Parameter Identification of Bilinear Dynamical Systems
Expectation Maximization using Factor Graphs

Andreas Isler

A thesis submitted to the
ETH Zurich
in partial fulfilment of the requirements for the degree of
Master of Science.

Master Thesis
Automatic Control Laboratory (IFA), ETH Zurich
Model Based Predictive and Distributed Control Lab, University of California, Berkeley

Supervisors:
Prof. Manfred Morari
Prof. Francesco Borrelli
Dr. Philipp Rostalski

September 19, 2011
Abstract

In this thesis a novel identification procedure for bilinear dynamical systems based on Expectation Maximization (EM) using factor graphs is introduced. Factor graphs are a method of representing factorisable functions and allow the implementation of various distributed and modular algorithms e.g. in system theory. EM is an iterative optimization approach for solving maximum likelihood problems. The combination of EM using factor graphs became a relevant research topic during the last decade. In the thesis the two known concepts factor graphs and EM are introduced, before their combined application is presented. Finally the novel approach for bilinear systems is introduced. The thesis concludes with exemplary results, including strength and weaknesses of the new approach.
Acknowledgments

With this thesis I conclude my masters degree at ETH Zurich and this would not have been possible without the support the following people.

In the context of the thesis I am especially grateful to my direct supervisor, Philipp Rostalski. He supported me on site in Berkeley and remotely by Skype after he started a new Job in Germany. I want to thank him for his extraordinary efforts related to my thesis.

Further my thanks go to the Prof. Francesco Borrelli, who offered me to write my master thesis at his lab at the UC Berkeley. He guided the thesis with his advice and interesting discussions.

Thanks also go to my master tutor at ETH, Prof. Manfred Morari, who established the initial contact to Francesco, which was the foundation for the realization of writing my thesis in Berkeley. Further he helped me with discussions and challenging questions in central meetings about the progress of my project. I would also like to thank Christoph Reller, for furthering my understanding for the matter of factor graphs and inspiring discussions on the topic.

For reviewing and giving important feedback, I want to thank my father and my friends.

Last but not least special thanks go to my significant other, who supported me in every stage of my thesis and during my whole studies.

Andreas Isler
## Contents

1 Introduction
   1.1 Parameter Identification and Kalman Filtering ........................................ 3
   1.2 Structure of the Thesis ............................................................................. 4
   1.3 Literature Overview ................................................................................ 4

2 Mathematical Preliminaries ......................................................................... 5
   2.1 Kalman Filter and Smoother ..................................................................... 5
   2.2 Expectation Maximization - Algorithm ...................................................... 7

3 Basics on Factor Graphs ............................................................................. 11
   3.1 Concept of Factor Graphs in an Example ................................................ 11
   3.2 Message Passing Rules ............................................................................ 16
      3.2.1 Gaussian Messages ........................................................................ 17
      3.2.2 Non-Gaussian Messages ................................................................. 19
   3.3 Kalman Filter and Smoother using Factor Graphs ..................................... 22

4 EM-Algorithm using Factor Graphs .............................................................. 27
   4.1 Basic Concept of EM using Factor Graphs .............................................. 27
   4.2 EM using Factor Graphs for Linear Systems ......................................... 30
      4.2.1 Algorithm and Schedules .............................................................. 31
      4.2.2 E-Step-Message of a Matrix Multiplication Node ......................... 32
   4.3 EM using Factor Graphs for Bilinear Systems ......................................... 34
   4.4 Multidimensional Gaussian Variance Matrix Identification ................... 36

5 Implementation Details .............................................................................. 39
   5.1 The Implementation of a Factor Graph ................................................... 39
   5.2 General Message Implementation ........................................................... 41
      5.2.1 Implementing Messages for the Kalman Filter/Smoother ............... 41
      5.2.2 Implementing EM-Messages ........................................................... 43
      5.2.3 Implementing MIMO Systems ......................................................... 43

6 Results ....................................................................................................... 45
   6.1 Identifying the Bilinear Term, Matrices $A$ and $F$ .................................. 45
      6.1.1 Exemplary Results - Low/High Noise Level ................................... 47
      6.1.2 Convergence Depending on Noise Levels ....................................... 51
   6.2 Variance Matrix Identification ................................................................. 53
   6.3 Combined Identification ......................................................................... 54

7 Summary and Outlook ............................................................................... 57

A Appendix: Numerical Example for a Bilinear Model used for the Results ........ 59

References .................................................................................................... 61
1 Introduction

In this thesis the problem of system identification is addressed for a special bilinear model structure. System identification is used to express the relation between known input and output data in a mathematical model. The idea is to fit the parameters of a mathematical model structure to the input and output data of a dynamical system. In this thesis this is done by solving an optimization problem over the output error of the identified system. As an alternative the dynamical processes can be modeled based on physical principals, but this is not studied in this thesis.

The bilinear model used in this thesis is motivated from thermal building control problems, where such bilinear relations play an important role. As for example for an air flow in a heating system, the transported enthalpy is calculated by the product of the mass flow rate, the air temperature difference and some constant. This is a bilinear relation between control inputs (mass flow rate) and the states (air temperature) of the system. But of course there are many other applications for such bilinear systems. The bilinear model structure used in this thesis is in a general form, which allows to use the presented approach in other applications too.

For the identification of such bilinear systems a new extension to an identification concept for linear systems is presented here. It is a combination of a graphical method called factor graphs and an identification algorithm called expectation maximization (EM). The graphical representation is especially beneficial, as it allows modular changes of an algorithm setup. Further a distributed implementation is possible, as for each calculation only local operations (in connection with a single factor graph node) are needed. The factor graph also gives a comprehensive graphical understanding of the algorithm. The algorithm expectation maximization is a widely used concept originating in statistics. It is in many applications successfully applied to solve maximum likelihood problems and other problems involving uncertainty. The modularity of factor graphs can be especially interesting in online applications of a factor graph approach. It gives flexibility for the structure of algorithm and model. It for example allows changes to the structure of input and measurement signals of the system without changing the whole algorithm.

1.1 Parameter Identification and Kalman Filtering

One major problem of system identification is normally the incomplete set of data (hidden data, unmeasured states), which introduces additional complexity to an identification problem. This complexity can make it hard to solve such identification problems with traditional methods. Therefore a way to circumvent the complexity is needed. In the context of this thesis the answer lies in the approach of expectation maximization, this algorithm splits the complex problem into a 2-step iterative process. The clue lies in the fact that the algorithm first estimates the hidden data of an identification problem in one step, and then continues the calculations based on these estimated data parts. By doing this in an iterative way, the estimates and the identification are improved at the same time, until a final result is found. For the estimation of the hidden data, a separate concept is needed. The most commonly used method is the Kalman filter.

Kalman filtering consist of a recursive estimator for the states of a dynamical systems based on a set of uncertain measurements. In this concept not all states have to be measured directly, but the system has to be observable. Underlying is a model to describe the relation between the inputs, the internal states and the measured outputs. Often this is given in terms
of a state space description, which is based on a set of parameter matrices to describe the dynamics of the system. These matrices can either be derived by physical concepts or, in the case of expectation maximization, come directly from the identification process. This is the connection where EM and Kalman filtering go hand in hand.

Now two related concepts lie on the table and factor graphs can give a support for both of them at the same time. In the literature it was shown that factor graphs can be used for Kalman filtering in different ways. Factor graphs give the flexibility to change the traditionally limited focus of Kalman filtering on Gaussian distributions by combining its idea with other distributions without changing the basic approach. In different publications it was presented that the expectation maximization itself can be implemented smoothly based on factor graphs. Some limitations to the convergence properties of the algorithm have to be made, but this is traded in for more flexibility in the application and for an easy to understand graphical representation based on factor graphs.

1.2 Structure of the Thesis

First the basic concepts of Kalman Filtering and expectation maximization (EM) are introduced briefly in Section 2. In the following the concept of factor graphs is explained in more detail. This means the basics are given and the applications for Kalman filtering/ smoothing are explained in Section 3. Later the concept of expectation maximization is explained for the usage with factor graphs in Section 4. Up to this point the work relies on published literature. Then the main contribution of this thesis, the application to a bilinear system structure, is introduced in Section 4.3. Finally the implementation is explained in Section 5 and a set of results (Section 6) is given as a proof of concept, before further topics of research in the area are proposed in the last Section.

1.3 Literature Overview

The two main concepts of this thesis are founded on a wide base of literature and only a limited selection will be given in this paragraph. For the identification concept, the expectation maximization, the base was laid by a publication also known under the abbreviation DLR-Paper [8] in 1977, which is named after the first letters of the names of its authors. It introduces the concept of EM in a general way. Further work to the topic was made in many publications over the last decades. One central contribution came in 1983 from Wu [26], who presented important points about the convergence analysis of the algorithm. Introductory material from newer times can be found by Dellart [7] and Minka [20].

The factor graph framework has its base in newer times. The first appearance in the form how it is used in this thesis was by Kschischang et al. [17] in 2001 and the underlying graph theory was introduced by Forney [10] in the same year. In combination with a Kalman filter it was introduced by Leoliger [18] in 2002.

The combination of EM and factor graphs was presented by Eckford [9] in 2004, the transfer to the usage of EM with state space models in connection with signal processing was made by Dauwels, Korl et al. in 2005 in [4] and [16]. The latter publication wraps up many different concepts and gives a self-contained overview of the central points used in this thesis. In 2009 Dauwels et al. presented in [3] a further important step for the generality of EM using factor graphs with state space models.
2 Mathematical Preliminaries

In this section the two concepts, which are used later for the identification, are introduced in their general form. The Kalman filter and smoother is a set of state estimation methods and is introduced first. Afterwards expectation maximization, a parameter identification method, is described.

2.1 Kalman Filter and Smoother

The Kalman filter and smoother are two directly related, recursive concepts to estimate the state variables of a dynamical system based on a noisy measurement and a discrete model description. The first concept is the Kalman filter, it was introduced by R.E. Kalman in [15]. It is widely used in many applications in different areas. In the context of this thesis it is used to estimate the unknown state of a dynamical system based on all past noisy measurements since its initialization. Iteratively the state is predicted and then updated with the information from the current measurement based on recursive update equations. This procedure can be done online, while the system is running and measurements are acquired in each time step. It can be shown, that the concept leads to the optimal least square solution for a linear filter based on a discrete model with Gaussian noise processes.

The other concept, the Kalman smoother, incorporates the Kalman filter. For one estimate it takes into account all the measurement information of a time window around this time step, not only the past ones. For a time window of length $N$ the estimate of a time step $k$ is based on all $N$ measurements of the time window. In general the Kalman smoother can be seen as an extension to the Kalman filter, because first the filter is applied to the data in a forward sweep over a set of $N$ time steps and in a second step the smoother is used to improve the estimate of the filter by the use of a backward sweep over the same set of $N$ steps. This backward sweep is based on recursive equations, which are based on the same concept as the Kalman filter equations. The combination of these forward and backward sweeps give an estimate for each time step in the window based on all the measured data in the time window, which is the final estimate of the smoother. This concept was e.g. introduced in [2, Chap. 13.2] introduced under the name ‘Optimal Smoothing for Multistage Processes’. For both concepts the underlying equations are given without proof. The interested reader is referred to the stated references; Kalman filter [15, 25]; Kalman smoother [2].

