, however, the simple reparameterization $p_1 = \omega^2$, $p_2 = 2 \omega \nu$, and $p_3 = 2 \omega \nu \mu$ makes the system dynamics linear in the parameters. The system dynamics can then be written as

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -p_1 x_1(t) + p_2 x_2(t) - p_3 x_1^2(t) x_2(t) \end{bmatrix} + v(t),$$  \hspace{1cm} (6.6)

with $p_1 = 0.25$, $p_2 = 1$, and $p_3 = 2$. The initial state was set to

$$x(0) = \begin{bmatrix} -0.75 \\ -1.35 \end{bmatrix}.$$  

The measurement equation is given by

$$y(t_k) = x_1(t_k) + w(t_k),$$  \hspace{1cm} (6.7)

with $t_k = kT$ and a sampling time of $T = 0.1$ seconds. The noise (co)variances are

$$\mathbb{E} v(t) v^\top(t') = 0.01 I \delta(t-t')$$

for the process noise and
for the measurement noise. The choice of the measurement noise variance corresponds to a noise-to-signal ratio of 8% (which is quite large compared to the settings of Astolfi et al. [1996], who used measurement noise with variance 0.025 and no process noise). To illustrate the influence of the process noise, parts of the state trajectories are shown as phase-plane plots for the noise-free and the noisy case in Figure 6.12. The output signal is shown in Figure 6.13.

The system considered in this example is a relatively “harmless” one, in the sense that all three identification algorithms give accurate results (this includes the fact that EKF estimates are insensitive to the noise covariance matrices). For the state-augmented EKF, the settings

\[
\hat{x}(0) = [0 \ 0]^T, \ \hat{p}(0) = [0 \ 0 \ 0]^T, \ P_{xx}(0) = 10 I, \ P_{xp}(0) = 0, \ P_{pp}(0) = I, \ Q = I, \ R = 1
\]

were used, and “start-up parameter pseudonoise” with

\[
Q_{pp}(t) = 0.1 \exp(-t/50) I
\]

was included to accelerate convergence.

Figure 6.12 State trajectories of the Van der Pol oscillator, noise-free (left) and noisy (right)

Figure 6.13 Output signal of the Van der Pol oscillator
For the adaptive EKF, the settings
\[
\dot{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T, \quad P_{xx}(0) = 10 \mathbf{I}, \quad Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \dot{p}_4^2 & 0 \\ 0 & \frac{1}{2} \dot{p}_5^2 \end{bmatrix}, \quad \hat{p}(0) = \begin{bmatrix} 0 \\ 0 \\ \sqrt{0.2} \\ \sqrt{0.2} \end{bmatrix}^T,
\]
\[
\frac{d\hat{x}(0)}{d\hat{p}^T} = 0, \quad \frac{dP_{xx}(0)}{d\hat{p}^T} = 0, \quad P_{pp}(0) = 10 \mathbf{I}, \quad R = 1,
\]
and for the adaptive CGEKF, the settings
\[
\dot{x}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}^T, \quad K(\hat{p}) = \begin{bmatrix} \dot{p}_4 \\ \dot{p}_5 \end{bmatrix}^T, \quad \hat{p}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T, \quad \frac{d\hat{x}(0)}{d\hat{p}^T} = 0, \quad P_{pp}(0) = \mathbf{I}, \quad A = 1
\]
were selected. For both adaptive filters, a start-up forgetting factor (Wellstead and Zarrop 1991, p. 158)
\[
\lambda(t_k) = 1 - 0.03 \exp(-t_k/50)
\]
was used to accelerate the parameter estimation. The duration of the identification experiment was 400 seconds.

Figures 6.14 – 6.16 show the parameter estimates obtained from the three different filters. As mentioned above, all filters give accurate parameter estimates for this example. This indicates that it is difficult to compare algorithms for nonlinear system identification and that this can only be done for a specific application, if at all. Furthermore, there are many tuning parameters involved, and when more time is spent on tuning, for example, in simulation studies, a set of parameters for which a specific algorithm works might be located.

**Figure 6.14** Parameter estimates obtained from the state-augmented EKF
6.4 Parameter Estimation for a Van der Pol Oscillator (III)

In this example, the Van der Pol oscillator of the previous section is considered again. Here, however, the measurement equation is modified. The system is given as

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t) \\
\dot{x}_2(t) &= -p_1 x_1(t) + p_2 x_2(t) - p_3 x_1^2(t) x_2(t) + v(t),
\end{align*}
\]

(6.8)

\[y(t_k) = x_1(t_k) + x_1(t_k) x_2(t_k) + w(t_k),\]  

(6.9)

with \(t_k = kT\) and a sampling time of \(T = 0.1\) seconds. The parameter values are, as before, \(p_1 = 0.25\), \(p_2 = 1\), and \(p_3 = 2\). The noise (co)variances were set to

\[
E v(t)v^\top(t') = 0.01 I \delta(t - t')
\]

for the process noise and
for the measurement noise. The choice of the measurement noise variance corresponds to a noise-to-signal ratio of 7%. The initial state was chosen as

\[ x(0) = \begin{bmatrix} -0.75 \\ -1.35 \end{bmatrix}. \]

Figure 6.17 shows the output signal for this system.

![Figure 6.17 Output signal of the Van der Pol oscillator](image)

For this example, a CGEKF cannot be used because the filter gain, which is given as

\[ K = P_{xx} \frac{\partial h}{\partial x} A^{-1} = \begin{bmatrix} P_{xx,11} & P_{xx,12} \\ P_{xx,12} & P_{xx,22} \end{bmatrix} \begin{bmatrix} \hat{x}_2 + 1 \\ \hat{x}_1 \end{bmatrix} A^{-1}, \]

will vary in time with \( \hat{x} \), even if the entries of the covariance matrices converged. Furthermore, as observed from an EKF for this example, the gain elements are not only time-varying, but they also change sign. Only filters that accommodate for these sign changes are able to track the states. Therefore, a constant-gain filter does not work for this example: for larger gains the constant-gain filter becomes unstable; and if the gain is chosen small enough such that the filter remains stable, the filter is effectively decoupled from the system, and the estimated states do not follow the true states. The stochastic-gradient algorithm that is used to update the parameters is based on constant or slowly time-varying parameters (slower than the states). It cannot be used to track parameters that vary fast; this, in fact, could be seen as the reason why states are not estimated with a gradient algorithm (Jakoby and Pandit 1987). With a constant forgetting factor, the algorithm tries to account for the time-varying filter gain, but it finally becomes unstable as well.
This constitutes a general drawback of the CGEKF: if a time-varying filter gain is required to stabilize the EKF, the CGEKF cannot be used; consequently, the parameters of such a system cannot be estimated through an adaptive CGEKF. Thus, before such a filter is applied, the question whether a time-varying gain is required has to be answered. Since no general indicators are available as to when a given nonlinear system requires a time-varying gain, simulation studies seem to be the only tool. Simulation studies, however, require some initial estimates or typical values for the system parameters. (The stability aspect discussed here is related to the sign condition for the gain in stochastic approximation algorithms, see Gelb [1984], Equation [9.3-7a] and Albert and Gardner [1967], Section 9.3 for the scalar case; and Albert and Gardner [1967], p. 84, Assumption B5 for the vector case.)

Thus, in this example, only the state-augmented EKF and the adaptive EKF could be used. Both algorithms give accurate estimates, yet these results are not too surprising since the system dynamics are identical to the previous example, where all algorithms worked well.

The settings for the state-augmented EKF were chosen as
\[
\dot{x}(0) = \begin{bmatrix} -1 & 0 \end{bmatrix}^T, \quad \hat{p}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T, \quad P_{xx}(0) = 10 I, \quad P_{xp}(0) = 0, \quad P_{pp}(0) = 10 I, \quad Q = I, \quad R = 1,
\]
and start-up parameter pseudonoise of
\[
Q_{pp}(t) = 0.1 \exp(-t/50) I
\]
was used for faster convergence.

For the adaptive EKF, the settings
\[
\dot{x}(0) = \begin{bmatrix} -1 & 0 \end{bmatrix}^T, \quad P_{xx}(0) = I, \quad Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \hat{p}_4^2 & 0 \\ 0 & \frac{1}{2} \hat{p}_5^2 \end{bmatrix}, \quad \hat{p}(0) = \begin{bmatrix} 0 & 0 & \sqrt{2} & \sqrt{2} \end{bmatrix}^T,
\]
\[
\frac{d\dot{x}(0)}{dp} = 0, \quad \frac{dP_{xx}(0)}{dp} = 0, \quad P_{pp}(0) = I, \quad R = 1,
\]
were selected. A start-up forgetting factor
\[
\lambda(t_k) = 1 - 0.03 \exp(-t_k/50)
\]
was included.

Figures 6.18 and 6.19 show the parameter estimates obtained from the state-augmented EKF and the adaptive EKF, respectively. As pointed out above, both algorithms give accurate results.
6.5 Identification of a Wiener System

The following examples demonstrate some further potential applications of the identification algorithm developed in this dissertation. Only the adaptive EKF is considered. The general drawbacks of the two other algorithms are discussed above; however, it is also shown that they do not exhibit these drawbacks in all cases. Therefore, for the systems considered below, it might be possible to use the state-augmented EKF as well as the adaptive CGEKF. As pointed out above, a comparison is difficult and is not attempted in the following examples.

A linear dynamic system with a static nonlinearity in the output path (a so-called Wiener system) is considered next. The linear part of the system is a second-order oscillator given in state-space form as
\[ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\xi \omega_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ k\omega_0^2 \end{bmatrix} u. \] (6.11)

The output of the system is given as
\[ y(t_k) = \begin{cases} \frac{2}{3} x_1(t_k) - \frac{1}{3} + w(t_k) & \text{if } x_1(t_k) \leq -\frac{1}{4} \\ 2x_1(t_k) + w(t_k) & \text{if } |x_1(t_k)| < \frac{1}{4} \\ \frac{2}{3} x_1(t_k) + \frac{1}{3} + w(t_k) & \text{if } x_1(t_k) \geq \frac{1}{4} , \end{cases} \] (6.12)

with \( t_k = kT \) and a sampling time of \( T = 0.1 \) seconds. The system parameters are defined as \( p_1 := \omega_0^2 = 40 \) and \( p_2 := 2\xi \omega_0 = 1.25 \), which corresponds to a natural frequency of approximately \( 2\pi \) and a damping coefficient of approximately 0.1.

For the identification, it was ensured that the parameter estimates \( \hat{p}_1 \) and \( \hat{p}_2 \) remained positive (so that the predictor remained stable) by reparameterizing them as
\[ \hat{p}_i := \frac{1}{2} \tilde{p}_i^2, \quad i = 1, 2 \]
and estimating \( \tilde{p}_1 \) and \( \tilde{p}_2 \). In the predictor, the output nonlinearity is approximated by a weighted sum of five Chebyshev polynomials (Unbehauen and Rao 1987, Table 3.6a) with the absolute term removed, that is, the predicted output is given as
\[ \hat{y} = \hat{p}_1 \tilde{T}_1(\hat{x}_1) + \hat{p}_2 \tilde{T}_2(\hat{x}_1) + \hat{p}_3 \tilde{T}_3(\hat{x}_1) + \hat{p}_4 \tilde{T}_4(\hat{x}_1) + \hat{p}_5 \tilde{T}_5(\hat{x}_1) \] (6.13)
with
\[ \tilde{T}_1(x) = x, \] (6.14)
\[ \tilde{T}_2(x) = 2x^2, \] (6.15)
\[ \tilde{T}_3(x) = 4x^3 - 3x, \] (6.16)
\[ \tilde{T}_4(x) = 8x^4 - 8x^2, \] (6.17)
and
\[ \tilde{T}_5(x) = 16x^5 - 20x^3 + 5x. \] (6.18)

The gain of the linear system is given as \( k = 0.5 \). This gain was also used in the predictor, which does not constitute a loss of generality since the gain is absorbed by the output nonlinearity; thus, the gain is estimated as well. Here, the gain of the linear part was chosen such that the state \( x_1 \) varies between minus one and one.
A uniformly distributed random white sequence with an amplitude of one and a sampling time of two seconds was chosen as the excitation signal. No process noise was used; and measurement noise with a variance of 0.1 was added to the output, corresponding to a noise-to-signal ratio of 9%. The system was initially at rest, that is,

\[ x(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

The input signal and the output signal are shown in Figures 6.20 and 6.21, respectively.

The total duration of the identification experiment was 500 seconds. The settings for the adaptive EKF were chosen as

\[ \hat{x}(0) = [0 \ 0]^T, \quad P_{xx}(0) = I, \quad \hat{p}(0) = [0.5 \ 0.5 \ 1 \ 0 \ 0 \ 0 \ \sqrt{2} \ \sqrt{2}]^T, \]

\[ Q(\hat{p}) = \begin{bmatrix} \frac{1}{\hat{\rho}_8} & 0 \\ 0 & \frac{1}{\hat{\rho}_9} \end{bmatrix}, \quad \frac{d\hat{x}(0)}{d\hat{p}} = 0, \quad \frac{dP_{xx}(0)}{d\hat{p}} = 0, \quad P_{pp}(0) = I, \quad R = 1, \]

and a start-up forgetting factor

\[ \lambda(t_k) = 1 - 0.03 \exp(-t_k/50) \]

was used.

Figure 6.22 shows the estimated system parameters. After 500 seconds, the final estimates were 40.06 and 1.30, which corresponds to an error of 0.16% and 4.25%, respectively. The estimated weights for the five Chebyshev polynomials are shown in Figure 6.23.

With these estimated weights, the output nonlinearity is approximated by the polynomial

\[ \hat{y} = 2.3089 \hat{x}_5^5 - 0.0083 \hat{x}_4^4 - 3.0789 \hat{x}_3^3 + 0.0267 \hat{x}_2^2 + 2.0220 \hat{x}_1, \]

which is (almost) an odd function. The true and the approximated nonlinearity are portrayed in Figure 6.24. The sum of the Chebyshev polynomials approximates the gain-changing characteristics of the output nonlinearity quite well.
6.5 Identification of a Wiener System

Figure 6.20 Input signal of the Wiener system

Figure 6.21 Output signal of the Wiener system

Figure 6.22 Estimates of the system parameters

Figure 6.23 Estimated weights of the Chebyshev polynomials for the output nonlinearity
6.6 Parameter Estimation and Identification of an Input Nonlinearity (I)

The system identified in this example is motivated by the physical model of a dc motor. The model for the dc motor is taken from Mohler (1991, Example 4.8), but it is modified to include a nonlinear relationship between the field current and the magnetic flux. The system dynamics are given as

\[
\frac{di_a}{dt} = -\frac{R_a}{L_a}i_a - \frac{k}{L_a}\psi(i_f)\omega + \frac{v_a}{L_a}, \tag{6.19}
\]

\[
\frac{d\omega}{dt} = -\frac{D}{J}\omega + \frac{k}{J}\psi(i_f)i_a, \tag{6.20}
\]

where \(i_a\) is the armature current, \(\psi\) is the magnetic flux, \(i_f\) is the field current, \(v_a\) is the armature voltage, and \(\omega\) is the angular speed; \(R_a\) and \(L_a\) are the armature resistance and the armature inductivity, respectively; \(J\) and \(D\) are the combined moment of inertia (motor and load) and the viscous damping ratio, respectively; \(k\) is a constant.

Introducing the relationship

\[
g(i_f) = \frac{k}{L_a}\psi(i_f), \tag{6.21}
\]

defining the parameters

\[
p_1 = \frac{R_a}{L_a}, \quad p_2 = \frac{1}{L_a}, \quad p_3 = \frac{D}{J}, \quad p_4 = \frac{L_a}{J},
\]

and considering the field current as the input signal and \(i_a\) and \(\omega\) as states, that is,
\( u := i_t \), \( x_1 := i_a \), \( x_2 := \omega \),

the system dynamics can be written as

\[
\begin{align*}
\dot{x}_1 &= -p_1 x_1 - g(u)x_2 + p_2 v_a + v, \\
\dot{x}_2 &= -p_3 x_2 + p_4 g(u)x_1
\end{align*}
\]  

(6.22)

where process noise \( v \) is included.

Using the data of Mohler (1991), the parameters have the following values (the dimensions are omitted):

\( p_1 = 20 \), \( p_2 = 20 \), \( p_3 = 0.5 \), and \( p_4 = 35.43 \).

Here, the relationship between the magnetic flux and the field current is described by a saturation characteristic as

\( g(u) = 1.4114 \tanh (3u) \),  

(6.23)

where the constant 1.4114 is obtained from Mohler’s (1991) value for \( k/L_a \). It is not claimed that the saturation characteristic resembles a physically realistic flux-current dependency of a dc motor; rather, the aim here is to construct a highly nonlinear system that is then identified with the proposed estimation algorithm.

For identification, the function \( g \) was approximated in the predictor by a weighted sum of shifted Chebyshev polynomials (shifted so that the argument lies between zero and one, see Unbehauen and Rao [1987, p. 70]), with the absolute terms removed, that is,

\[
\hat{g}(u) = \hat{p}_3 T'_3(u) + \hat{p}_6 T'_6(u) + \hat{p}_9 T'_9(u) + \hat{p}_{12} T'_{12}(u) + \hat{p}_{15} T'_{15}(u),
\]

(6.24)

with

\[
T'_3(x) = 2u,
\]

(6.25)

\[
T'_6(x) = 8u^2 + 8u,
\]

(6.26)

\[
T'_9(u) = 32u^3 - 48u^2 + 18u,
\]

(6.27)

\[
T'_9(u) = 128u^4 - 256u^3 + 160u^2 - 32u,
\]

(6.28)

and

\[
T'_5(u) = 512u^5 - 1280u^4 + 1120u^3 - 400u^2 + 50u.
\]

(6.29)

The model used in the predictor is thus linear in the states, nonlinear in the input, and nonlinear in the parameters because of the product term \( p_4 g(u) \) in the second state equation.
A total of 11 parameters were estimated (four system parameters, five weights of the polynomials, and two diagonal elements of the process noise covariance). The output signals are the states with added measurement noise, that is, the output vector is given as

\[ y(t_k) = x(t_k) + w(t_k), \]

(6.30)

with \( t_k = kT \) and a sampling time of \( T = 0.1 \) seconds.

It is assumed that the field current is under perfect control, so that a piecewise constant field current can be applied. In the simulation, this piecewise constant field current was obtained from resampling a white sequence uniformly distributed between 0.05 and 1 with a sampling time of two seconds filtered through a first-order lag with a time constant of 0.1 seconds. In reality, a piecewise constant field current could not be achieved due to the inductivity in the field circuit. The assumption of perfect control, however, would be justified if the field current control loop acted much faster than the remaining system dynamics.

In the identification experiment, process noise with a covariance matrix of

\[
E v(t)v^T(t') = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.1 \end{bmatrix} \delta(t-t')
\]

and measurement noise with a covariance matrix of

\[
E w(t_i)w^T(t_i) = \begin{bmatrix} 0.5 & 0 \\ 0 & 25 \end{bmatrix} \delta_y
\]

were used. The measurement noise strength corresponds to noise-to-signal ratios of 16% and 10% for the first and the second output, respectively. The initial state was set to

\[
x(0) = \begin{bmatrix} 1.5 \\ 135 \end{bmatrix}.
\]

The settings for the identification experiment were selected as

\[
\dot{x}(0) = \begin{bmatrix} 2 & 130 \end{bmatrix}^T, \quad P_{xx}(0) = 10I, \quad \dot{p}(0) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} \end{bmatrix}^T,
\]

\[
Q(\dot{p}) = \begin{bmatrix} \frac{1}{2} \hat{P}_{10}^2 & 0 \\ 0 & \frac{1}{2} \hat{P}_{11}^2 \end{bmatrix}, \quad \frac{d\dot{x}(0)}{dp^T} = 0, \quad \frac{dP_{xx}(0)}{dp^T} = 0, \quad P_{xx}(0) = 1, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix},
\]

and a start-up forgetting factor of

\[
\lambda(t_k) = 1 - 0.03 \exp \left( -t_k / 50 \right)
\]

was used. The total duration of the experiment was 500 seconds. The input, the first and the second output signal are shown in Figures 6.25, 6.26, and 6.27, respectively.
Figure 6.25 Input signal (field current)

Figure 6.26 Output signal $y_1$ (armature current)

Figure 6.27 Output signal $y_2$ (angular speed)

Figure 6.28 shows the estimated system parameters. After 500 seconds, the parameter estimates were $\hat{p}_1 = 19.72$, $\hat{p}_2 = 19.67$, $\hat{p}_3 = 0.52$, and $\hat{p}_4 = 36.91$, which corresponds to errors of 1.4%, 1.6%, 3.2%, and 4.2%, respectively. The estimated weights of the five Chebyshev polynomials are shown in Figure 6.29.