Before the recursive equations for both concepts are given the notation needed is introduced based on the state space model structure, which is used throughout the whole thesis. This linear, time variant, discrete state space model is defined as

$$
x_{k+1} = A_k \cdot x_k + B_k \cdot u_k + v_k \quad \forall k \in \{1, N-1\},
\$$

$$
z_k = C_k \cdot x_k + D_k \cdot u_k + w_k \quad \forall k \in \{1, N\},
\$$

(1)
where

\[ x_k \in \mathbb{R}^{n_x}, \quad u_k \in \mathbb{R}^{n_u}, \quad z_k \in \mathbb{R}^{n_z}, \quad v_k \sim \mathcal{N}(m_v, V_v), \quad w_k \sim \mathcal{N}(m_w, V_w) \]

are system state for time \( k \), control input for time \( k \), deterministic, system output for time \( k \), Gaussian Process Noise, Gaussian Measurement Noise, respectively.

For the Kalman filter and smoother the additional notation \( x_{k|k-1} \) is used. It stands for the prior estimate, meaning the estimate of \( x \) at time \( k \) given the measurements up to the step \( k-1 \). The posterior estimate \( x_{k|k} \) stands for the estimate \( x \) at the time \( k \) given all the measurements up to the current time step \( k \). For both estimates an explicit calculation step is used. The prior estimate is the result of the prediction step:

\[
\begin{align*}
    x_{k|k-1} &= A_k \cdot x_{k-1|k-1} + B_k \cdot u_k, \\
    V_{x_{k|k-1}} &= A_k \cdot V_{x_{k-1|k-1}} \cdot A_k^T + V_V,
\end{align*}
\]

where \( V_{x_{k|k-1}} \) is the variance of \( x \) for the corresponding estimate. As a next step the measurement update gives the posterior estimate:

\[
\begin{align*}
    K_k &= V_{x_{k|k-1}} \cdot C_k^T \cdot \left( C_k \cdot V_{x_{k|k-1}} \cdot C_k^T + V_V \right)^{-1}, \\
    x_{k|k} &= x_{k|k-1} + K_k \left( z_k - C_k \cdot x_{k|k-1} \right), \\
    V_{x_{k|k}} &= (I - K_k \cdot C_k) \cdot V_{x_{k|k-1}},
\end{align*}
\]

where \( K_k \) is the time dependent Kalman gain. These steps are repeated for every time step to calculate the Kalman filter estimate.

As said the calculation of the Kalman smoother is based on the output of the Kalman filter. This output is seen as forward sweep for the Kalman smoother. The Kalman smoother can then be understood as a backward sweep, and the following recursive equations go backward through a time window of length \( N \). They are defined as:

\[
\begin{align*}
    x_{k|N} &= x_{k|k} - V_{x_{k|k}} \cdot A_k^T \cdot \lambda_N, \\
    \lambda_{k-1} &= \left( I - V_{x_{k|k}} \cdot S_k \right) \cdot (A_k^T \cdot \lambda_k - C_k^T \cdot V_W^{-1} \left( z_k - C_k \cdot x_{k|k-1} \right)) = 0, \\
    V_{x_{k|N}} &= P_{k|k} - P_{k|k} \cdot A_k^T \cdot A_k \cdot P_{k|k}, \\
    A_{k-1} &= \left( I - V_{x_{k|k}} \cdot S_k \right) \cdot A_k^T \cdot A_k \cdot A_k^T \cdot \left( I - V_{x_{k|k}} \cdot S_k \right) + S_k \cdot \left( I - V_{x_{k|k}} \cdot S_k \right),
\end{align*}
\]

where \( S_k = C_k^T \cdot V_W^{-1} \cdot C_k \). The notation \( x_{k|N} \) stands for the estimate at time step \( k \) given all measurements in a window of length \( N \). This window for the calculation is assumed to be fixed in time and in an online implementation the estimate \( x_{1|N} \) would have a delay of at least \( N \) time steps.

For the above equations it is assumed that \( D_k = 0 \ \forall k \). If this is not the case all the measurement variables \( z_k \) would be replaced by \( z_k - D_k \cdot u_k \) as \( u_k \) is assumed to be deterministic.
2.2 Expectation Maximization - Algorithm

A nonlinear regression problem can be solved with many different approaches, one of them is presented here. The idea is to find the most likely (i.e. best fitting) set of system parameters given a set of observed data. This is expressed as a maximization problem in

\[ \hat{\theta} = \arg \max_{\theta} \log f(z|\theta), \tag{9} \]

where \( f \) is a probability density function (pdf) of \( z \), a set of observable variables, given the set of model parameters \( \theta \). The function \( f \) maximized in (9) is sometimes also referred to as incomplete log-likelihood. This problem is often hard to solve as it usually is nonlinear and non-convex. One first step to overcome this deficit is to increase the number of dimensions, which describe the relation between the data and parameter space, by introducing a new set of hidden variables \( x \). The new function to maximize is called complete log-likelihood \( f(x,z|\theta) \), and problem (9) can be written as

\[ \hat{\theta} = \arg \max_{\theta} \log \int_{D_x} f(x,z|\theta) \, dx = \arg \max_{\theta} \log f(z|\theta). \]

If the set \( x \) or its distribution was known, the maximum likelihood problem based on the complete log-likelihood function could degenerate to a simpler regression problem and therefore be much easier to solve. This thought is somehow reflected in the 2-step iterative approach of Expectation Maximization (EM). Instead of solving the original problem with the maximization of the incomplete log-likelihood \( \log f(z|\theta) \) directly, it is the idea of the EM to first calculate the expected value of the complete log-likelihood function, \( f(x,z|\theta) \), given the measured set \( z \) and a set of parameters \( \theta^j \) and second to maximize this expected value to get \( \theta^{j+1} \). In other words EM means to iteratively compute the distribution of the hidden variables \( p(x|z,\theta^{(j)}) \) (assuming \( \theta^j \) is fixed) for the expected value of the complete log-likelihood function and then maximizing the expected value over \( \theta \) (assuming \( p(x|z,\theta^{(j)}) \) is fixed), which leads to the new set \( \theta^{j+1} \).

The expected value of the complete log-likelihood function is written as

\[ Q(\theta|\theta^{(j)}) = E_p \left[ \log f(x,z|\theta) \left| z, \theta^{(j)} \right. \right] \]

\[ = \int_{D_x} p(x|z,\theta^{(j)}) \log f(x,z|\theta) \, dx, \tag{10} \]

where \( p(x|z,\theta^{(j)}) \) is the probability density function of the hidden set \( x \) given the observed data \( z \) and the set of \( \theta^j \) from the previous iteration step. To motivate this approach the relation between the expected value and the incomplete log-likelihood function is derived.
Starting with the incomplete log-likelihood function one finds, cf. [20]:

\[
\log f(z|\theta) = \log \int_{D_x} f(x,z|\theta) \, dx
= \log \int_{D_x} p(x|z,\theta^{(j)}) \frac{f(x,z|\theta)}{p(x|z,\theta^{(j)})} \, dx
\geq \int_{D_x} p(x|z,\theta^{(j)}) \log \frac{f(x,z|\theta)}{p(x|z,\theta^{(j)})} \, dx =
q(\theta|\theta^{j}) = \int_{D_x} p(x|z,\theta^{(j)}) \log f(x,z|\theta) \, dx - \int_{D_x} p(x|z,\theta^{(j)}) \log p(x|z,\theta^{(j)}) \, dx.
\]

The step from (11) to (12) holds with the Jensen Inequality. The derived relation \( q \) in (13) is a lower bound to the incomplete log-likelihood function

\[
\log f(z|\theta) \geq q(\theta|\theta^{j}),
\]

where equality holds for

\[
\log f(z|\theta^{j}) = q(\theta^{j}|\theta^{j}),
\]

For a qualitative illustration for the lower bound to the incomplete log-likelihood function see Figure 2.1.

The two iterative steps of the EM-Algorithm are

E-Step: \( Q(\theta | \theta^{(j)}) = E[\log f(x,z|\theta) | z, \theta^{(j)}] \)

M-Step: \( \theta^{(j+1)} = \arg\max_{\theta} Q(\theta | \theta^{(j)}) \).

The maximization defined for the M-step satisfies the needs of the algorithm, because the second part of the lower bound \( q(\theta|\theta^{j}) \) in (13) is not related to the maximization variable \( \theta \). The actual function value of the lower bound is not of interest in the context of the EM-Algorithm, as only the maximizer \( \hat{\theta} \) is needed.

In [26] Wu gives a proof which guarantees local convergence under the condition that the likelihood function is unimodal \(^1\). Readers interested in a detailed tutorial on the algorithm are also referred to [7, 20].

\(^1\) A distribution is called unimodal if the distribution has only one maximum. The function argument for this maximum is called mode.
2.2 Expectation Maximization - Algorithm

Objective Function
\[ \log f(z|\theta) \]

Lower Bound
\[ q(\theta | \theta^j) \]

New Guess
\[ \theta^{j+1} \]

Current Guess
\[ \theta^j \]

\textbf{Figure 2.1:} Qualitative illustration for the lower bound to the complete log-likelihood in Equation (14) and for the EM-algorithm in Equation (15).
3 Basics on Factor Graphs

This section explains the concept of factor graphs, which is a graphical method to represent factorisable functions in graphs. On these graphs a method called message passing can be used for calculations in many different ways. The graph representation facilitates the calculation of complex multi-variable functions with simple message passing rules and even allows the implementation of many different algorithms based on message passing. It will be shown later in this section how state estimation in the sense of a Kalman filter/smooth can be done as message passing on a factor graph without prior knowledge of the Kalman concepts. But this is only one of many applications, which use the message passing on factor graphs as an underlying framework for complex calculations.

The way this concept is presented in this thesis focuses on dynamical systems and was proposed for the first time in [18], later [3] and [16] elaborated it in more detail. Factor graphs in a more general way were elaborated in [17] and [10]. In the latter paper Forney introduced a special form of factor graphs, usually referred to as Forney-style factor graphs (FFG). The following sections introduce the basic concepts of factor graphs, which are needed for the concept of expectation maximization using factor graphs, which is explained in Section 4. The introduction is based on the above expositions and for further details the reader is referred to these publications.

3.1 Concept of Factor Graphs in an Example

The concept of factor graphs is first introduced by means of a simple example, without going into details of proofs. Readers which are interested in more details can find further introductory material in [16, 19].

Example 3.1.1, cf. [19, Eq.14] As an example a factorizable function of seven variables is used, which can represent any function that allows a factorization of the form

\[ f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = f_a(x_1)f_b(x_1, x_2, x_3)f_c(x_3, x_4, x_5)f_d(x_4)f_e(x_5, x_6, x_7).\]

This function \( f \) can be represented by a factor graph as shown in Figure 3.1.

![Figure 3.1](image-url)  

**Figure 3.1:** Representation of function \( f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \) as a factor graph used in Example 3.1.1

This graph represents the entire function \( f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \) and consists of edges, half edges and nodes, which are defined in the following way:
• Every node represents one factor of the global function, which is also referred to as node function.
• Each edge represents exactly one variable of the global function.
• A half-edge stands for a variable that is only part of one node function. Later, half-edges are often used to represent external inputs to the function/system or for initial conditions.

The definition of edges implies that the variable can technically only be part of two node functions as every edge only has at most two ends. Equality nodes are established to handle this issue by introducing two artificial variables, which in fact always need to attain the same value, but may represent different sources of information and in this sense may be different during the process of message passing. This type of node is derived in Section 3.2.1, and provides a prime example of how factor graphs can be used to combine different uncertain information about one variable.

Message Passing on a Factor Graph

The following calculation of a node function illustrates the so-called Summary-Propagation algorithm (SPA). The SPA is also known as belief propagation or message passing and is defined in more detail below. Only two special cases of the SPA will be used. First the Sum-Product algorithm and then the Max-Product algorithm are introduced in this section.