With the estimated weights, the saturation characteristic is approximated by the polynomial
\[
\hat{g}(u) = -3.6562u^5 + 9.3038u^4 - 6.0856u^3 - 2.5683u^2 + 4.3887u.
\] (6.31)

The true and the approximated saturation characteristics are portrayed in Figure 6.30; the polynomial is an excellent approximation of the true nonlinearity.

**Figure 6.28** Estimates of the system parameters

**Figure 6.29** Estimated weights of the Chebyshev polynomials for the input nonlinearity

**Figure 6.30** True (dashed) and identified input nonlinearity (solid)
6.7 Parameter Estimation and Identification of an Input Nonlinearity (II)

As the last example of this chapter, a mass-spring-damper system with a time-varying mass is considered (this is motivated by the filling system example considered by Jakoby and Pandit [1987]). The system is schematically shown in Figure 6.31.

A stream of material $m_{in}(t)$ with a constant velocity $v$ falls into a container that is part of a spring-damper suspension system. The spring force is assumed to be given according to Hooke’s law as $F_{spring} = ky$, where $y$ is the displacement and $k$ a constant; and the damper force is given as $F_{damper} = d\dot{y}$, where $d$ is a constant. The stream of material entering the container results in a force of $F = m_{in}v$ (the velocity $v$ is considered to be large compared to $\dot{y}$, so that $v$ can be used instead of the difference $v - \dot{y}$). The incoming material flow is related to an input signal $u$ as

$$m_{in}(t) = m_{in,0} + K \bar{u}(t). \quad (6.32)$$

The mass in the container is given by $m_{0} + m(t)$, where $m_{0}$ is the steady-state value of the mass for $u = 0$ and $m(t)$ the deviation from this steady-state value. The flow of material out of the container is governed by

$$m_{out}(t) = q \sqrt{m_{0} + m(t)}, \quad (6.33)$$

where $q$ is a constant (this is different from Jakoby and Pandit [1987], who considered a container without an outlet). The system can then be described by the differential equations

$$(m_{0} + m(t))\ddot{y}(t) + d\dot{y}(t) + ky(t) = m(t)g + Kv \bar{u}(t) \quad (6.34)$$

and
\[ \dot{m}(t) = -q\sqrt{m_0 + m(t)} + K \bar{u}(t), \]  
(6.35)  

where \( g \) is the acceleration of gravity.

By introducing the states \( x_1 := y, \ x_2 := \ddot{y}, \) and \( x_3 := m, \) the system can be written as a state-space model

\[ \dot{x}_1 = x_2, \]  
(6.36)  

\[ \dot{x}_2(t) = \frac{-kx_1(t) - dx_2(t) + gx_1(t) + Kv \bar{u}(t)}{m_0 + x_3(t)}, \]  
(6.37)  

\[ \dot{x}_3(t) = -q\sqrt{m_0 + x_3(t)} + K \bar{u}(t). \]  
(6.38)  

Only the displacement of the container is measured, that is, the measurement equation is given as

\[ y(t_k) = x_1(t_k) + w(t_k), \]  
(6.39)  

with \( t_k = kT \) and a sampling time of \( T = 0.1 \) seconds. Additionally, an input nonlinearity is introduced into the system by passing the input signal through a deadzone with a slope of one and lower and upper bounds of \(-0.2\) and \(0.4\), respectively, that is,

\[ \bar{u} = g(u), \]  
(6.40)  

with

\[ g(u) = \begin{cases} 
  u + 0.2 & \text{for } u < -0.2 \\
  0 & \text{for } -0.2 \leq u \leq 0.4 \\
  u - 0.4 & \text{for } u > 0.4, 
\end{cases} \]  
(6.41)  

(again, this differs from Jakoby and Pandit’s [1987] example, which did not include an input nonlinearity).

The parameter values are taken as

\[ k = 81, \ d = 45, \ g = 10, \ Kv = 200, \ m_0 = 25, \ q = 3, \text{ and } K = 6. \]

The spring constant \( k \), the damper constant \( d \), and the outlet parameter \( q \) are unknown, whereas all other parameters are known. Defining \( p_1 := k \), \( p_2 := d \), and \( p_3 := q \), the model can be written as

\[ \dot{x}_1 = x_2, \]  
(6.42)  

\[ \dot{x}_2(t) = \frac{-p_1x_1(t) - p_2x_2(t) + 10x_1(t) + 200 g(u(t))}{25 + x_3(t)}, \]  
(6.43)
\[ \dot{x}_1(t) = -p_3 \sqrt{25 + x_1(t)} + 6 g(u(t)). \quad (6.44) \]

In addition to the three unknown system parameters, the input nonlinearity should be identified. For this purpose, it is approximated by a weighted sum of five Chebyshev polynomials (with the absolute term removed), that is,
\[ \hat{g}(u) = \hat{p}_4 \tilde{T}_4(u) + \hat{p}_3 \tilde{T}_3(u) + \hat{p}_2 \tilde{T}_2(u) + \hat{p}_1 \tilde{T}_1(u) + \hat{p}_0 \tilde{T}_0(u). \quad (6.45) \]

This corresponds to an unknown gain since the system gain is included in the input nonlinearity. To prevent the estimated system parameters from becoming negative, they are parameterized as
\[ \hat{p}_i = \frac{1}{2} \bar{p}_i^2, \quad i = 1, 2, 3, \]
and \( \bar{p}_1, \bar{p}_2, \) and \( \bar{p}_3 \) are estimated.

As an input signal, a uniformly distributed random white sequence with an amplitude of one and a sampling time of one second was used. The duration of the identification experiment was 500 seconds. Measurement noise of variance 0.07 was added to the output (corresponding to a noise-to-signal ratio of 9%). Although no process noise was used, the process noise assumption still makes sense since the difference between the input deadzone and its approximation can be interpreted as process noise.

Figures 6.32 – 6.36 show the input signal, the states, and the measured output signal for this experiment.

The settings for the identification experiment were selected as
\[
\begin{align*}
x(0) &= [0 \ 0 \ 0]^T, \quad \dot{x}(0) = [0 \ 0 \ 0]^T, \quad P_{xx}(0) = I, \quad Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \bar{p}_1^2 & 0 & 0 \\ 0 & \frac{1}{2} \bar{p}_2^2 & 0 \\ 0 & 0 & \frac{1}{2} \bar{p}_3^2 \end{bmatrix}, \\
\hat{p}(0) &= [50 \ 50 \ 0.5 \ 1 \ 0 \ 0 \ 0 \ 0 \ \sqrt{2} \ \sqrt{2} \ \sqrt{2}]^T, \quad \frac{d\hat{x}(0)}{d\hat{p}} = 0, \quad \frac{dP_{xx}(0)}{d\hat{p}} = 0, \\
P_{pp}(0) &= I, \quad R = 1,
\end{align*}
\]

and a start-up forgetting factor of
\[ \lambda(t_\kappa) = 1 - 0.03 \exp(-t_\kappa/50) \]
was applied. The estimated system parameters are shown in Figure 6.37. After 500 seconds, the estimates obtained were 79.87, 43.30, and 3.04, which corresponds to errors of 1.4%, 3.8%, and 1.4%, respectively.
CHAPTER 6: SIMULATION STUDIES

Figure 6.32 Input signal for the oscillatory system

Figure 6.33 Evolution of the state $x_1$

Figure 6.34 Evolution of the state $x_2$

Figure 6.35 Evolution of the state $x_3$
Figure 6.36 Measured output signal

Figure 6.37 Estimates of the system parameters

Figure 6.38 shows the estimated weights for the Chebyshev polynomials; with these weights, the input nonlinearity is approximated by the polynomial

$$\hat{g}(u) = -0.7913 u^5 + 0.2225 u^4 + 1.3741 u^3 - 0.2865 u^2 + 0.0641 u.$$ 

The true and the approximated input nonlinearity are portrayed in Figure 6.39; again, the polynomial approximates the deadzone quite well. A better approximation could be obtained if more Chebyshev polynomials were used. Alternatively, other approximations could be employed, for example, different systems of polynomials or local approximations such as neural networks. Since the identification of these static nonlinearities corresponds to black-box modeling (the type of the nonlinearity present is initially completely unknown and not derived from first principles), these examples demonstrate that the identification algorithm is also applicable to black-box models.
In this chapter, the algorithms presented in Chapter 4 have been applied to several simulation examples. Since parameter estimation algorithms for nonlinear systems are usually based on nonlinear optimization methods, no general results regarding convergence or stability are available. It is, therefore, difficult to compare these algorithms. Such a comparison can either be based on specific examples or on intuitive concepts rather than on mathematical rigor.

From the examples presented in this chapter and from other examples considered by the author of this dissertation, the following conclusions can be drawn:

- Since the noise statistics are generally unknown, it is advisable to use adaptive filtering for state estimation. The adaptation should preferably be carried out with respect to the noise covariance matrices rather than the filter gain since this keeps the filter gain time-varying. The time-varying filter gain will give more accurate state estimates (see
Section 6.1). Interestingly, the adaptive filtering algorithm automatically adds pseudonoise.

- The state-augmented EKF is a simple and straightforward algorithm for the joint estimation of states and parameters. It often gives accurate results; however, biased estimates may occur if the noise covariance matrices are improperly tuned (see Section 6.2.) Unless simulation studies indicate that the state-augmented filter is relatively insensitive to the covariance matrices, the EKF should not be used to obtain the final estimates.

- The bias problem of the state-augmented EKF stems from incorrect noise assumptions. A corrected gradient alone cannot overcome this problem; therefore, it is necessary to adjust the noise statistics (or the filter gain) as well (see Section 6.2).

- The adaptive CGEKF avoids the computational complexity of the adaptive EKF. Nevertheless, it often works well (see Sections 6.2 and 6.3). Yet it tends to give less accurate state estimates (see Section 6.1); furthermore, it cannot be applied to all nonlinear systems since some systems will require a time-varying filter gain for a stable filter (see Section 6.4). Again, simulation studies are a tool to analyze this algorithm, if typical parameter values are available.

- The adaptive EKF gives more accurate state estimates than the constant-gain filter and can also be applied to systems which require a time-varying filter gain. However, this is achieved at the cost of increased computational complexity. Therefore, at least for online parameter estimation, the algorithm seems to be limited to low-dimensional systems with not too many unknown system parameters or to higher-dimensional systems with large sampling times.

- If certain parameters are known to lie in specific ranges or must be constrained to ensure a stable predictor, these constraints should be incorporated in the parameterization (see Sections 6.5 and 6.7, where the system parameters have been restricted to positive values).

- Convergence of the parameter estimates was found to be slow for all algorithms considered. In almost all examples, including a start-up forgetting factor or parameter pseudonoise has been found necessary to increase the convergence speed.

Generally, the identification of a nonlinear system is a complex task that incorporates the modeling stage and the selection of a suitable estimation algorithm. “It is generally agreed that no one technique can be recommended to ensure an acceptable solution” (Unbehauen and Rao 1987, p. 330), so it might be necessary to try different algorithms and to spend a significant amount of time on tuning to obtain satisfactory results.


Chapter 7

Real-Time Identification

The adaptive Extended Kalman Filter developed in Chapter 4 was applied to real-time parameter estimation and tracking for a three-tank system. In this chapter, the results of two real-time experiments are presented. The three-tank system, which is schematically shown in Figure 7.1, is described in detail in Appendix D.

![Figure 7.1 Three-Tank System](image)

In the operating set-up considered here, the two valves connecting tank 1 and tank 3 and tank 3 and tank 2 are open, one or both outlets of tank 2 are open, and the remaining outlets are closed. The system is modeled in the state space as

\[
\dot{x} = f(x, u, p) + v,
\]

where the state vector

\[
x = [x_1 \ x_2 \ x_3]^T
\]

contains the water levels in the three tanks. Assuming that the flows through the valves and the outlets follow Toricelli’s law (that is, they are proportional to the square root of the corre-
sponding level difference) and that the pumps can be modeled as pure proportional elements, the state transition function is given as

\[
f(x, u, p) = \begin{bmatrix}
- p_1 \sqrt{x_1 - x_3} + p_4 u_1 \\
p_2 \sqrt{x_3 - x_2} - p_3 \sqrt{x_2 + p_5 u_2} \\
p_1 \sqrt{x_1 - x_3} - p_2 \sqrt{x_3 - x_2}
\end{bmatrix}.
\]

The vector \( u = [u_1 \quad u_2]^T \) is the input vector that contains the normalized pump command inputs. The parameter vector \( p = [p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5]^T \) consists of the parameters that determine the flow through the connecting valves \( (p_1 \text{ and } p_2) \), the combined flow through the outlets of tank 3 \( (p_3) \), and the gains of the two pumps \( (p_4 \text{ and } p_5) \). Since all water levels are measured, the output is given as

\[ y = x + w. \]

### 7.1 Estimation of Constant Parameters

In the first identification experiment, a constant parameter vector was estimated. The total time was 5000 seconds, and the sampling time was 5 seconds; thus, 1000 samples of the output signal were collected. The parameters were estimated from the first 200 samples, and the remaining 800 samples were used for model validation. Both outlets of tank 2 were open. The settings for this experiment were chosen as

\[
\begin{align*}
\hat{x}(0) &= [47 \quad 21 \quad 34]^T, \quad P_{xx}(0) = I, \\
\hat{p}(0) &= [0 \quad 0 \quad 0 \quad 0 \quad \sqrt{0.2} \quad \sqrt{0.2} \quad \sqrt{0.2}]^T,
\end{align*}
\]

\[
Q(\hat{p}) = \begin{bmatrix}
\frac{1}{4} \hat{p}_6^2 & 0 & 0 \\
0 & \frac{1}{2} \hat{p}_7^2 & 0 \\
0 & 0 & \frac{1}{2} \hat{p}_8^2
\end{bmatrix}, \quad \frac{\text{d} \hat{x}(0)}{\text{d} \hat{p}^T} = 0, \quad \frac{\text{d} P_{xx}(0)}{\text{d} \hat{p}^T} = 0, \quad P_{pp}(0) = I, \quad R = 0.1 I.
\]

The algorithm was quite insensitive to these settings, and these values represent a reasonable first guess. Figure 7.2 shows the input and output signals for this experiment. The estimated parameters are portrayed in Figure 7.3.

The final parameter estimates \( \hat{p}_1 = 0.0692, \quad \hat{p}_2 = 0.0655, \quad \hat{p}_3 = 0.1970, \quad \hat{p}_4 = 0.7799, \) and \( \hat{p}_5 = 0.8158 \) are in excellent accordance with the parameter values that have been determined from offline experiments as described in Appendix D. The differences between both estimates are 1.8%, 1.5%, 1.2%, 5.2%, and 0.8%, respectively.
Figure 7.2 Input and output signals for the first experiment

Figure 7.3 Estimated parameters for the first experiment
For model validation, the remaining 800 samples of the measured water levels were compared against the ones that were obtained from simulating the model with the estimated parameter vector and the same input signal. The results are shown in Figure 7.4; the simulated outputs match the measured values quite closely. This confirms the accuracy of the estimated parameters.

7.2 Tracking Parameter Changes

The second experiment demonstrates a potential application in monitoring process parameters, for example, for failure detection. After initial convergence, the algorithm was required to track changes in the parameters $p_3$ and $p_4$. Initially, only one of the two outlets of tank 2 was open; the second outlet was opened and closed over the course of the experiment. The algorithm was then to adjust the estimate of the parameter $p_3$ to roughly double (following the opening of the second outlet) or half (following the closing of the second outlet) its previous value. Also, during the experiment, a failure of pump 1 was simulated by reducing the pump’s flow by 50%. Again, the estimate of the corresponding parameter $p_4$ was then to converge to half its previous value (and back to its original value, after the pump is switched back to normal mode).

The settings for this identification experiment were selected as

$$\hat{x}(0) = \begin{bmatrix} 48 & 26 & 37 \end{bmatrix}^T, \quad P_{xx}(0) = I, \quad \hat{p}(0) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \sqrt{0.02} & \sqrt{0.02} & \sqrt{0.02} \end{bmatrix}^T,$$

$$Q(\hat{p}) = \begin{bmatrix} \frac{1}{2} \hat{p}_6^2 & 0 & 0 \\ 0 & \frac{1}{2} \hat{p}_7^2 & 0 \\ 0 & 0 & \frac{1}{2} \hat{p}_8^2 \end{bmatrix}, \quad \frac{d\hat{x}(0)}{dp^T} = 0, \quad \frac{dP_{xx}(0)}{dp^T} = 0, \quad P_{pp}(0) = 10I, \quad R = 0.1I.$$

**Figure 7.4** Model validation: measured (solid) and simulated tank levels (dashed)
The input and output signals for this experiment are shown in Figure 7.5. To achieve the tracking of the parameters $p_3$ and $p_4$, artificial lower bounding of the parameter covariance matrix (Maybeck 1982, p. 27) was employed: after 1000 seconds, the corresponding diagonal entries of the covariance matrix were bounded from below by 0.1. Figure 7.6 shows the estimated parameters and demonstrates that the algorithm is able to track the parameter changes fast and accurately.

Since the model is time-varying in this case, a model validation as in the first experiment is not possible. Instead, the prediction errors are displayed in Figure 7.7. It is evident that the algorithm achieves an accurate prediction and that the increased prediction errors that follow a change in the system are reduced quite fast.
Figure 7.6 Estimated parameters for the second experiment

Figure 7.7 Prediction errors for the second experiment
Overall, the results presented in this chapter demonstrate the applicability of the parameter estimation algorithm in a real-time setting. The algorithm is able to estimate constant parameters and to track time-varying parameters, provided that measures for covariance management (such as artificial lower bounding, forgetting, pseudonoise) are included.
Chapter 8

Discussion and Conclusions

The task of recursively estimating the parameters for nonlinear state-space models can be formulated either as a standard filtering problem with an augmented state vector or as an adaptive filtering problem.

State augmentation leads to a computationally relatively simple solution; however, it requires specifying the covariance matrices of the stochastic disturbances acting upon the system. These covariance matrices are usually unknown; and if they are incorrectly specified, biased estimates may result.

In the adaptive filtering approach, a cost functional is formulated that contains the prediction errors and their covariance matrix. These quantities are obtained from an approximate nonlinear filter, and the parameters of this filter are adjusted so that the cost functional is minimized. Commonly, a gradient-based approach is employed to minimize the cost functional. This requires the gradient of the cost functional with respect to the unknown parameters, and, consequently, the gradients of both the output prediction and the prediction error covariance matrix (although the latter is often neglected in the computation of the gradient). The evaluation of these gradients relies on a sensitivity model. Even for the simplest approximate nonlinear filter, the EKF, the derivation of such a sensitivity model is extremely tedious. To avoid these complications, almost all approaches reported in the literature used the assumption of a constant filter gain. This assumption is, however, only justified for linear systems.

In this dissertation, full sensitivity models have been derived for four common approximate nonlinear filters: the EKF, the FBF, the TSF, and the GSF. To the best of the author’s knowledge, this is the first time that sensitivity models for the latter three filters have been formulated. These sensitivity models are based on a new, elegant, and compact formulation of the additional terms that are present in the higher-order filters. With these sensitivity models, recursive parameter estimation algorithms have been developed for all four filters. The sensitivity model can of course also be used in a nonrecursive, iterative algorithm.
To arrive at this new formulation and to develop the sensitivity models, matrix differential calculus has been rigorously applied. This resulted in a form that offers great advantages in the implementation of the adaptive filtering algorithm; furthermore, the equations containing derivatives stay interpretable since bookkeeping of indices and summations over individual entries of derivative matrices are entirely avoided.

Further aspects of implementing the algorithms have been addressed. A way of algebraically grouping the matrix propagation differential equations into one vector differential equation and eliminating redundant entries has been proposed.

Results of various simulations and a real-time experiment have been included. Both the estimation of parameters that enter the dynamic equations and the parameters that define static input and output nonlinearities have been considered. In all these examples, the adaptive Extended Kalman Filter gave sufficiently accurate parameter estimates, and it was not necessary to use adaptive higher-order filters. The tracking of time-varying parameters has also been considered in a real-time setting; the algorithm was able to track parameter changes fast and accurately.

On a conceptual level, comparing the least-squares cost functional to the negative log-likelihood function has provided additional insight into the nonlinear adaptive filtering algorithm. It turns out that the adaptive filtering algorithm provides all terms that are necessary for the computation of the gradient of the log-likelihood function; thus, an extension to maximum-likelihood estimation is straightforward. Furthermore, the differences between the adaptive EKF and the state-augmented EKF have been analyzed. In contrast to earlier comparisons, the reformulation of the EKF as an adaptive filter is exact.