If a factor graph is cycle free, the node function for one or several variables can be calculated by a such a message passing on a factor graph. In the context of a probability density function such node functions are also called marginalizations.

The Sum-Product algorithm is introduced by continuing Example 3.1.1. Now the function \( f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \) is assumed to be a probability density function of seven stochastic variables. Even though this example is also valid for any other function of this form, this classification is made to introduce the related terminology, which is used later. In this setup the marginalized probability function e.g. for the variable \( x_3 \) is

\[
\bar{f}(x_3) = \int \int \int \int f_a(x_1) f_b(x_1, x_2, x_3) f_c(x_3, x_4, x_5) \ldots f_d(x_4) f_e(x_5, x_6, x_7) \, dx_1 \, dx_2 \, dx_4 \, dx_5 \, dx_6 \, dx_7
\]

(16)

\[
= \int_{x_1} \left[ f_a(x_1) \int_{x_2} f_b(x_1, x_2, x_3) \, dx_2 \right] dx_1 \ldots
\]

(17)

In Equation (17) the multivariate integral of (16) is factorized in simpler integrals, in terms of factor graphs local integrals. This factorization simplifies the calculation of the marginalization for \( x_3 \). By applying the Summary-Propagation algorithm on the factor graph for the global function \( f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \), exactly this factorization is achieved in an intuitive manner. With some standardization introduced later the integrals can be calculated easily.

This iterative procedure known as message passing starts on one of the end nodes of the factor graph in Figure 3.2. On every node the summary of all ‘incoming’ messages is calculated and propagated along the edge of the outgoing message. The ‘incoming’ and ‘outgoing’ is regarded from the node’s perspective and can change from one situation to another. Starting from every node the outgoing message is chosen in the general direction towards the
3.1 Concept of Factor Graphs in an Example

The message from node \( f_a \) on \( x_1 \) in direction of \( f_b \) is \( \mu_{f_a \rightarrow x_1} \). As the node \( f_a \) has no other edges coming in, the message consists only of the function itself depending only on the edge variable \( x_1 \) and is

\[
\mu_{f_a \rightarrow x_1}(x_1) = f_a(x_1).
\]

The next message of interest is already the message on edge \( x_3 \) pointing upwards. It summarizes all messages coming into the node \( f_b \), which is in fact summarizing the entire lower dashed box of the graph as a function of \( x_3 \). The outgoing node message is in general calculated as a marginalization of all incoming messages times the node function. For the upward message on edge \( x_3 \) it is

\[
\mu_{f_b \rightarrow x_3}(x_3) = \int_{x_1} \int_{x_2} \mu_{f_a \rightarrow x_1}(x_1)f_b(x_1, x_2, x_3) \, dx_1 \, dx_2.
\]

The message on \( x_3 \) coming from above is calculated in the same way starting from nodes \( f_d \) and \( f_e \).

\[
\mu_{f_d \rightarrow x_4}(x_4) = f_d(x_4), \quad \mu_{f_e \rightarrow x_5}(x_5) = \int_{x_6} \mu_{f_c \rightarrow x_5}(x_5)f_e(x_5, x_6, x_7) \, dx_6 \, dx_7.
\]

The last step may appear unfamiliar, but messages from half edges are defined neutral. In the case of multiplication this means they carry a message equal to 1 unless stated otherwise. The message out of node \( f_c \) is created by integrating the product of all its incoming messages and the node function over the variables of all edges except the one carrying the outgoing message.

\[
\mu_{f_c \rightarrow x_3}(x_3) = \int_{x_4} \mu_{f_d \rightarrow x_4}(x_4)\mu_{f_e \rightarrow x_5}(x_5)f_c(x_3, x_4, x_5) \, dx_4 \, dx_5.
\]

To complete the calculation of the marginalized probability density function of \( x_3 \), the messages on the edge \( x_3 \), one in each direction, are multiplied. In Equation (19) it can be seen
that the multiplication of the two messages on edge $x_3$ lead to the same factorization of the integral as derived algebraically in (17):

$$
\bar{f}(x_3) = \mu_{f_b \rightarrow x_3} \mu_{f_c \rightarrow x_3}
$$

(18)

$$
= \int_{x_1} f_a(x_1) \int_{x_2} f_b(x_1, x_2, x_3) \, dx_2 \, dx_1 \cdot
\begin{array}{c}
\mu_{f_a \rightarrow x_1} \\
\mu_{f_b \rightarrow x_3} \\
\mu_{f_c \rightarrow x_3}
\end{array}
$$

(19)

This factorization of the integral probably does not appear as a real simplification, which would justify the use of factor graphs, but in Section 3.2 it is introduced how the calculation of such integrals is simplified by the use of the right parameterization. Another advantage is that all calculations can be done locally. Figure 3.2 illustrates the Summary-Propagation algorithm to derive the node function depending on $x_3$ pictorially. Every message is the summary of everything within its dashed box. To terminate the SPA all other node functions are calculated in the same way, but of course already calculated messages can be reused.

After introducing the message passing method in an example the underlying definitions for the Sum-Product algorithm and the SPA following [16] are given. Remember the Sum-Product algorithm is a special case of the Summary-Propagation algorithm.

Definition: Summary-Propagation-Algorithm (SPA) cf. [16, Section 2.2]: In general the SPA calculates two messages on each edge of the factor graph, one in each direction. By the combination of these two messages the local summary as function of the edge variable can be calculated. If the factor graph is cycle free the SPA starts at the leaves and proceeds with the next node message as soon as all input messages of the next node are available. This guarantees that none of the messages are calculated twice. The algorithm stops as soon as the node function consisting of the product of two opposed messages is available for every edge.

In factor graphs with cycles the calculation of the node function is an iteration over the cycles.

There is no guarantee that the SPA converges on a factor graph with cycles, but in practice many examples have shown that the SPA often converges to a stable point with close enough approximations [16, 19]. In [23] and [24] it is even shown that purely Gaussian factor graphs lead to correct node functions regardless of the cycles, provided the graph converges. For each graph with cycles optimal update schedules and the analysis of the convergence needs to be elaborate separately. The SPA is also known under the name belief propagation.
Definition: Sum-Product algorithm cf. [16, Section 2.2]: The message out of a factor node $f(x_1, \ldots, x_N, y)$ along the edge $y$ is the product of the function $f(x_1, \ldots, x_N, y)$ and all messages towards $f$ along all other edges, summarized (integrated) over all variables except $y$. This concept is underlying the message passing rules introduced in Section 3.2.

$$
\mu_{f \rightarrow g}(y) = \int_{x_N} \cdots \int_{x_1} f(x_1, \ldots, x_N, y) \cdot \mu_{x_1 \rightarrow f}(x_1) \cdots \mu_{x_N \rightarrow f}(x_N) \, dx_1 \cdots dx_N
$$

Note: For discrete variables the integrals are to be replaced by summations.

Beside the introduced Sum-Product algorithm another variant of the SPA can be formulated for solving problems of the form

$$(\hat{x}_1, \ldots, \hat{x}_N) = \arg \max_{x_1, \ldots, x_N} f(x_1, \ldots, x_N),$$

cf.[19]. In two steps this is, first

$$\hat{f}_k(x_k) = \max_{x_1, \ldots, x_k \text{ except } x_k} f(x_1, \ldots, x_N)$$

then

$$\hat{x}_k = \arg \max_{x_k} \hat{f}_k(x_k).$$

In this setup the Max-Product algorithm can be used to calculate the $\hat{f}_3(x_3)$ by message passing. If now the function $f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ from Example 3.1.1 is maximized over $x_3$ this is achieved by applying the same steps as in the Sum-Product example and replacing every integration with a maximization function over the same variable. Starting from the

$$\hat{f}(x_3) = \max_{x_1, \ldots, x_7 \text{ except } x_3} f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$

$$= \max_{x_1} \left[ \max_{x_2} \left[ \left[ \left[ \max_{x_3} \left[ \left[ \max_{x_4} \left[ \left[ \max_{x_5} \left[ \left[ \max_{x_6} \left[ \left[ \max_{x_7} f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \right] \right]\right]\right]\right]\right]\right]\right]\right]\right]$$

Then the maximizer $\hat{x}_3$ is

$$\hat{x}_3 = \arg \max_{x_3} \hat{f}_3(x_3) = \arg \max_{x_3} \mu_{f_b \rightarrow x_3} \mu_{f_c \rightarrow x_3}.$$
In [19] the Max-Product algorithm is defined like in the box below. Note that for some special function parameterization the Sum-Product and the Max-Product algorithms are equivalent, one example thereof are Gaussian probability distributions, for which the mean and the mode (i.e. the maximum of \( f(x) \)) agree.

**Definition: Max-Product algorithm** The message out of the node/factor \( f(x_1, \ldots, x_N, y) \) along the edge \( y \) is formed as the maximization over the product of \( f \) and all incoming messages except \( y \). The maximization is done over all involved variables except \( y \).

\[
\mu_{f \rightarrow y}(y) = \max_{x_1 \ldots x_N} (f(x_1, \ldots, x_N, y) \cdot \mu_{x_1 \rightarrow f}(x_1) \cdots \mu_{x_N \rightarrow f}(x_N))
\]

**Node Function Calculation for more than one Variable** If a node function, e.g. marginalization, of more than one variable is of interest, the edges of the involved variables are grouped in a 'box' and a node function for this box is derived. Then the node function is the result of the product of the box-node function and all incoming messages. A similar approach is used for the derivation of the E-step message in Section 4.2.2.

**Factor Graph Notation** Even though factor graph edges are not directed they are often drawn with arrows. This is to simplify the notation of the messages. Then \( \mu_X \) refers to the message in the direction of the 'directed' edge of variable \( X \) and \( \mu_X \) is the corresponding message on the same edge \( X \) but in the inverse direction. Further capital letters in factor graphs are used to denote edges carrying messages, on the other hand lowercase letters are in general used for known constants. This notation will be used for all succeeding sections of this thesis.

### 3.2 Message Passing Rules

For a specified type of factorisable functions represented by graphs it is possible to specify specific message passing rules. The function type used in this section will be probability density functions (pdf) of continuous random variables. A pdf is often of the form of a specific probability distribution, e.g. Gaussian distribution. For some probability distributions the message passing using factor graphs can be reduced to simple update rules of the distribution parameters. An important property of a distribution to allow the usage of such rules is that it is closed under multiplication, but for many applications of factor graphs this restriction can be relaxed as long as it holds "up to scale", i.e. with a proportionality constant not depending on the random variable of the distribution. The omitted constant would only be used for the normalization of the distribution \( \left( \int_{-\infty}^{\infty} f(x) dx = 1 \right) \).

In the following sections some basic message formats and rules are introduced, first for Gaussian messages cf.[16, 19], later a non-Gaussian concept with the Inverse-Wishart distribution.
is introduced as an extension to existing concepts.

### 3.2.1 Gaussian Messages

The Gaussian distribution is widely used for the approximation of real world stochastic processes. One among many reasons is that it is closed under summation and "up to scale" under multiplication. This is favorable for its usage with factor graphs. It allows the formulation of simple parameter update rules for the passing of Gaussian messages over certain standardized node functions. These update rules are presented in this section, but only one exemplary rule derivation is elaborated in detail, every other rule is only listed for completeness. The whole introduction of these Gaussian message passing rules is based on [18].

First of all the two equivalent parameterizations of the Gaussian distribution used here are introduced. Even though they are equivalent, their parameters are not in every situation equally comfortable to work with, numerically stable or even feasible, despite that both can be used to describe Gaussian messages. The two parameterizations are

\[
\mathcal{N}(m_x, V_x) \sim \frac{|V_x|^{-\frac{1}{2}}}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}(x-m_x)^T V_x^{-1} (x-m_x)},
\]

\[
\tilde{\mathcal{N}}(W_x m_x, W_x) \sim \frac{|W_x|^{\frac{1}{2}}}{\sqrt{(2\pi)^n}} e^{-\frac{1}{2}(W_x m_x)^T W_x^{-1} (W_x m_x)} e^{-\frac{1}{2}x^TW_x x + x^TW_x m_x},
\]

where \(x, m_x \in \mathbb{R}^n\) and \(V_x, W_x\) are positive definite matrices of \(\mathbb{R}^{n\times n}\). The variable \(m_x\) is the mean of the stochastic variable \(x\), \(V_x\) is its variance and \(W_x = V_x^{-1}\) its information matrix.

**Note:** \(|\bullet|\) stands for the determinant of the matrix. Often the first constant coefficient is ignored, in that case equations hold only up to scale and are noted with the \(\propto\) operator.

**Note:** For a Gaussian factor graph edge \(X\), which is drawn with an arrow, this notation is used throughout the thesis:

- \(\tilde{m}_X\) Mean of the message in direction of the arrow on the edge
- \(\tilde{V}_X\) Variance of the message in direction of the arrow on the edge
- \(\tilde{W}_X\) Information matrix of the message in direction of the arrow on the edge, equal to \(\tilde{V}_X^{-1}\)
- \(\tilde{W}_X \tilde{m}_X\) Weighted mean of the message in direction of the arrow on the edge
- \(\tilde{m}_X\) Mean of the message in the opposite direction of the arrow on the edge
- \(\tilde{V}_X\) Variance of the message in the opposite direction of the arrow on the edge
- \(\tilde{W}_X\) Information matrix of the message in the opposite direction of the arrow on the edge, equal to \(\tilde{V}_X^{-1}\)
- \(\tilde{W}_X \tilde{m}_X\) Weighted mean of the message in the opposite direction of the arrow on the edge
- \(m_X\) Mean of the marginal on the edge, \(m_X = V_X \cdot W_X \cdot m_x\)
- \(V_X\) Variance of the marginal on the edge, \(V_X = W_X^{-1}\)
- \(W_X\) Information matrix of the marginal on the edge, \(W_X = \tilde{W}_X + \tilde{W}_X\)
- \(W_X m_X\) Weighted mean of the marginal on the edge, \(W_X m_X = \tilde{W}_X \tilde{m}_X + \tilde{W}_X \tilde{m}_X\)
This notation is taken from [19]. The calculation of the marginal parameters results from the product of the opposing messages on an edge.

**Gaussian Equality Node Function** As mentioned earlier, at most two node functions can share one variable/edge. To overcome this limitation the variable is replicated by an equality node, which is the first standardized node function introduced here. This node function is in the context of messages as distributions used to combine uncertain information from two different sources (subgraphs) in one message.

The definition of the equality node propagation rule relies on the relaxed condition that messages are only proportionally equal to each other. This in general does not violate the assumption of parameters being equal, which is the only part of the message that is of interest.

The notation for the equality node is shown in Figure 3.3. This derivation of the equality node update rule is taken from [18]. It is based on $\hat{N}(W_m,W)$ as the parametrization for the Gaussian messages.

![Figure 3.3: Notation of an equality node.](image)

The equality node is introduced to satisfy the condition

$$X \triangleq Y \triangleq Z.$$  \hspace{1cm} (22)

The node function $f_-(x,y,z)$ for this condition makes use of the Dirac function defined as

$$\int \delta(x) \, dx = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{else.} \end{cases}$$

The node function can thus be defined as

$$f_-(x,y,z) = \delta(z-x)\delta(z-y).$$

Applying the Sum-Product algorithm to the graph in Figure 3.3 leads to the equation for the message of edge $Z$.

$$\mu_Z(z) = \int \int \delta(z-x)\delta(z-y) \mu_X(x) \mu_Y(y) \, dx \, dy$$  \hspace{1cm} (23)

$$= \mu_X(z) \cdot \mu_Y(z)$$

Because of the Dirac function the integral in (23) can be solved easily and the message $\mu_Z$ is derived. Up to this step this is true for the equality node for any message parameterization not limited to Gaussian distributions. For the Gaussian case the parameter update rules are derived next. For this derivation it is enough to satisfy equality up to scale as only the
3.2 Message Passing Rules

parameters and not the normalized values are of interest. This is why the constant coefficients before the exponential functions are omitted directly.

\[ \mu_Z(z) \propto e^{-\frac{1}{2}(z-m_Z)^T W_Z^{-1} (z-m_Z)} \]

Comparing Equations (24) and (25) and applying [19].(Eq. 141-143) leads to the update rules for the parameterization \( \mathcal{N}(Wm,W) \) which are

\[ \mu_Z(z) \propto e^{-\frac{1}{2}(z-m_Z)^T W_Z^{-1} (z-m_Z)} \]

Comparing Equations (24) and (25) and applying [19].(Eq. 141-143) leads to the update rules for the parameterization \( \mathcal{N}(Wm,W) \) which are

\[ \mu_Z(z) \propto e^{-\frac{1}{2}(z-m_Z)^T W_Z^{-1} (z-m_Z)} \]

For the other parameterization of the Gaussian messages the update rules are neither straightforward nor practical. If numerically possible, simple conversions between the parameterizations, can help to overcome this problem. A numerically better option is to use the node combination in Table 3.2.(1).

Other Gaussian message passing rules for a variety of useful node functions used in this thesis are listed in Table 3.1.

3.2.2 Non-Gaussian Messages

Inverted Gamma Messages were introduced in different papers ([16],[19]) to use e.g. for the identification of scalar variances of Gaussian distributions. For interested readers these papers can serve as source for look up tables and the proof of the message passing rules.

Inverse-Wishart Messages In this paper the identification of variances of Gaussian distributions is expanded to a full matrix problem. The process of identification is further explained in Section 4.4, but the necessary message passing rule is already introduced here. The Inverse-Wishart distribution is the multivariate extension of the Inverted Gamma distribution and is often used as a conjugate distribution for the variance of multivariate Gaussian distributions in Bayesian statistics.

The probability density function of the Inverse-Wishart distribution for a square matrix \( A \in \mathbb{R}^{n \times n} \) is defined as

\[ f(A) \sim W^{-1}(\Xi, \rho) = |\Xi|^{\rho/2} |A|^{-(\rho+n+1)/2} e^{-\frac{1}{2} tr(\Xi A^{-1})} \]

with the parameters \( \Xi \in \mathbb{R}^{n \times n} \) and \( \rho \in \mathbb{R} \). The Inverse-Wishart distribution also has the property of keeping proportionality under multiplication, this property can be used to formulate the message passing rule for the equality constraint node, like in Figure 3.3. This rule will be sufficient for the application of the Inverse-Wishart messages in this thesis.

Starting from the general derivation for the equality constraint node in Equation (23) it can
Table 3.1: Gaussian message passing rules, see also Table 3.2. Further details can be found in [16] and [19].

Note: The right arrow notation is used to denote that these are parameters of the message in direction of the notation arrow on the graph. Arrows to the left denote parameters of the message in the opposite direction of the notation arrows on the graph.

\^1\: for an invertible A, for other cases [19, Tab.5-6]
### 3.2 Message Passing Rules

<table>
<thead>
<tr>
<th>Node</th>
<th>( \mathcal{N}(m,V) )</th>
<th>( \tilde{\mathcal{N}}(Wm,W) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>( \vec{m}_Z = \vec{m}_X + \vec{V}_X A^T \cdot g(\vec{V}_Y, \vec{V}_X) \cdot (\vec{m}_Y - A\vec{m}_X) )</td>
<td>( \vec{W}_Z \vec{m}_Z = \vec{W}_X \vec{m}_X + A \vec{W}_Y \vec{m}_Y )</td>
</tr>
<tr>
<td></td>
<td>( \vec{V}_Z = \vec{V}_X - \vec{V}_X A^T \cdot g(\vec{V}_Y, \vec{V}_X) A \vec{V}_X )</td>
<td>( \vec{W}_Z = \vec{W}_X + A \vec{W}_Y A^T )</td>
</tr>
<tr>
<td>( Y )</td>
<td>( \vec{m}_X = \vec{m}_Z + \vec{V}_Z A^T \cdot g(\vec{V}_Y, \vec{V}_Z) \cdot (\vec{m}_Y - A\vec{m}_Z) )</td>
<td>( \vec{W}_X \vec{m}_X = \vec{W}_Z \vec{m}_Z + A \vec{W}_Y \vec{m}_Y )</td>
</tr>
<tr>
<td></td>
<td>( \vec{V}_X = \vec{V}_Z - \vec{V}_Z A^T \cdot g(\vec{V}_Y, \vec{V}_Z) A \vec{V}_Z )</td>
<td>( \vec{W}_X = \vec{W}_Z + A \vec{W}_Y A^T )</td>
</tr>
</tbody>
</table>

**Note:** The right arrow notation is used to denote that these are parameters of the message in direction of the notation arrow on the graph. Arrows to the left denote parameters of the message in the opposite direction of the notation arrows on the graph.

\[ g(a,b) = (a + AbA^T)^{-1} \]

\[ h(a,b) = (a + A^TbA)^{-1} \]

**Table 3.2:** Gaussian message passing rules, further details can be found in [16] and [19].

---

**Figure:**

- **Node:** \( X \rightarrow Z \)
- **Equality / Matrix Multiplication Combination:**
  - \( \vec{m}_Z = \vec{m}_X + \vec{V}_X A^T \cdot g(\vec{V}_Y, \vec{V}_X) \cdot (\vec{m}_Y - A\vec{m}_X) \)
  - \( \vec{V}_Z = \vec{V}_X - \vec{V}_X A^T \cdot g(\vec{V}_Y, \vec{V}_X) A \vec{V}_X \)
  - \( \vec{m}_X = \vec{m}_Z + \vec{V}_Z A^T \cdot g(\vec{V}_Y, \vec{V}_Z) \cdot (\vec{m}_Y - A\vec{m}_Z) \)
  - \( \vec{V}_X = \vec{V}_Z - \vec{V}_Z A^T \cdot g(\vec{V}_Y, \vec{V}_Z) A \vec{V}_Z \)
  - \( \vec{m}_Z = \vec{m}_X + A \vec{m}_Y \)
  - \( \vec{V}_Z = \vec{V}_X + A \vec{V}_Y A^T \)
  - \( \vec{m}_X = \vec{m}_Z - A \vec{m}_Y \)
  - \( \vec{V}_X = \vec{V}_Z + A \vec{V}_Y A^T \)

**Node:** \( X \rightarrow Z \)

- **Equality / Matrix Multiplication Combination:**
  - \( \vec{m}_Z = \vec{m}_X + A \vec{m}_Y \)
  - \( \vec{V}_Z = \vec{V}_X + A \vec{V}_Y A^T \)
  - \( \vec{m}_X = \vec{m}_Z - A \vec{m}_Y \)
  - \( \vec{V}_X = \vec{V}_Z + A \vec{V}_Y A^T \)

- **Table 3.2:** Gaussian message passing rules, further details can be found in [16] and [19].