An actual identification experiment includes many choices for the user, including which algorithm and which cost functional to use. In Figure 8.1, an attempt has been made to break up the identification of a nonlinear state-space model into a step-by-step procedure. The general maxim is to start with simulation studies if this is possible and to try simple procedures first, that is, to test a state-augmented EKF (if simulation studies are possible) or a CGEKF (unless highly accurate state estimates are required). If no online estimation is required, nonrecursive, iterative variants of the adaptive filtering algorithms can of course be used.
FIGURE 8.1 Step-by-step procedure for the identification of a nonlinear state-space model
At the present state, the online applicability of the algorithms presented in this dissertation seems to be limited to systems with relatively few states and parameters unless the sampling frequency is sufficiently low. This particularly holds for the adaptive EKF or the adaptive higher-order filters, which will have a large number of states even for relatively small systems. Table 8.1 showcases this by giving the total number of states for the different filters. Since the numerical integration of the propagation equations is the major computational requirement of the algorithms, the number of continuous states is also included. As an example, the last column gives the actual number of states that result if the algorithms are applied to the three-tank system considered in the previous chapter.

Clearly, the adaptive Extended Kalman Filter is the most complex algorithm, but even the adaptive CGEKF suffers from a large number of states. This high dimension of the adaptive nonlinear filters is, however, the price that has to be paid for the joint estimation of states and parameters.

<table>
<thead>
<tr>
<th>Filter</th>
<th>Number of states</th>
<th>Example (three-tank system)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>$n = m = 3, \ s = 5$</td>
</tr>
<tr>
<td>state-augmented EKF$^1$</td>
<td>$\frac{(n + s)(n + s + 3)}{2}$</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>$n + \frac{(n + s)(n + s + 1)}{2}$</td>
<td>39</td>
</tr>
<tr>
<td>adaptive CGEKF</td>
<td>$\frac{2n + (s + nm)(2n + s + nm + 3)}{2}$</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td>$n(s + nm + 1)$</td>
<td>45</td>
</tr>
<tr>
<td>adaptive EKF, FBF, TSF, GSF$^2$</td>
<td>$\frac{n(n + 3) + (n + s)(n(n + 4) + s + 3)}{2}$</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>$\frac{n(n + 3)(n + s + 1)^2}{2}$</td>
<td>81</td>
</tr>
</tbody>
</table>

1) These figures are based on the general case of parameter pseudonoise; without pseudonoise, the elements of the parameter covariance matrix would be discrete states.

2) These figures are based on a diagonal parameterization of the process noise covariance matrix; the filters would have even more states if a full parameterization of the process noise covariance were used or if the measurement noise covariance were parameterized as well.
A potential future extension of this work is the development of adaptive reduced-order state-estimators (Gelb 1974, pp. 327-32). These would be applicable to systems where certain states or linear combinations of states can be measured with (almost) no measurement noise. A significant reduction in computational effort could be expected from such filters. Another important aspect would be the possible use of approximately discretized nonlinear state-space models. The propagation differential equations of the adaptive filter could then be reduced to difference equations, which would also lower the computational burden. At least for small sampling times, it should be possible to recover approximately the parameters of the continuous-time system. An adaptive Extended Kalman Filter based on an approximately discretized model was presented in the context of adaptive noise estimation by Bohn and Unbehauen (1999a, b), but the problem of parameter estimation has not been addressed yet.

Finally, it needs to be stressed that, with the exceptions of methods that use linearly parameterized models, algorithms for nonlinear system identification are essentially trial-and-error methods that often require extensive tuning (this is why the option “Retune; . . . or give up” has been included in Figure 8.1). The statement of Friedland (1997) seems to hold for a variety of nonlinear identification methods:

Needless to say, there are many other methods for estimating the parameters of linear and nonlinear systems. In a specific application, the method presented here may or may not be better than some other method. But the more methods that are available, the greater is the likelihood of finding the most appropriate method for the application.

Thus, the development of nonlinear system identification techniques is and remains an important, topical, and ongoing task.
Zusammenfassung


Einem recht einfachen Algorithmus (zustandserweitertes EKF) bei der Methode der Zustandserweiterung steht die notwendige Einstellung der Kovarianzmatrizen des Prozess- und des Messrauschens gegenüber. Eine schlechte Einstellung dieser in der Regel unbekannten Matrizen kann zu systematischen Schätzfehlern in den Parametern (biased estimates) führen.

Bei der adaptiven Filterung wird ein Gütefunktional formuliert, in welches die Vorhersagefehler und deren Kovarianzmatrizen eingehen. Diese werden von einem Filter erzeugt, welches dann so eingestellt wird, daß das Gütefunktional einen minimalen Wert annimmt. Dabei können auch die Kovarianzmatrizen des Prozess- und des Messrauschens eingestellt werden. Für nichtlineare Systeme muss ein nichtlineares Filter verwendet werden, und die wesentliche Schwierigkeit besteht in der Aufstellung eines Empfindlichkeitsmodells für dieses Filter. In der Regel wird dies durch die Verwendung eines adaptiven Filters mit einer konstant angenommenen Verstärkungsmatrix vermieden (Innovationsmodell); dies ist jedoch nur für lineare Systeme zulässig.

In der vorliegenden Dissertation wurde ein allgemeiner adaptiver Filteralgorithmus für nichtlineare, kontinuierlich-diskrete Zustandsraummodelle vorgeschlagen, in dem neben dem EKF auch gängige Filter höherer Ordnung, wie das FBF, TSF oder GSF eingesetzt werden können. Über die Anwendung der Matrizendifferentialrechnung wurden die Empfindlichkeitsmodelle für diese Filter in kompakter und unmittelbar implementierbarer Form aufgestellt. Hierzu wurde zunächst im dritten Kapitel eine neue Formulierung für die im FBF, TSF und GSF auftretenden zusätzlichen Terme entwickelt. Obwohl es sich hierbei streng genommen nur um eine alternative Herleitung bereits bekannter Ausdrücke handelt, ist die neue Formulierung sehr attraktiv, da sie die Aufstellung der Empfindlichkeitsmodelle wesentlich vereinfacht.

Die Empfindlichkeitsmodelle und die Bestandteile des allgemeinen Filteralgorithmus wurden im vierten Kapitel hergeleitet. Dem Algorithmus liegen ein Kleinste-Quadrate-Gütefunktional und ein mit üblichen Näherungen berechneter Gradient zugrunde. Es wurde

Das fünfte Kapitel beinhaltet die Umwandlung der Vorhersagegleichungen in eine leicht implementierbare Form: Die Matrixdifferentialgleichungen wurden vektorisiert und redundante Zustände entfernt.


Insgesamt ist festzustellen, daß (nichtlineare) Zustandsraummodelle in der Systemidentifikation unterrepräsentiert sind, obwohl sie eine attraktive Beschreibungsform darstellen
Appendix A

Matrix Differential Calculus

In this appendix, rules of matrix differential calculus are listed. The definition of a matrix derivative proposed by Vetter (1973) and Weinmann (1991) is introduced (Definition A.1). Rules for this derivative are given (Rules A.1 – A.4). In Rule A.5, the Taylor expansion of matrix-valued functions using this derivative notation is presented. The derivatives of special matrix functions are given in Rules A.6 – A.8.

Definition A.1 (Matrix Derivative)
The matrix derivative is defined as

\[
\frac{dA(M)}{dM} := \frac{d}{dM} \otimes A(M) = \begin{bmatrix}
\frac{d}{d} & \cdots & \frac{d}{d} \\
\frac{dM_{11}}{d} & \cdots & \frac{dM_{1s}}{d} \\
\vdots & \ddots & \vdots \\
\frac{dM_{r1}}{d} & \cdots & \frac{dM_{rs}}{d}
\end{bmatrix} \otimes A(M) = \\
\begin{bmatrix}
\frac{dA(M)}{dM_{j1}} & \cdots & \frac{dA(M)}{dM_{js}} \\
\vdots & \ddots & \vdots \\
\frac{dA(M)}{dM_{r1}} & \cdots & \frac{dA(M)}{dM_{rs}}
\end{bmatrix}.
\]

With this definition, the standard rules of differential calculus can be generalized for matrices as follows.

Rule A.1 (Matrix Product Rule)
The derivative of a matrix product is given as

\[
\frac{dAB}{dM} = \frac{dA}{dM} (I_r \otimes B) + (I_r \otimes A) \frac{dB}{dM}.
\]

In this framework, only the multiplication of terms with “matching” dimensions is allowed. Therefore, the product of a scalar and a matrix must be treated as a Kronecker product, that is, \( \alpha B \) must be replaced by \( \alpha \otimes B \) and the Kronecker product rule (A.3) must be used.
Rule A.2 (Matrix Chain Rule)

The derivative of a composite matrix function \( A(B(M)) \) is given as

\[
\frac{d A^{(\text{com})}(B(M))}{dM^{(r\times c)}} = \left( \begin{array}{c} \frac{d A}{d \text{row } B} \end{array} \right) \left( \begin{array}{c} \frac{d \text{col } (B^T)}{dM} \end{array} \right) \right) \otimes I_m
\]

\[
= \left( \frac{d \text{col } B}{dM} \right)^T \otimes I_n \left( \begin{array}{c} \frac{d A}{d \text{col } B} \end{array} \right)
\]

In the following two rules, the derivatives of the Kronecker operations are presented.

Rule A.3 (Kronecker Product Rule)

The Kronecker product rule is given as

\[
\frac{d A^{(\text{com})} \otimes B^{(k\times d)}}{dM^{(r\times c)}} = \frac{d A}{dM} \otimes B + \left( \begin{array}{c} \frac{d A}{d \text{vec } B} \end{array} \right) \left( \begin{array}{c} \frac{d B}{dM} \otimes A \end{array} \right) \left( \begin{array}{c} \frac{d A}{d \text{vec } B} \end{array} \right)
\]

Rule A.4 (Derivative of the Kronecker Sum)

The derivative of the Kronecker sum \( B(M) \oplus C(M) \) is

\[
\frac{d}{dM^{(r\times c)}} (B^{(\text{com})} \oplus C^{(\text{com})}) = \left( \begin{array}{c} \frac{d B}{dM} \otimes I_m + C \end{array} \right) \left( \begin{array}{c} \frac{d A}{d \text{vec } B} \end{array} \right)
\]

The Taylor expansion of a matrix function with regard to a vector argument is treated in the next rule.