**Note:** The right arrow notation is used to denote that these are parameters of the message in direction of the notation arrow on the graph. Arrows to the left denote parameters of the message in the opposite direction of the notation arrows on the graph.
be written as (constant terms are omitted)

\[
\bar{\mu}_Z(Z) \propto \bar{\mu}_X(X) \cdot \bar{\mu}_Y(Y)
\]
\[
\propto |Z|^{-(\rho_X+n+1)/2} e^{-\frac{1}{2}tr(\Xi_X Z^{-1})}.
\]
\[
\propto |Z|^{-(\rho_Y+n+1)/2} e^{-\frac{1}{2}tr(\Xi_Y Z^{-1})}.
\]
\[
\propto |Z|^{-(\rho_X+\rho_Y+2n+2)/2} e^{-\frac{1}{2}tr((\Xi_X+\Xi_Y) Z^{-1})}.
\]
\[
\propto |Z|^{-(\rho_X+n+1)/2} e^{-\frac{1}{2}tr(\Xi_Z Z^{-1})}.
\]

where the bold print variables represent square, positive definite matrices and the equality node message passing rule reads straightforward as

\[
\Xi_Z = \Xi_X + \Xi_Y,
\]
\[
\rho_Z = \rho_X + \rho_Y + n + 1.
\]

### 3.3 Kalman Filter and Smoother using Factor Graphs

In the upcoming paragraphs it will be presented how the message passing on factor graphs can be used to do state estimation in the sense of a Kalman filter/smooother. For this the factorisable function will be a probability density function of all variables of a dynamical system. The Gaussian message format introduced in Section 3.2.1 will be used. Remember each message is a parametrized probability density function, which represents a summary of the sub graph ‘behind’ the message.

First it is shown what abstraction needs to be done to represent the Kalman concept on a single probability density function.

The time dependent Kalman filter introduced in Section 2.1 leads to the posterior estimate of the state \(x_k\) depending on the noisy measurement \(z_k\) and the known input \(u_k\) of all time steps in the past since the initialization. This can be written in the form of a conditional probability density function as

\[
f(x_k|z_1, \ldots, z_k, u_1, \ldots, u_k, x_1, V_{x_1}),
\]

where \(x_1\) and \(V_{x_1}\) are initial conditions for the Gaussian mean and variance for state \(x\). Following the definition of conditional probability [1, Section 2.6] this conditional expression is proportional to the probability density function of all involved variables.

\[
f(x_k, z_1, \ldots, z_k, u_1, \ldots, u_k, x_1, V_{x_1}) \propto f(x_k|z_1, \ldots, z_k, u_1, \ldots, u_k, x_0, V_{u_0}),
\]

This density function can be drawn as a factor graph on which a simple message passing sequence leads, by taking the maximum a posteriori of the resulting distribution of \(x\), directly to the derivation of the traditional Kalman filter. Remember the marginalized messages for an edge represent the probability distribution of the edge variable.

In the regular Kalman filter all variables are assumed to be Gaussian distributed, but the factor graph would in a similar way allow to apply the concept of the Kalman filter to other distributions or even combinations, as long as message passing rules can be derived. One time section of such a factor graph for the Kalman filter is depicted in Figure 3.4 (\(D_k = 0 \ \forall \ k\)), where edges of random variables are labeled with capital letters and deterministic variables are labeled with a lowercase letters. It strongly resembles a traditional state space block diagram, even though it is the representation of the probability density function. For the Kalman filter
Figure 3.4: Factor graph for the density function \( f(x_k, x_{k+1}, z_k, z_{k+1}, u_k) \) under the state space description of (1) \( (D_k = 0 \forall k) \). Under the proportionality condition of (32) this is one time section of a factor graph of a Kalman filter/smoother. The numbers in \( \bigcirc \) refer to the messages defined in the Equations (33) and (34).
up to step \( k \), \( k \) such time sections are connected in a series. By applying the rules of the Tables 3.1 - 3.2 the messages for the Kalman filter can be defined. The messages are for \( D_k = 0 \ \forall \ k \) defined as:

<table>
<thead>
<tr>
<th>Msg #</th>
<th>Kalman Measurement Update Step</th>
<th>Kalman Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \bar{m}<em>{Y_k} = z_k - \bar{m}</em>{W_k} )</td>
<td>( x_{k</td>
</tr>
<tr>
<td></td>
<td>( \bar{V}<em>{Y_k} = 0 + \bar{V}</em>{W_k} )</td>
<td>( V_{x_{k</td>
</tr>
<tr>
<td>2</td>
<td>( \bar{m}<em>{X_k} = \bar{m}</em>{X_k}' + \bar{V}<em>{X_k}'(\bar{V}</em>{Y_k} + C_k^T \bar{V}<em>{X_k}' C_k)^{-1}(\bar{m}</em>{Y_k} - C_k \bar{m}_{X_k}') )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \bar{V}<em>{X_k} = \bar{V}</em>{X_k}' - \bar{V}<em>{X_k}' C_k^T (\bar{V}</em>{Y_k} + C_k^T \bar{V}<em>{X_k}' C_k)^{-1} C_k \bar{V}</em>{X_k}' )</td>
<td></td>
</tr>
</tbody>
</table>

### Kalman Prediction Step

| 3     | \( \bar{m}_{X_{k+1}''} = A_k \bar{m}_{X_k} \) | \( x_{k+1|k} \) |
|       | \( \bar{V}_{X_{k+1}''} = A_k \bar{V}_{X_k} A_k^T \) | \( V_{x_{k+1|k}} \) |
| 4     | \( \bar{m}_{X_{k+1}''} = \bar{m}_{X_{k+1}''} + \bar{m}_{V_{k}} \) | |
|       | \( \bar{V}_{X_{k+1}''} = \bar{V}_{X_{k+1}''} + \bar{V}_{V_{k}} \) | |
| 5     | \( \bar{m}_{X_{k+1}'''} = \bar{m}_{X_{k+1}''} + B_k u_k \) | |
|       | \( \bar{V}_{X_{k+1}'''} = \bar{V}_{X_{k+1}'''} + 0 \) | |

The numbers in the left column are equivalent to the message numbers in Figure 3.4.

As it can be seen in these equations, simply by combing the messages along the graph, the traditional Kalman update equations are found (compare Equations (2)-(4)), but in the case of message passing this is only half the way to go and assumes that there is no new information in the backward messages. Actually using only the forward messages would be an approximation, as no marginalization on the edges is done. More about the use of backward messages is explained in the next paragraphs.

Referring to the introduction of the Summary Propagation Algorithm in Section 3.1 the message passing is as mentioned not finished. To complete the SPA the message in the opposite direction needs to be calculated for all edges/variables of interest. This will allow to calculate the marginalized probability density function for these variables.

For this step it is assumed that \( N \) time steps are of interest and all measurements thereof are available. This allows to start the message passing at the end of the time series, from step \( N \) to 1. This backward pass leads to the equivalent of the Kalman smoother, which is explained in Section 2.1. The smoother improves the estimate at each time step \( k \) of the Kalman filter by taking into account the future measurements between step \( k \) and step \( N \).

For the Kalman smoother the resulting conditional probability density function is

\[
f(x_k | z_1, \ldots, z_N, u_1, \ldots, u_N, x_1, V_{x_1}, x_N, V_{x_N}),
\]

where \( x_N \) and \( V_{x_N} \) are the assumed values of the last step \( N \). These are used to initialize the smoother equations or the backward pass of the messages, respectively. Like for the

\[\text{2The deterministic measurement } z \text{ has variance equal to zero.}\]

\[\text{3The deterministic input } u \text{ has variance equal to zero.}\]
Kalman filter this is proportional to the unconditioned probability density function of the same variables. It is
\[ f(x_k, z_1, \ldots, z_N, u_1, \ldots, u_N, x_1, V_{x_1}, V_{x_2}), \propto f(x_k | z_1, \ldots, z_N, u_1, \ldots, u_N, x_1, V_{x_1}, x_N, V_{x_N}), \]
and finally leads also from the factor graph in Figure 4.3 to the messages for the Kalman smoother. The messages are shown in the following list.

<table>
<thead>
<tr>
<th>Msg #</th>
<th>Kalman Smoother Message Passing Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>[ \bar{m}<em>{X_k}^m = \bar{m}</em>{X_{k+1}} - B_k u_k ]</td>
</tr>
<tr>
<td>7</td>
<td>[ \bar{V}<em>{X_k}^m = \bar{V}</em>{X_{k+1}} + 0 ]</td>
</tr>
<tr>
<td>8</td>
<td>[ \bar{m}<em>{X_k} = A_k^{-1} \bar{m}</em>{X_{k+1}} ]</td>
</tr>
<tr>
<td>9</td>
<td>[ \bar{V}<em>{X_k} = A_k^{-1} \bar{V}</em>{X_{k+1}} A_k^{-T} ]</td>
</tr>
</tbody>
</table>

The similarities to the traditional smoother equations are not that apparent (compare Equations (5)-(8)), but this has one simple reason: The smoother calculation is not yet completed. As it has been introduced in Section 3.1 marginals of the specific variables are calculated by multiplying the forward and the backward message on the specific edge. This is the same as having an imaginary equality node between the forward and the backward message of one edge, therefore the same message update rule as given in Table 3.1.(1) can be used. Note that this message combination rule is not applicable directly to the Gaussian parameterization in mean and variance, but by inverting the variance the message combination e.g. edge \( x_k \) can be expressed as
\[ V_{x_k} = \left( \bar{V}_{X_k}^{-1} + \bar{V}_{X_k}^{-1} \right)^{-1} \]
\[ m_{x_k} = V_{x_k}^{-1} \left( \bar{V}_{X_k}^{-1} \bar{m}_{X_k} + \bar{V}_{X_k}^{-1} \bar{m}_{X_k} \right). \]

These are the parameters for the estimated Gaussian probability distribution of \( x_k \) (not messages anymore). The estimate \( \hat{x} \) of the variable is then the maximum a posteriori (MAP) argument of this density function and reads as
\[ \hat{f}_{x_k} = \frac{\bar{m}_{X_k} \cdot \bar{m}_{X_k}}{\bar{V}_{X_k}} \sim \mathcal{N}(m_{x_k}, V_{x_k}) \]
\[ \hat{x}_k = \arg \max_{x_k} \hat{f}_{x_k} \]
\[ \hat{x}_k = m_{x_k}. \]

In the case of a Gaussian distribution the MAP is directly the mean value \( m_{x_k} \). Note that the messages above are all defined for the Gaussian parameterization of mean \( m \) and variance \( V \). It is also possible to use the parameterization in weighted mean \( Wm \) and weight matrix \( W \). The filter and smoother in this last parameterization are also referred to as information filter and smoother, but in both cases the end results can easily be converted back and forth between the two parameterizations. The final result, which is of interest in
this thesis, is in the format of mean and the variance, i.e. the result of the Kalman filter/smooth, this is why these concepts are consistently called Kalman filter and smoother and the parameterization is only regarded in terms of the implementation.
4 EM-Algorithm using Factor Graphs

This section outlines the central points of this thesis and also includes the two extensions contributed to the topic. First the basic concept of expectation maximization (EM) using factor graphs is given, afterwards the application to linear models is shown on one exemplary derivation for a matrix multiplication node, then the extension to the bilinear model is introduced and the extension for the multivariate variance identification is presented. The underlying general EM-Algorithm is outlined in Section 2.2.

4.1 Basic Concept of EM using Factor Graphs

The approach of EM using factor graphs found its application primarily in the fields of signal processing and coding, cf. [3, 13, 16]. In the field of signal processing the approach was applied in the context of dynamical systems, which is also the context for this thesis. This subsection follows the explanation in [16].

In Section 2.2 the general EM-Algorithm was introduced with an iteration between the two basic steps

\[\text{E-Step: } Q(\theta \mid \theta^{(j)}) = E \left[ \log f(x, y \mid \theta) \mid z, \theta^{(j)} \right] \]

\[\text{M-Step: } \theta^{(j+1)} = \arg \max_{\theta} Q(\theta \mid \theta^{(j)}) \tag{35}\]

where \(x\) stands for the set of all hidden variables in \(f\) and \(z\) is the set of all observed variables (observed data). Together this is denoted as the complete set of data. The objective is to identify the parameter set \(\theta\), that the likelihood of the observed variables is maximized given these parameters. In the sense that the best fitting model given the observed data is determined. The function \(f(x, z \mid \theta)\) is also referred to as global function.

In the rest of this section these steps will be transferred to the environment of factor graphs and from there they will be calculated as messages on the graph.

**E-Step** For a factorizable function a local E-step can be derived. The global E-step in Equation (35) is transformed to a local form, which only deals with local variables connected to the corresponding node in the factor graph. The factorization is assumed to be

\[f(x, z \mid \theta) = \prod_k f_k(x_k, z \mid \theta),\]
where $x_k$ is a subset of $x$. The function $f_k$ can be a node function or groups of node functions (subgraph). Now the global E-step is transformed by these steps

$$Q(\theta \mid \theta^{(j)}) = E\left[\log p(x, z \mid \theta) \bigg| z, \theta^{(j)}\right]$$

$$= E\left[\log \prod_k f_k(x_k, z \mid \theta) \bigg| z, \theta^{(j)}\right]$$

$$= E\left[\sum_k \log f_k(x_k, z \mid \theta) \bigg| z, \theta^{(j)}\right]$$

$$= \sum_k E\left[\log f_k(x_k, z \mid \theta) \bigg| z, \theta^{(j)}\right]$$

$$= \sum_k \eta_k(\theta).$$

In Figure 4.1 the general notation for a local EM factor graph node is given. The function $f_k$ can assume any form of node function involving the set of parameters $\theta$. By taking the $f_k(x_1, \ldots, x_m, \theta)$ with $\{x_1, \ldots, x_m\}$ being the set of local variables for $f_k$, the local E-step function $\eta_k(\theta)$ can be written as

$$\eta_k(\theta) = E_p\left[\log f_k(x_1, \ldots, x_m, \theta)\right]$$

$$= \int \cdots \int p(x_1, \ldots, x_m \mid z, \hat{\theta}^{(j)}) \log f_k(x_1, \ldots, x_m, \theta) \, dx_1 \cdots dx_m,$$

where the underlying conditional distribution $p$ is proportional to the actual probability message of the $\Theta$-edge out of the node $f_k$. The variable $\hat{\theta}^{(j)}$ represents the set of identified parameters from the previous iteration. The distribution $p$ is defined as

$$p(x_1, \ldots, x_m \mid z, \hat{\theta}^{(j)}) = \frac{1}{\gamma} \vec{\mu}_1(x_1) \cdots \vec{\mu}_m(x_m) \cdot f_k \left( x_1, \ldots, x_m, \hat{\theta}^{(j)} \right),$$

where

$$\gamma = \int \cdots \int \vec{\mu}_1(x_1) \cdots \vec{\mu}_m(x_m) \cdot f_k \left( x_1, \ldots, x_m, \hat{\theta}^{(j)} \right) \, dx_1 \cdots dx_m.$$

is a (constant) normalization coefficient. From Equation (37) the EM-message for the dashed edges could be defined, but first Equation (36) is taken one step further, on both sides the
4.1 Basic Concept of EM using Factor Graphs

\[ f_k = f_{k+1} = f_{k+2} \]

\[ \hat{\theta} \uparrow e^{\eta_k(\theta)} \downarrow \eta_{k+1}(\theta) \uparrow \hat{\theta} \downarrow \eta_{k+2}(\theta) \]

\[ \Theta \downarrow \hat{\theta} \uparrow e^{\eta_{k+1}(\theta)} \]

\[ \Theta' \downarrow \hat{\theta} \uparrow e^{\eta_{k+2}(\theta)} \]

**Figure 4.2:** EM-Nodes connected with equality nodes. The upward messages are local EM-messages and the downward arrows symbolize the result of the M-step and are due to the equality nodes equal for all \( k \). It is the set of parameters for iteration \( j + 1 \). The variables \( X_i \) are the local variables of the node functions they are connected to, e.g. \( f_k(X_i, X_{i+1}, \theta) \).

The introduced exponential format for EM-messages, \( e^{\eta_k(\theta)} \), is ideally parameterized as a distribution, for which message passing rules can be formulated. These message passing rules can be used to do the message passing over the equality nodes of the EM-edges (dashed lines). The message passing consist of a forward pass, a backward pass and a marginalization of the resulting forward and backward message per edge. The exponent of this EM message passing is then the global E-step function \( Q(\theta \mid \theta^{(j)}) \) on every \( \Theta' \)-edge. At the same time this is also well motivated as the parameter set \( \theta \) needs to be equal over all connected functions. Remember in Equation (14) the global E-step function \( Q(\theta \mid \theta^{(j)}) \) is directly connected to the lower bound of the objective function of interest. Some typical EM-messages can be found in [16, Table 5.5]. In Section 4.2.2 one example for such an EM-message is derived in detail.

A special case is if the EM-messages are parameterized as Gaussian distribution. For such a Gaussian parameterization, \( \mathcal{N}(\bar{W}_\theta, \bar{m}_\theta) \), the message passing of the equality nodes (EM-edges) leads to a quadratic sum of squares in the exponent of the marginalized EM-messages and the update rules in Table 3.1 can be applied.

After the marginalization the resulting function can be used directly for the M-step.

Of course the proposed connection over equality nodes only makes sense, if all the involved
local $\theta_k$ are equal, in other words satisfy
\[ \theta_k = \theta \quad \forall \ k. \]

If this is not the case the EM using factor graphs offers additional flexibility by some simple extensions to this concept, which allow non-trivial connections, e.g. physical dynamics, between the different $\theta_k$. This was for example introduced in [16] and [3], but will not be discussed further in this thesis.

**M-Step** For the second part of the EM-Algorithm, the M-step (35), the parameterization of the marginalized EM-message, $Q(\theta, \theta^{(j)})$, plays an important role. Remember after the marginalization of the EM-messages the function $Q(\theta, \theta^{(j)})$ is equal on all $\Theta'$-edges. The final maximization depends strongly on the parameterization of this function. If it is parameterized as a common probability distribution with a unique mode, the M-step can be concluded by calculating the mode value, otherwise a regular maximization has to be done. Remark: For a unimodal distribution the mode is equivalent to the maximizer of the random variable (i.e. $\arg\max_{\theta}$). The modes for the distributions, used in this thesis as a parameterization of the EM-message, are listed in Table 4.1. As an example the M-step for a $Q(\theta, \theta^{(j)})$ parametrized as Gaussian, is given as:

\[
\begin{align*}
\theta & \sim \mathcal{N}(m_\theta, W_\theta) \quad \propto e^{Q(\theta, \theta^{(j)})} \\
\theta^{(j+1)} & = \text{mode}(\mathcal{N}(m_\theta, W_\theta)) = \arg\max_{\theta} Q(\theta, \theta^{(j)}) \\
& = m_\theta
\end{align*}
\]

<table>
<thead>
<tr>
<th>Probability Distribution</th>
<th>Mode</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta \sim \mathcal{N}(m_\theta, V_\theta)$</td>
<td>$m_\theta$</td>
<td>$\theta \in \mathbb{R}^{n \times 1}$</td>
</tr>
<tr>
<td>$\theta \sim \Gamma^{-1}(\alpha_\theta, \beta_\theta)$</td>
<td>$\frac{\beta_\theta}{\alpha_\theta + 1}$</td>
<td>$\theta \in \mathbb{R}$</td>
</tr>
<tr>
<td>$\theta \sim \mathcal{W}^{-1}(\Xi_\theta, \rho_\theta)$</td>
<td>$\frac{\Xi_\theta}{\rho_\theta + n + 1}$</td>
<td>$\theta \in \mathbb{R}^{n \times n}$</td>
</tr>
</tbody>
</table>

Table 4.1: Mode equations for the three commonly used M-step distributions

**4.2 EM using Factor Graphs for Linear Systems**

In [3], [16] and [19] different variants for EM using factor graphs for linear model identification are elaborated in detail. The general idea is to extend the Kalman filter/smooother graph in Figure 3.4. For every parameter of the model to be identified, the factor graph
4.2 EM using Factor Graphs for Linear Systems

can be extended with an EM-connection. All node functions using the same parameter to be identified, e.g. for every time step, are connected by $\Theta'_A$-edges and equality nodes, as shown in Figure 4.3. This allows the application of the EM message passing introduced in Section 4.1. For every identification parameter and its node function a suitable EM-message must be found. In the papers mentioned above the most common EM-messages for linear systems are derived. The derivation of the EM-message for a matrix multiplication node as in Table 3.1.(3) will be shown as an example in Section 4.2.2. All other EM-messages used in this thesis can be found in [16, Table 5.5].

Figure 4.3: The factor graph for the EM parameter identification using factor graphs. The original Kalman graph from Figure 3.4 is extended by EM connections (dashed). For clarity only the connections for the identification of the matrix $A$ and the variance $V_W$ are drawn. In the same way connections to for the parameters $C$ and $V_V$ can be added. The matrix $B$ is an exception, the model structure defining $u_k$ as noise free makes it impossible to apply EM using factor graphs for the matrix $B$. Here other identification methods have to be employed.

4.2.1 Algorithm and Schedules

The newly introduced $\Theta$-edges added to the Kalman graph in Figure 4.3 introduce cycles and therefore increase the possibilities for update schedules. This adds on one hand an interesting degree of flexibility for the design of schedules, but on the other hand makes it impossible to guarantee the convergence of the EM-Algorithm using factor graphs in general.

On the EM extended Kalman graph in Figure 4.3 the model parameters e.g. $A, C, V_V, V_W, m_V$ and $m_W$ can be identified (not all needed EM-edged are drawn).
The attained flexibility of local update rules, allows some flexibility in the design of the 
schedule for the identification algorithm, but as stated the convergence needs to be studied 
for each schedule and model/graph separately. One example for a simple schedule is given 
below.

\[ \textbf{while} [\text{Model Error}] > [\epsilon] \textbf{do} \]

\begin{itemize}
  \item \textbf{Update} all Kalman filter/smooth messages \( \forall k \) based on all \( \theta^{(j)}_{\star} \) and calculate the 
    marginals on all edges needed.
  \item \textbf{Calculate} all \( e^{\eta_k(\theta_{\star})} \)-messages for \( A, C, V_V, V_W \) (local E-step)
  \item \textbf{Do} the message passing over the \( \Theta_{\star} \)-edges for all time steps, which leads to a lower 
    bound \( Q_{\star} \) per parameter (global E-step)
  \item \textbf{Maximize} the lower bounds \( Q_{\star} \) per parameter \( A, C, V_V, V_W \), to get all new iterations 
    for \( \hat{\theta}_k \) (M-Step)
  \item \textbf{Set} \( j = j + 1 \)
\end{itemize}

\textbf{end while}

This schedule is close to the concept of the standard EM introduced in [11], but it lacks 
the identification of \( B \) and \( D \) and is only based on local calculations. This means e.g. the 
calculation of \( e^{\eta_k(\theta_{\star})} \) is only based on \( X_k \) and \( X'_k \), which rely on the parameters of the 
previous iteration \( j \). In comparison the original EM in [11] takes for the same calculation 
the new estimate \((j + 1)\) of the other parameters into account. In this thesis only simple 
schedules are used, comparable to the one above, and the study of schedules is left as a 
topic of further research.