Rule A.5 (Taylor Expansion)

The Taylor expansion of a matrix function \( A \) depending on the vector-valued argument \( p \) is given as

\[
A^{(\text{com})}(p_0 + \Delta p) = A(p_0) + \sum_{i=1}^{\infty} \frac{d^i A(p_0)}{d p_0} \left( \Delta p \right)^\otimes i \otimes I_m,
\]

where the superscript \( \otimes, i \) stands for the \( i \)th Kronecker power, that is,

\[
(\Delta p)^\otimes i = \Delta p \otimes \Delta p \otimes \ldots \otimes \Delta p \ (i \text{ multiplications}).
\]

Writing this out for terms up to second order yields

\[
A^{(\text{com})}(p_0 + \Delta p) = A(p_0) + \frac{d A(p_0)}{d p_0} (\Delta p \otimes I_m) + \frac{d^2 A(p_0)}{d p_0^2} (\Delta p \otimes \Delta p) \otimes I_m + R(\Delta p),
\]

where the remainder term satisfies...
\[ \lim_{\Delta p \to 0} \frac{R(\Delta p)}{\|\Delta p\|^2} = 0. \]

For a vector function \( f \) this reduces to

\[ f(p_0 + \Delta p) = f(p_0) + \frac{df(p_0)}{dp_0^T} \Delta p + \frac{d^2 f(p_0)}{dp_0^T dp_0^T} (\Delta p \otimes \Delta p) + r(\Delta p). \]

The derivatives of three special matrix functions are presented in the following three rules.

**Rule A.6 (Derivative of the Identity Function)**

Some derivatives of a matrix with respect to itself follow as

\[ \frac{dM^{(rs)}}{dM^T} = U_{rs}, \quad \frac{dM^{(rs)}}{dM} = U_{rs}, \quad \frac{d\col{M^{(rs)}}}{(\col M)^T} = I_{rs}, \]

with the self-derivative matrix

\[ U_{kl}^{(klsd)} = \sum_{i=1}^k \sum_{j=1}^l E^{(kds)}_{ij} \otimes E^{(kds)}_{ij}. \]

The second derivative of this rule is one of the major points of Magnus and Neudecker’s (1999) criticism of the notation used here. They argue that the derivative of a matrix with respect to itself should be the identity matrix. Therefore, they propose always using derivatives that look like the third derivative of this rule. This, however, produces other inconsistencies, for example, the derivative of a matrix with respect to a scalar would be a vector.

**Rule A.7 (Derivative of the Matrix Inverse)**

The derivative of the inverse of a matrix function \( A(M) \) is given as

\[ \frac{dA^{-1}}{dM^{(rs)}} = - (I_r \otimes A^{-1}) \frac{dA}{dM} (I_r \otimes A^{-1}). \]

**Rule A.8 (Derivative of the Logarithm of the Determinant)**

The derivative of the logarithm of the determinant of a matrix \( A \) with respect to a vector \( p \) is given by

\[ \frac{d \ln \det A(p)}{dp^{(rs)}} = (I_r \otimes \text{row}(A^{-1})) \col \frac{dA(p)}{dp^T}. \]
Appendix B

Auxiliary Derivations for Chapter 3

In this appendix, some auxiliary derivations for Chapter 3 are given. The material presented here is also reported in Bohn and Unbehauen (2000).

B.1 Derivation of Equation (3.21)

From the expression for \( h - E h \) given in Equation (3.20), the quadratic form \((h - E h)(h - E h)^T\) follows as

\[
(h - E h)(h - E h)^T = \frac{\partial h}{\partial x}^T (x - \hat{x})(x - \hat{x})^T \frac{\partial h}{\partial x}^T + \frac{1}{2} \frac{\partial h}{\partial x^T} (x - \hat{x}) [(x - \hat{x}) \otimes (x - \hat{x})] \left[ \frac{\partial^2 h}{\partial x^T \partial x^T} \right]^T (x - \hat{x}) \frac{\partial h}{\partial x}
\]

\[
- \frac{1}{2} \frac{\partial h}{\partial x^T} (x - \hat{x}) (\text{col } P_{\hat{x}})^T \left[ \frac{\partial^2 h}{\partial x^T \partial x^T} \right]^T \frac{\partial h}{\partial x}
\]

\[
+ \frac{1}{2} \frac{\partial^2 h}{\partial x^T \partial x^T} [(x - \hat{x}) \otimes (x - \hat{x})] [x - \hat{x}]^T \frac{\partial h}{\partial x}^T
\]

\[
+ \frac{1}{4} \frac{\partial^2 h}{\partial x^T \partial x^T} [(x - \hat{x}) \otimes (x - \hat{x})] [(x - \hat{x}) \otimes (x - \hat{x})] \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T \left( \frac{\partial^2 h}{\partial x^T \partial x^T} \right)^T
\]

\[
(B.1)
\]

Then, Equation (3.21) follows from taking the expectation, using the fact that all expressions involving first and third moments are zero (the second, third, fourth, and seventh term on the right-hand side), and noting that, with Equation (3.16), the last two terms on the right-hand side cancel.
B.2 Derivation of Equation (3.25)

The moment-generating function is given in Equation (3.23) as

\[ \Phi(\mu) = \exp\left(-\frac{1}{2} \mu^T P_{xx} \mu\right). \quad (3.23) \]

The first derivative of this expression (an \( n \) by 1 vector) follows from the chain rule as

\[ \frac{\partial \Phi}{\partial \mu} = -\Phi P_{xx} \mu. \quad (B.2) \]

The second derivative can be easily computed by writing the product with the scalar function \( \Phi \) as a Kronecker product and applying the Kronecker product rule to evaluate

\[ \frac{\partial^2 \Phi}{\partial \mu^T \partial \mu} = -\frac{\partial}{\partial \mu^T} (\Phi \otimes P_{xx} \mu), \quad (B.3) \]

which results in

\[ \frac{\partial^2 \Phi}{\partial \mu^T \partial \mu} = -\left( \frac{\partial \Phi}{\partial \mu^T} \otimes (P_{xx} \mu) + \frac{\partial P_{xx} \mu}{\partial \mu^T} \otimes \Phi \right) \]
\[ = (\Phi \mu^T P_{xx}) \otimes (P_{xx} \mu) - P_{xx} \otimes \Phi \]
\[ = \Phi \left[ (\mu^T P_{xx}) \otimes (P_{xx} \mu) - P_{xx} \right], \quad (B.4) \]

an \( n \) by \( n \) matrix. Taking the derivative of this expression, again writing the product with the scalar \( \Phi \) as a Kronecker product, gives

\[ \frac{\partial^3 \Phi}{\partial \mu^T \partial \mu^T \partial \mu} = \Phi \left[ (P_{xx} \mu) \otimes P_{xx} + P_{xx} \otimes (P_{xx} \mu) + (\text{col } P_{xx}) \otimes (\mu^T P_{xx}) \right. \]
\[ \left. - (P_{xx} \mu) \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu) \right], \quad (B.5) \]

Finally, taking the derivative once more gives

\[ \frac{\partial^4 \Phi}{\partial \mu^T \partial \mu^T \partial \mu^T \partial \mu} = \frac{\partial \Phi}{\partial \mu^T} \otimes \left[ (P_{xx} \mu) \otimes P_{xx} + P_{xx} \otimes (P_{xx} \mu) \right. \]
\[ + (\text{col } P_{xx}) \otimes (\mu^T P_{xx}) - (P_{xx} \mu) \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu) \]
\[ \left. + \left\{ \frac{\partial}{\partial \mu^T} \left[ (P_{xx} \mu) \otimes P_{xx} + P_{xx} \otimes (P_{xx} \mu) + (\text{col } P_{xx}) \otimes (\mu^T P_{xx}) \right. \right. \right. \]
\[ \left. \left. \left. - (P_{xx} \mu) \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu) \right] \otimes \Phi, \right) \quad (B.6) \]

an \( n^2 \) by \( n^2 \) matrix. Substituting \( \mu = 0 \) causes the first term on the right-hand side of this equation to disappear. The individual derivatives arising in the second term can be computed as
\[
\frac{\partial (P_{xx} \mu) \otimes P_{xx}}{\partial \mu^T} = P_{xx} \otimes P_{xx}, \quad (B.7)
\]

\[
\frac{\partial P_{xx} \otimes (P_{xx} \mu)}{\partial \mu^T} = U_{nn} (P_{xx} \otimes P_{xx}), \quad (B.8)
\]

\[
\frac{\partial (\text{col} P_{xx}) \otimes (\mu^T P_{xx})}{\partial \mu^T} = (\text{col} P_{xx}) \otimes (\text{col} P_{xx})^T, \quad (B.9)
\]

and

\[
\frac{\partial (P_{xx} \mu) \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu)}{\partial \mu^T} = P_{xx} \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu)
\]

\[+ U_{nn} [(\text{col} P_{xx})^T \otimes (P_{xx} \mu) \otimes (P_{xx} \mu)] \]

\[+ P_{xx} \otimes (\mu^T P_{xx}) \otimes (P_{xx} \mu)]. \quad (B.10)
\]

Substituting \( \mu = 0 \), only the terms in Equations (B.7), (B.8), and (B.9) remain. Then, noting that \( \phi(0) = 1 \), it follows that

\[
\frac{\partial^4 \phi}{\partial \mu^T \partial \mu \partial \mu^T \partial \mu} \bigg|_{\mu=0} = (I_n + U_{nn}) (P_{xx} \otimes P_{xx}) + (\text{col} P_{xx}) \otimes (\text{col} P_{xx})^T, \quad (B.11)
\]

which, inserted into Equation (3.24), yields the desired result, Equation (3.25).
Appendix C