### 4.2.2 E-Step-Message of a Matrix Multiplication Node

In this subsection the derivation of local EM-messages, also called \( e^{\eta_k(\theta)} \)-messages, is shown 
on the example of a multiplication node introduced in Table 3.1.(3). The parameter to be 
identified is the vector \( \theta_A \), where the matrix \( A \) is a function of \( \theta_A \). One of these blocks is 
e.g. used in Figure 4.3 for the multiplication node with matrix \( A \). The derivation shown here 
was introduced and elaborated in detail in [3]. For such a multiplication node (Figure 4.4) 
the regular node function can be written as

\[ f(X, Z, \theta_A) = \delta(Z - A(\theta_A) \cdot X), \quad (38) \]

where \( A(\theta_A) \) is the matrix to be identified. Due to the form of the Dirac function \( \delta() \) this node 
function cannot be used directly in the EM-Algorithm. The Dirac function being understood 
as the limit of a Gaussian distribution, with variance going towards zero, leaves no room 
for the maximization step in the expected value, calculated for the local E-step. It defines 
a strict relation between \( X \) and \( Y \) as expressed in (38). On one hand the maximization of 
the expected value as a Dirac function leads to a constant maximizer, and on the other 
hand the message passing over equality nodes is not defined for Dirac functions. See also 
the illustration of the EM-Step in Figure 2.1 to understand how the expected value needs to 
be optimized.

To overcome this problem a new function \( g \) grouping the multiplication node and a Gaussian

![Figure 4.4: Multiplication node function with parameter \( \theta_A \)](image-url)
noise source is introduced. The Gaussian noise source causes the desired uncertainty in the expected value over the new function $g$, therefore it is possible to identify the multiplication node parameter $\theta_A$ with EM. The function $g$ is depicted in Figure 4.5. An existing noise source of the model can be used.

\[
g(X,Y,\theta_A) = \int_V \int_Z \delta(Z - A(\theta_A) \cdot X) \cdot \delta(V - (Y - Z)) \cdot \frac{|V| - \frac{1}{2} (V - m_V)^T V^{-1} (V - m_V)}{\sqrt{(2\pi)^n}} dZ dV
\]

where $\{X,Y,Z,V\} \in \mathbb{R}^n$. The term const. refers to a constant regarding $\theta_A$. The function $g(X,Y,A(\theta_A))$ is used to formulate the E-step message for the matrix multiplication node. According to Section 4.1 the message can be parameterized as being proportional to the Gaussian $\theta_A \sim \mathcal{N}(W_{\theta_0} m_{\theta_0}, W_{\theta_0})$, with $\theta_A \in \mathbb{R}^n$. The local E-step function $\eta(A(\theta_A))$ is the exponent of the proportional EM-message of the form $e^{\eta(\theta_A)}$ and is defined as

\[
\eta(\theta_A) = E[\log(g(X,Y,A(\theta_A)))]
\]

\[
= \text{const.} - E \left[ \frac{1}{2} (Y - A(\theta_A) \cdot X - m_V)^T W_V ((Y - A(\theta_A) \cdot X) - m_V) \right]
\]

\[
= \text{const.} - E \left[ \frac{1}{2} (Y^T W_V Y + A(\theta_A) \cdot X) W_V A(\theta_A) \cdot X + m_V^T W_V m_V - 2 Y^T W_V m_V - 2 (A(\theta_A) \cdot X)^T W_V Y + 2 (A(\theta_A) \cdot X)^T W_V m_V \right]
\]

\[
= \text{const.} - E \left[ \frac{1}{2} (A(\theta_A) \cdot X)^T W_V A(\theta_A) \cdot X \right] - E \left[ (A(\theta_A) \cdot X)^T W_V Y \right] + E \left[ (A(\theta_A) \cdot X)^T W_V m_V \right]
\]

To bring this $\eta(\theta_A)$ function into the right format for message passing two auxiliary functions need to be introduced:

The functions cvect() and rvect() are two functions to bring any matrix into a representation
as a column or a row vector. If the matrix $A$ is

$$
A = \begin{bmatrix}
a_{1,1} & a_{1,2} & \cdots & a_{1,j} \\
a_{2,1} & a_{2,2} & \cdots & a_{2,j} \\
\vdots & \vdots & \ddots & \vdots \\
a_{i,1} & a_{i,2} & \cdots & a_{i,j}
\end{bmatrix}
$$

and

$$
a_{.j} \text{ is the } \ j\text{-th column of } A,
a_{i.} \text{ is the } i\text{-th row of } A,
$$

then

$$
cvect(A) = \begin{bmatrix} a_{.1} \\ a_{.2} \\ \vdots \\ a_{.j} \end{bmatrix}, \quad (40)
$$

$$
rvect(A) = \begin{bmatrix} a_{1.} & a_{2.} & \cdots & a_{i.} \end{bmatrix}. \quad (41)
$$

With these defined functions the matrix $A$ is parameterized by $\theta_A$ as a vector containing all matrix elements,

$$
\theta_A = cvect(A(\theta_A)),
$$

the expected values in (39) can be split up further. The proof of this transformation is given in [3, Appendix C.E]. The $\eta(\theta_A)$ message exponent results in

$$
\eta(\theta_A) = \text{const.} - \frac{1}{2} rvect(A(\theta_A))(W_V \otimes E[XX^T])rvect(A(\theta_A)^T)
$$

$$
- rvect(A(\theta_A))(W_V \otimes I_n)(cvect(E[XY^T]) - cvect(E[X]m_Y^T))). \quad (42)
$$

And finally, under proportionality, the message parameterization as Gaussian for the message $e^{\eta(\theta_A)}$ yields to

$$
\hat{W}_{\theta_A} = W_V \otimes (V_X + m_Xm_X^T)
$$

$$
\hat{W}_{\theta_A} \hat{m}_{\theta_A} = (W_V \otimes I_n)(cvect(V_{XY^T} + m_Ym_Y^T - m_Xm_X^T)). \quad (44)
$$

The calculation of $V_{XY^T}$ based on the messages $\hat{m}_X, \hat{m}_Y, \hat{V}_X$ and $\hat{V}_Y$ can be found in [3, Tab.IV].

Note: Only matrices which appear as a factor between two uncertain variables can be identified with EM and the described EM-rule, i.e. matrices $B$ and $D$ of the standard model used in this thesis cannot be identified.

### 4.3 EM using Factor Graphs for Bilinear Systems

After an overview of the underlying concepts of factor graphs and EM in the preceding sections, this section introduces the central contribution of this thesis: The extension of
the EM using factor graphs to bilinear system identification. The bilinear model structure of interest is

\[
x_{k+1} = A \cdot x_k + F \cdot (u_k \otimes x_k) + B \cdot u_k + v_k
\]

\[
z_k = C \cdot x_k + G \cdot (u_k \otimes x_k) + D \cdot u_k + w_k,
\]

where \( \otimes \) denotes the Kronecker product. This model structure adds the two additional parameters \( F \) and \( G \) to the identification problem, the parameters \( A \) and \( C \) are identified together with these new bilinear parameters, but all the other parameters of the bilinear model are treated the same as in the linear case.

As it will be seen the extension for \( A \) and \( F \) can equivalently be applied to the identification of \( C \) and \( G \). This is why only the first case is described here.

One property of the model structure to be exploited by the new approach is that if the input \( u_k \) is known the bilinear model can be transformed into a linear, time variant systems model. The transformation is based on the transformation of the matrices \( A \) and \( F \) to a matrix \( A_k \), defined as

\[
A_k = A + F \cdot (u_k \otimes I_n),
\]

Of course the transformation holds for the matrices \( C \) and \( G \) in the same way. The linear, time variant model is then

\[
x_{k+1} = A_k \cdot x_k + B \cdot u_k + v_k \quad \forall k \in \{1, N-1\},
\]

\[
z_k = C_k \cdot x_k + D \cdot u_k + w_k \quad \forall k \in \{1, N\}.
\]

To bring the transformation in (45) into a linear format a trivial lemma needs to be introduced. In general the matrix product is not commutative

\[
P \cdot Q \neq Q \cdot P.
\]

for any real matrices \( P \) and \( Q \). By using the vectorized form of the matrices \( P \) and \( Q \) and introducing a Kronecker product with the identity matrix it can be shown that the order of the matrices can be changed:

**Lemma, cf. [21]:** For any real matrix \( P \) and \( Q \)

\[
cvect(P \cdot Q) = (I \otimes P) \cdot cvect(Q) = (Q^T \otimes I) \cdot cvect(P)
\]

Using this lemma the transformation in (45) can be rewritten to the following transformation

\[
A_k = A + F \cdot (u_k \otimes I_n)
\]

\[
= \begin{bmatrix} A & F \end{bmatrix} \cdot \begin{bmatrix} I_n \vline u_k \otimes I_n \end{bmatrix}
\]

\[
cvect(A_k) = J_k(u_k) \cdot cvect([A,F]) \quad \text{with:} \quad J_k = \begin{bmatrix} I_n \vline u_k \otimes I_n \end{bmatrix}^T \otimes I_n,
\]

where \( cvect() \) is an auxiliary function as defined in (40). If \( u_k \) is assumed to be known, this transformation becomes linear. This makes it possible to use the local E-step function of the linear case introduced in Section 4.2.2 and to add a linear transformation with matrix
As illustrated in Figure 4.6, the linear transformation is done by a regular multiplication node function. In this context the extension for the bilinear EM using factor graphs is a linear transformation of the involved message or in fact of its parameters. The properties of the factor graph message passing rules allow a transformation of the upward E-step message, without explicitly inverting the possibly singular matrix $J_k$. Following the message passing rules in Table 3.1.(3), the parameters of the E-step message are

$$\begin{align*}
\widehat{W}_{\theta A,F} &= J_k^T(u_k) \cdot \widehat{W}_{\theta A} \cdot J_k(u_k) \\
\widehat{W}_{\theta A,F} \cdot \widehat{m}_{\theta A,F} &= J_k^T(u_k) \cdot \widehat{W}_{\theta A} \cdot \widehat{m}_{\theta A},
\end{align*}$$

(47)

where $\widehat{W}_{\theta A}$ and $\widehat{W}_{\theta A,F} \cdot \widehat{m}_{\theta A}$ follow from Equations (44). And of course the two matrices $A$ and $F$ follow directly from $\theta_{A,F}$, following from the same message passing and M-step as in the linear case in Section 4.2. With the introduced linear transformation of the parameters for the E-step message the factor graph for the bilinear model identification only changes by one additional linear multiplication block. The Figure 4.6 illustrates the difference to the regular linear case in Figure 4.3.

The EM approach described in this section for the matrices $A$ and $F$, can be used equally for the matrices $C$ and $G$.

### 4.4 Multidimensional Gaussian Variance Matrix Identification

The identification of a Gaussian variance matrix follows the same concept, as for the other identification with EM using factor graphs. The derivation of an EM-message for the appropriate node stands at the heart of this problem. In the case of the variance identification the node of interest is the Gaussian distribution node introduced in Table 3.1.(4). In [16] the identification of scalar variances for Gaussian nodes is introduced using the Inverted-Gamma distribution. This multivariate extension is basically the same, just the message parametrization is changed to fit the condition, that the identified parameter is a positive definite matrix. Therefore not only the calculation of the E-step message changes but also the message passing rules for equality nodes, which are used to connect the same parameters over several time steps, see Figure 4.2.