Reformulation of the State-Augmented Extended Kalman Filter

In this appendix, the Extended Kalman Filter is rewritten as a sensitivity-based algorithm. The strategy here is that the equations are reformulated such that the term $P_{xp}$ is replaced by a new term $S_x$. This new term is chosen such that its propagation and update equation closely resemble the corresponding equations for the sensitivity. Thus, only equations that contain $P_{xp}$ have to be rewritten. The starting point for the derivation is the state-augmented Extended Kalman Filter given by the following equations:

$$\dot{x} = f(\tilde{x}, u, \tilde{p}(t_k))$$  \hspace{1cm} (C.1)

$$\dot{P}_{xx} = \frac{\partial f}{\partial x^T} P_{xx} + P_{xx} \frac{\partial f^T}{\partial x} + \frac{\partial f}{\partial p^T} P_{xp} + P_{xp} \frac{\partial f^T}{\partial p} + Q$$ \hspace{1cm} (C.2)

$$\dot{P}_{xp} = \frac{\partial f}{\partial x^T} P_{xp} + \frac{\partial f}{\partial p^T} P_{pp}$$ \hspace{1cm} (C.3)

$$\dot{P}_{pp} = 0$$ \hspace{1cm} (C.4)

$$\dot{y}(t_{k+1}) = h(\tilde{x}(t_{k+1}), u(t_{k+1}), \hat{p}(t_k))$$ \hspace{1cm} (C.5)

$$e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1})$$ \hspace{1cm} (C.6)

$$A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} P_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} + \frac{\partial h}{\partial p^T} P_{xp}(t_{k+1}) \frac{\partial h^T}{\partial p} + \frac{\partial h}{\partial p^T} P_{pp}(t_{k+1}) \frac{\partial h^T}{\partial p}$$ \hspace{1cm} (C.7)

$$K(t_{k+1}) = P_{xx}\left(t_{k+1}\right) \frac{\partial h^T}{\partial x} + P_{xp}\left(t_{k+1}\right) \frac{\partial h^T}{\partial p} \right) A^{-1}(t_{k+1})$$ \hspace{1cm} (C.8)

$$L(t_{k+1}) = P_{xp}\left(t_{k}\right) \frac{\partial h^T}{\partial x} + P_{pp}\left(t_{k+1}\right) \frac{\partial h^T}{\partial p} \right) A^{-1}(t_{k+1})$$ \hspace{1cm} (C.9)
\( \hat{x}(t_{k+1}) = \hat{x}(t_{k+1}) + K(t_{k+1})e(t_{k+1}) \)  \( (C.10) \)

\( \hat{p}(t_{k+1}) = \hat{p}(t_k) + L(t_{k+1})e(t_{k+1}) \)  \( (C.11) \)

\[ P_{xx}(t_{k+1}) = P_{xx}(t_{k+1}) - K(t_{k+1})A(t_{k+1})K^T(t_{k+1}) \]  \( (C.12) \)

\[ P_{xp}(t_{k+1}) = P_{xp}(t_{k+1}) - K(t_{k+1})A(t_{k+1})L^T(t_{k+1}) \]  \( (C.13) \)

\[ P_{pp}(t_{k+1}) = P_{pp}(t_k) - L(t_{k+1})A(t_{k+1})L^T(t_{k+1}) \]  \( (C.14) \)

An assumption here is that no pseudonoise is used, that is, the derivative of the error covariance matrix for the parameter estimates in Equation (C.4) is zero. Thus, \( P_{pp} \) does not change between \( t_k \) and \( t_{k+1} \), and that is why \( P_{pp}(t_k) \) and \( P_{pp}(t_{k+1}) \) are used instead of \( P_{pp}(t_k) \) and \( P_{pp}(t_{k+1}) \), respectively.

Next, a new term is defined as

\[ S_x = P_{xp}P_{pp}^{-1}. \]  \( (C.15) \)

The propagation equation for this new term can be derived from Equation (C.3) by post-multiplication with \( P_{pp}^{-1} \) and noting that the time derivative of \( P_{pp} \) is zero as

\[ \dot{S}_x = \frac{\partial f}{\partial x} S_x + \frac{\partial f}{\partial p} T. \]  \( (C.16) \)

Further, a modified error covariance matrix is defined as

\[ \overline{P}_{xx} = P_{xx} - S_x P_{pp} S_x^T. \]  \( (C.17) \)

Taking the derivative, substituting for \( \dot{P}_{xx} \) from Equation (C.2), and noting that the terms containing \( P_{xp} \) cancel yields the propagation equation

\[ \dot{\overline{P}}_{xx} = \frac{\partial f}{\partial x} \overline{P}_{xx} + \overline{P}_{xx} \frac{\partial f^T}{\partial x} + Q. \]  \( (C.18) \)

Adding and subtracting the term \( \frac{\partial h}{\partial x} S_x(t_{k+1})P_{pp}(t_k)S_x^T(t_{k+1}) \frac{\partial h^T}{\partial x} \) on the left-hand side of Equation (C.7) and using the above definitions, the expression for the approximate prediction error covariance matrix can be rewritten as

\[ A(t_{k+1}) = R + \frac{\partial h}{\partial x} \frac{\partial h^T}{\partial x} + \left( \frac{\partial h}{\partial x} S_x(t_{k+1}) + \frac{\partial h}{\partial p} \right) P_{pp}(t_k) \left( \frac{\partial h}{\partial x} S_x(t_{k+1}) + \frac{\partial h}{\partial p} \right)^T. \]  \( (C.19) \)

To simplify the notation, the term
APPENDIX C: REFORMULATION OF THE STATE-AUGMENTED EKF

\( S_y(t_{k+1}) := \frac{\partial h}{\partial x^T} S_x(t_{k+1}^-) + \frac{\partial h}{\partial p^T} \) \tag{C.20}

is defined; and Equation (C.19) is rewritten as

\[ A(t_{k+1}) = R + \frac{\partial h}{\partial x^T} \bar{P}_{xx}(t_{k+1}) \frac{\partial h^T}{\partial x} + S_y(t_{k+1}) P_{pp}(t_k) S_y^T(t_{k+1}) \] \tag{C.21}

Using the definition of Equation (C.15), the parameter adaptation gain can be rewritten as

\[ L(t_{k+1}) = P_{pp}(t_k) \frac{\partial h}{\partial x^T} S_x(t_{k+1}^-) + \frac{\partial h}{\partial p^T}^T A^{-1}(t_{k+1}) \] \tag{C.22}

or, with Equation (C.20), as

\[ L(t_{k+1}) = P_{pp}(t_k) S_y^T(t_{k+1}) A^{-1}(t_{k+1}) \] \tag{C.23}

The gain for the state estimate can be expressed as

\[ K(t_{k+1}) = \bar{P}_{xx}(t_{k+1}^-) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1}) + S_x(t_{k+1}^-) L(t_{k+1}) \] \tag{C.24}

If an expression for the first term on the right-hand side of Equation (C.24) is defined as

\[ \bar{K}(t_{k+1}) := \bar{P}_{xx}(t_{k+1}^-) \frac{\partial h^T}{\partial x} A^{-1}(t_{k+1}) \] \tag{C.25}

and the parameter correction \( \Delta \hat{p}(t_{k+1}) \) is introduced as

\[ \hat{p}(t_{k+1}) = \hat{p}(t_k) + \Delta \hat{p}(t_{k+1}) \], \tag{C.26}

with

\[ \Delta \hat{p}(t_{k+1}) = L(t_{k+1}) e(t_{k+1}) \], \tag{C.27}

the state update can be written as

\[ \hat{x}(t_{k+1}^+) = \hat{x}(t_{k+1}^-) + \bar{K}(t_{k+1}) e(t_{k+1}) + S_x(t_{k+1}^-) \Delta \hat{p}(t_{k+1}) \] \tag{C.28}

With these substitutions, the term \( P_{xp} \) is removed from the propagation equations, the expression for the prediction error covariance matrix, and from the expressions for the filter gains. However, a new set of update equations still must be found that does not depend on \( P_{xp} \). First, an update equation for \( S_y \) is developed. Such an update equation can be obtained from the fact that

\[ S_x(t_{k+1}^-) P_{pp}(t_{k+1}) = S_x(t_{k+1}^-) P_{pp}(t_k) - S_x(t_{k+1}^+) L(t_{k+1}) A(t_{k+1}) L^T(t_{k+1}) \] \tag{C.29}

and

\[ P_{xp}(t_{k+1}) = S_x(t_{k+1}^-) P_{pp}(t_k) - K(t_{k+1}) A(t_{k+1}) L^T(t_{k+1}) \] \tag{C.30}
must be equal, which follows from the definition in Equation (C.15).

Setting these equations equal and solving for the difference $S_x(t_{k+1}) - S_x(t_{k+1})$ yields the update equation

$$S_x(t_{k+1}) = S_x(t_{k+1}) - \overline{K}(t_{k+1})S_y(t_{k+1})P_{pp}(t_k)P_{pp}^{-1}(t_{k+1}).$$  \hspace{1cm} (C.31)

Finally, by a series of straightforward manipulations, an update equation for $P_{xx}$ can be found as

$$P_{xx}(t_{k+1}) = P_{xx}(t_{k+1}) - \overline{K}(t_{k+1})A(t_{k+1})\overline{K}^T(t_{k+1})$$

$$- \overline{K}(t_{k+1})A(t_{k+1})L^T(t_{k+1})P_{pp}^{-1}(t_{k+1})L(t_{k+1})A(t_{k+1})\overline{K}^T(t_{k+1}).$$  \hspace{1cm} (C.32)

Thus, an alternative representation of the state-augmented Extended Kalman Filter is given by the following equations:

$$\dot{x} = f(\hat{x}, u, \hat{p}(t_k))$$  \hspace{1cm} (C.33)

$$\dot{P}_{xx} = \frac{\partial f^T}{\partial x}P_{xx} + P_{xx}\frac{\partial f^T}{\partial x} + Q$$  \hspace{1cm} (C.34)

$$\dot{S}_x = \frac{\partial f^T}{\partial x}S_x + \frac{\partial f^T}{\partial p}$$  \hspace{1cm} (C.35)

$$\dot{y}(t_{k+1}) = h(\hat{x}(t_{k+1}), u(t_{k+1}), \hat{p}(t_{k+1}))$$  \hspace{1cm} (C.36)

$$e(t_{k+1}) = y(t_{k+1}) - \hat{y}(t_{k+1})$$  \hspace{1cm} (C.37)

$$S_y(t_{k+1}) := \frac{\partial h^T}{\partial x}S_x(t_{k+1}) + \frac{\partial h^T}{\partial p}$$  \hspace{1cm} (C.38)

$$A(t_{k+1}) = R + \frac{\partial h^T}{\partial x}P_{xx}(t_{k+1})\frac{\partial h^T}{\partial x} + S_y(t_{k+1})P_{pp}(t_k)S_y^T(t_{k+1})$$  \hspace{1cm} (C.39)

$$\overline{K}(t_{k+1}) := \overline{P}_{xx}(t_{k+1})\frac{\partial h^T}{\partial x}A^{-1}(t_{k+1})$$  \hspace{1cm} (C.40)

$$L(t_{k+1}) = P_{pp}(t_k)\left(\frac{\partial h^T}{\partial x}S_x(t_{k+1}) + \frac{\partial h^T}{\partial p}\right)^TA^{-1}(t_{k+1})$$  \hspace{1cm} (C.41)

$$\Delta \hat{p}(t_{k+1}) = L(t_{k+1})e(t_{k+1})$$  \hspace{1cm} (C.42)

$$\hat{p}(t_{k+1}) = \hat{p}(t_k) + \Delta \hat{p}(t_{k+1})$$  \hspace{1cm} (C.43)

$$\dot{x}(t_{k+1}) = \hat{x}(t_{k+1}) + \overline{K}(t_{k+1})e(t_{k+1}) + S_x(t_{k+1})\Delta \hat{p}(t_{k+1})$$  \hspace{1cm} (C.44)
\[ P_{pp}(t_{k+1}) = P_{pp}(t_k) - L(t_{k+1})A(t_{k+1})L^T(t_{k+1}) \] (C.45)

\[
P_{xx}(t_{k+1}^-) = \bar{P}_{xx}(t_{k+1}^-) - \bar{K}(t_{k+1}^-)A(t_{k+1})\bar{K}^T(t_{k+1}^-) \\
- \bar{K}(t_{k+1}^-)A(t_{k+1})L^T(t_{k+1}^-)P_{pp}^{-1}(t_{k+1}^-)L(t_{k+1}^-)A(t_{k+1})\bar{K}^T(t_{k+1}^-) \] (C.46)

\[ S_x(t_{k+1}^-) = S_x(t_{k+1}^-) - \bar{K}(t_{k+1}^-)S_y(t_{k+1}^-)P_{pp}^{-1}(t_k)P_{pp}^{-1}(t_{k+1}^-) \] (C.47)

It is discussed in Chapter 4 how this algorithm can be interpreted as a sensitivity-based approach.
Appendix D

Description of the Three-Tank System

D.1 The Three-Tank System

The three-tank system that is used for the real-time experiment is the amira DTS 2000 system. The system is schematically shown in Figure D.1. It consists of three cylindrical tanks that are connected through valves. Tank 1 and tank 3 have one outlet each, and tank 2 has two outlets. Pump 1 and pump 2 feed water from the bottom basin into tank 1 and tank 2, respectively. The control inputs are the pump voltages $u_1(t)$ and $u_2(t)$, and the outputs are the water levels in each tank; they are measured as voltages from differential pressure sensors. The pumps and the differential pressure sensors are connected to an electronic control unit (part of the amira DTS 2000 system) that converts external input voltages between $-10$ and $+10$ volts to the pump command voltages and provides the sensor signals in the range of $-10$ and $+10$ volts. For the real-time experiments, this control unit is connected to a PC, and the experiments are carried out through a Simulink block diagram with Vetsch Consulting's InteractiveRealtime Toolbox (Vetsch Consulting 1997). In Simulink, the input signals are normalized to the range of $0 ... 1$, and the sensor signals are converted to the range of $0 ... 60$ (centimeters).

![Figure D.1 Three-tank system](image-url)
In the operating setup used here, the two connecting valves are open, the outlets of tank 1 and tank 3 are closed, and both outlets of tank 2 are open (as shown in Figure D.1). From this setup, it follows that the level in tank 1 will always be higher than (or equal to) the level in tank 3, and the level in tank 3 will always be higher than (or equal to) the level in tank 2, that is, \( x_1(t) \geq x_3(t) \geq x_2(t) \), if the initial conditions also satisfy this relation.

A model for this system is developed as follows. According to Toricelli’s law, the flows between the tanks are proportional to the square root of the difference in the water levels, that is,
\[
q_{13}(t) = p_1 \sqrt{x_1 - x_3}
\]
and
\[
q_{32}(t) = p_2 \sqrt{x_3 - x_2}.
\]
The combined flow through both outlets of tank 2 is given as
\[
q_{20}(t) = p_3 \sqrt{x_2}.
\]

Since the three tanks are identical in shape, it is convenient to use the water level (in centimeters) as a measure of the volume and to express the “flows” in centimeters per second. Accordingly, the units of the valve constants \( p_1 \), \( p_2 \), and \( p_3 \) are centimeters per second (flow) per square root of centimeters (level difference).

The pumps are modeled as pure proportional gain elements; thus, the flows into tank 1 and tank 2 are given as
\[
q_1(t) = p_4 u_1(t)
\]
and
\[
q_2(t) = p_5 u_1(t).
\]
Since \( u_1(t) \) and \( u_2(t) \) are dimensionless, the pump constants \( p_4 \) and \( p_5 \) are expressed in centimeters per second (flow). The continuity equation for the volumes (water levels) in the three tanks leads to the state-space model given by
\[
\dot{x}_1(t) = -q_{13}(t) + q_1(t),
\]
\[
\dot{x}_2(t) = q_{32}(t) + q_2(t) - q_{20}(t),
\]
\[
\dot{x}_3(t) = q_{13}(t) - q_{32}(t),
\]
and
\[
y_i = x_i, \ i = 1, 2, 3;
\]
D.2 Determining the Parameters

The parameters of the system can be experimentally determined as follows. To find the pump characteristics, a staircase signal that starts at zero and increases by 0.1 every 12 seconds up to its final value of one is applied, and the level is measured every 12 seconds. This way, the flow for a given, constant input signal is obtained as the level change over each sampling period (twelve seconds) divided by twelve seconds. The results for both pumps are shown in Figure D.2, where the circles represent the measurements, and the solid line is the straight-line fit minimizing the sum of the squared errors. This leads to slopes of $p_4 = p_5 \approx 0.8223$ (and negligible biases). These are only coarse estimates, though, since only 10 points are used for the straight-line fit: if two separate fits are performed for the low range (0.1 ... 0.5) and the high range (0.6 ... 1) of the input signal, different slopes of $p_4 \approx 0.8170$ (low range) and $p_4 \approx 0.7905$ (high range) and of $p_5 \approx 0.8655$ (low range) and $p_5 \approx 0.7261$ (high range) are obtained. Although the difference is hardly visible, these low-range and high-range fits are also shown as broken lines in Figure D.2.

Judging from the overall fit, both pumps show good linearity and nearly identical behavior. With a base diameter of 14 centimeters, the maximum flow of approximately 0.8 centimeters per second corresponds to a volume flow of approximately 7.4 liters per second, which means that the pumps are slightly more powerful than specified (7.0 liters per second).
The parameters of the valves can be determined from the transient behavior of the system. To determine the parameter \( p_3 \) that governs the combined flow through both outlets of tank 2, the decrease of the water level in this tank from an initial value is observed with both outlets open. The differential equation that governs this behavior is

\[
\dot{x}_3(t) = -p_3 \sqrt{x_3(t)},
\]

which has the solution

\[
x_3(t) = \left( \sqrt{x_3(t_0)} - \frac{1}{2} p_3 (t - t_0) \right)^2,
\]

for \( x_3(t) > 0 \). An estimate of the parameter \( p_3 \) can then be obtained from the least-squares fit of this solution to the measurements. The measurements and the best least-squares fit are shown in Figure D.3. The least-squares fit gives the estimate \( p_3 = 0.1922 \).

To determine the parameters \( p_1 \) and \( p_3 \), the transients that follow an initial level difference between tank 1 and 3 and tank 2 and 3, respectively, are used. The level difference satisfies the differential equation

\[
\dot{e}_{13/32}(t) = -2 \frac{p_1}{2} \sqrt{e_{13/32}(t)},
\]

with \( e_{13} = x_1 - x_3 \) and \( e_{32} = x_3 - x_2 \). This differential equation solves as

\[
e_{13/32}(t) = \left( \sqrt{e_{13/32}(t_0)} - \frac{1}{2} \frac{p_1}{2} \left( t - t_0 \right) \right)^2.
\]

The measurements and the least-squares fits are shown in Figure D.4. From the least-squares fits, the parameters \( p_1 \) and \( p_2 \) follow as \( p_1 = 0.0680 \) and \( p_2 = 0.0665 \). Table D.1 lists the parameters of the model for the three-tank system and their values.
Figure D.3  Water level over time for tank 2 with both outlets open.

Figure D.4  The transients following an initial level difference between tank 1 and 3 (left), and tank 2 and 3 (right).

Table D.1  Parameters of the model for the three-tank system and their values

<table>
<thead>
<tr>
<th>Connection</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tank 1 and 3</td>
<td>$p_1$</td>
<td>$0.0680 , \frac{cm}{s \cdot \sqrt{cm}}$</td>
</tr>
<tr>
<td>tank 2 and 3</td>
<td>$p_2$</td>
<td>$0.0665 , \frac{cm}{s \cdot \sqrt{cm}}$</td>
</tr>
<tr>
<td>Both outlets of tank 3</td>
<td>$p_3$</td>
<td>$0.1922 , \frac{cm}{s \cdot \sqrt{cm}}$</td>
</tr>
<tr>
<td>Pump 1</td>
<td>$p_4$</td>
<td>$0.8223 , \frac{cm}{s}$</td>
</tr>
<tr>
<td>Pump 2</td>
<td>$p_5$</td>
<td>$0.8223 , \frac{cm}{s}$</td>
</tr>
</tbody>
</table>
References


Lebenslauf

Name: Christian Bohn
in: Hamburg
Familienstand: verheiratet, keine Kinder

Schulbildung

1976 – 1980 Grundschule in Hamburg
1980 – 1989 Gymnasium in Hamburg

Studium

1993 – 1994 Studium der Elektrotechnik an der TU Braunschweig
Abschluß Diplom-Ingenieur

Berufs- und Forschungstätigkeit

1994 – 1995 Wissenschaftlicher Mitarbeiter am Lehrstuhl für Elektrische Steuerung und Regelung der Ruhr-Universität Bochum
1996 Forschungsstipendiat der Deutsch-Japanischen Akademischen Burse der Universität Ulm (von der Ruhr-Universität beurlaubt)
1997 – 2000 Wissenschaftlicher Mitarbeiter am Lehrstuhl für Elektrische Steuerung und Regelung der Ruhr-Universität Bochum
2000 Lehrbeauftragter für das Fach Regelungstechnik III an der Technischen Fachhochschule Bochum