The node discussed is illustrated in Figure 4.7. It is e.g. used in Figure 4.3 for the identifica-
tion of the variance of the measurement noise $W$. The E-step derivation follows the definition

$$X_k \sim \mathcal{N}(m_X, V_X(\theta_X))$$

$$\Theta_{V_X} \downarrow \hat{\theta}_{V_X}$$

$$\Theta_{V_X} \uparrow e^\eta_k(\theta_X)$$

$$X_k \sim \mathcal{N}(m_X, V_X(\theta_X))$$

**Figure 4.7:** Unknown variance estimation block, $V_X$ is equal to the estimation parameter $\theta_X$, a positive definite matrix

of the local calculation for the expected value in (37). The EM-message, also referred to as $e^\eta_k(\theta_X)$, is calculated from the Gaussian node function $f_k(X_k) \sim \mathcal{N}(m_X, V_X(\theta_X))$ with $V_X(\theta_X) = \theta_{V_X}$ as

$$\eta_k(\theta_X) = E \left[ \log f_k(X_k) \right| z, \hat{\theta}^{(j)}]$$

$$= \log \left( \sqrt{\frac{\theta_{V_X}^{-1}}{2\pi^n}} \right) - \frac{1}{2} E \left[ (X_k - m_X)^T \theta_{V_X}^{-1} (X_k - m_X) \right]$$

$$= \text{const.} + \log \left| \theta_{V_X}^{-\frac{1}{2}} \right| - \frac{1}{2} E \left[ \text{trace} \left( (X_k - m_X)^T \theta_{V_X}^{-1} (X_k - m_X) \right) \right]$$

where $z$ is the set of observed data and $\hat{\theta}^{(j)}$ is the set of parameters from the last iteration, both are used to calculate the expected value over the factor graph. In the last step it was possible to introduce a trace function, because the exponent is a scalar. The trace function is cyclic invariant i.e. $\text{trace}(ABC) = \text{trace}(CAB)$, this allows to reorder the elements in the trace function. Setting this proportional to the log of the Inverse-Wishart distribution introduced earlier yields to

$$\eta_k(\theta_X) = \text{const.} + \log \left| \theta_{V_X}^{-\frac{1}{2}} \right| - \frac{1}{2} \text{trace} \left( (X_k - m_X) (X_k - m_X)^T \theta_{V_X}^{-1} \right)$$

$$\propto \log \left| \theta_{V_X}^{-\left(\rho_{\theta_{V_X}} + n + 1\right)/2} \right| - \frac{1}{2} \text{trace} \left( \Xi_{\theta_{V_X}} \theta_{V_X}^{-1} \right),$$

this makes the parametrization as Inverse-Wishart distribution message clear:

$$\Xi_{\theta_{V_X}} = V_{X_k} + m_{X_k} m_{X_k}^T + m_X m_X^T - 2E[X_k] m_X^T$$

$$\rho_{\theta_{V_X}} = -n$$

The parametrized EM-Message can then be passed as introduced in Section 4.1 following the message passing rules for Inverse-Wishart messages derived in Section 3.2.2.
5 Implementation Details

In this section the implementation of the concepts described up to this point are explained. First an overview over the implementation of factor graphs is given. Later the implementation of EM Messages is discussed and finally a special case for the implementation of MIMO systems is described.

5.1 The Implementation of a Factor Graph

In the development phase of factor graph algorithms it makes sense to write a library of the basic message passing update rules. These include all the Gaussian message update rules in the Tables 3.1, 3.2 and also the other rules introduced in Section 3. This library makes it possible to build factor graph algorithms simply by calling these update rule functions and allows to change the model structure and algorithms more flexible. Another advantage is also that the implementation is less error-prone. As soon as it comes to efficiency, a further simplification of the code can make sense, although this is traded in for the flexibility of the modular structure and distributed calculation capabilities.

Out of the basic functions from such a library the Kalman filter/smoother can be built as independent function to be called before the EM steps are taken. Of course this limits the scheduling in the sense that only full Kalman filter/smoother updates for a certain number of time steps can be done. More nested schedules would then probably require to split up the Kalman filter/smoother function.

In this thesis the implementation of the following time variant model structure is used, it includes the bilinear representation explained in Section 4.3:

\[
\begin{align*}
    x_{k+1} &= A_k \cdot x_k + B \cdot u_k + v_k \\
    z_k &= C_k \cdot x_k + w_k,
\end{align*}
\]

where

\[
\begin{align*}
    x_k &\in \mathbb{R}^{n_x} & \text{System state for time } k \\
    u_k &\in \mathbb{R}^{n_u} & \text{Control input for time } k, \text{ deterministic} \\
    z_k &\in \mathbb{R}^{n_z} & \text{System output for time } k \\
    v_k &\sim \mathcal{N}(m_V, V_V) & \text{Gaussian Process Noise} \\
    w_k &\sim \mathcal{N}(0, V_W) & \text{Gaussian Measurement Noise} \\
    A_k &= A + F \cdot (u_k \otimes I_{n_u}) \\
    C_k &= C + G \cdot (u_k \otimes I_{n_u}) \\
    A &\in \mathbb{R}^{n_x \times n_x} \\
    F &\in \mathbb{R}^{n_x \times (n_u \cdot n_u)} \\
    B &\in \mathbb{R}^{n_x \times n_u} \\
    C &\in \mathbb{R}^{n_z \times n_x} \\
    G &\in \mathbb{R}^{n_z \times (n_u \cdot n_u)}
\end{align*}
\]

This model is represented by the factor graph in Figure 5.1. For \( F = 0 \) and \( G = 0 \) the same factor graph can be used for linear time invariant models.
Figure 5.1: Implemented Factor graph for model structure in Equation (48). Note: $m_V = 0$, $A_k = A + F \cdot (u_k \otimes I_n)$, $C_k = C + G \cdot (u_k \otimes I_n)$ and $B$ is constant. EM-edges are introduced as needed, see also Figures 6.1 and 6.6.
5.2 General Message Implementation

The main message parameterization in the applications of this thesis is the Gaussian distribution, this is why this section focuses on its implementation. In this thesis it is used for all Kalman filter/smoother messages and even for some EM-messages.

In Section 3.2.1 two Gaussian parameterizations are introduced. Both of them have their advantages and disadvantages, e.g. the parameterization in mean and variance (see Equation (20)) is more practical in conjunction with summation nodes. On the other hand for the parameterization in weighted mean and information matrix (see Equation (21)) the calculation over equality nodes is straightforward, but not the summation. These two examples show the conflict for the selection of the appropriate Gaussian message parameterization. In Table 3.2 combined message update rules for both of these conflicting operations are introduced. This allows to calculate the messages over such conflicting nodes in both parameterizations in an acceptable way. In both cases of the combined rules at maximum one matrix inversion is needed. A main reason for the usage of the information matrix parameterization is that the message marginalization, referring to the multiplication of two opposing messages, is easily defined for the weighted mean and information matrix, but not for mean and variance. This can also be seen in the fact that the marginalization leads to the same calculation as having an equality node between the two opposed messages and this cannot be defined directly in mean and variance. Due to this fact mainly the information matrix parameterization is used. This leads in total to less conversions for the implementation of the EM using factor graphs algorithm.

Ideally each parameterization would be used in the appropriate situation, but this would need many conversions on the way through a factor graph. Such conversions always involve the inversion of the variance/information matrix and can lead to numerical problems. For example if a variable has a very small variance, this leads to the inversion of a badly conditioned matrix or in the worst case of a singular matrix. This is especially a problem if sections of a factor graph are repeated many times, e.g. for each time step in the Kalman filter/smoother. Therefore bad inversions of matrices would cumulate over the number of repetitions and errors are propagated with the messages. Along these lines do the next subsections explain the two main implementation cases, Kalman filter/smoother and EM-messages.

5.2.1 Implementing Messages for the Kalman Filter/Smoother

As there exist numerous possibilities for combining different parameterizations, only the version which is finally used is explained. The used concept is focusing on the parameterization with weighted mean and information matrix as the marginalization of messages is only defined for the information matrix parameterization. The model used is shown in (48) and the corresponding factor graph is shown in Figure 5.1. The implementation of the messages ① to ⑩, which are labeled in the graph, is explained in detail:

Message ① is the message backward on the edge $Y_k$. It is a combination out of the known measurement value and the uncertainty added by the measurement noise $W$. This noise is assumed to have a mean value of zero. This allows a simple derivation of ①. The message coming from $W_k$ is parameterized as mean and variance and for the further propagation the variance from this message has to be inverted. With the derived information matrix $\overrightarrow{W}_{W,k}$ and the known measurement $z_k (\overrightarrow{W}_{Z,k} = 0)$ the message going backward on edge $Y_k$ is calculated by the product $\overrightarrow{W}_{Y,k} \overrightarrow{m}_{Y,k} = \overrightarrow{W}_{W,k} \cdot z_k$ and by $\overrightarrow{W}_{Y,k} = \overrightarrow{W}_{W,k} \cdot z_k$.
This adds one matrix inversion per time step for this message.

Message ② is the message forward on edge $X_k$. It is calculated straight forward in weighted mean and information matrix using the combined equality / matrix multiplication update rule introduced in 3.2.(1). This adds one matrix inversion per time step for this message.

Message ③ results from a rather simple forward matrix multiplication message update (Table 3.1.(3)) for edge $X'_k$, but in weighted mean and information matrix parametrization it needs the matrix $A_k$ to be regular. In mean and variance this message calculation would be straight forward, but the conversion would add two matrix inversions, which is what is to be avoided if possible. In [19, Tab.5] special message update rules are introduced, which can be used if the matrix $A_k$ is singular. Only the matrix $A_k$ needs to be inverted (if nonsingular).

Message ④ is combining the state information in $X''_k$ and the process noise from edge $V_k$. To calculate it the combined message update (Table 3.2.(2)) can be use, by introducing an identity matrix multiplication node on the edge $V$. This calculation adds one matrix inversion per time step.

Message ⑤ is the first message of the next time step $k+1$. It is the result of the sum of the state variable $X''_k$ and the known input $u_k$. Due to the fact that the $u_k$ is known it is enough to update the weighted mean. Unfortunately this is still not a nice operation, due to the parameterization in weighted mean and information matrix. The calculation is $\overrightarrow{W}_{X',k+1} m_{X''_k,k+1} = \overrightarrow{W}_{X'',k} \left( u_k + \overrightarrow{W}_{X'',k}^{-1} \left( \overrightarrow{W}_{X''_k,k} m_{X''_k,k} \right) \right)$ and $\overrightarrow{W}_{X',k+1} = \overrightarrow{W}_{X''_k,k}$. This adds one matrix inversion per time step to the total.

The calculation of the messages ⑥ to ⑨ follows the same principles like the five mentioned messages above. For ⑧ it is even easier, because the matrix multiplication node does, in weighted mean and information matrix parameterization, not need any inversions. With the remarks made above all needed messages for the Kalman Filter/smoother can be calculated. Note: For summation nodes the signs (+/−) are not the same for forward and backward message passing. The appropriate signs can be found in Table 3.1. Finally to calculate the marginalization on one edge the equality node update rules can be used. Equality nodes and marginalization are both based on the multiplication of the messages, this is why an equality node between the two opposed messages on one edge would be the same as the marginalization for this edge, i.e. for the edge $X_k$ that $\overrightarrow{W}_{X,k} = \overrightarrow{W}_{X,k} + \overrightarrow{W}_{X,k}$ and $W_{X,k} m_{X,k} = \overrightarrow{W}_{X,k} m_{X,k} + \overrightarrow{W}_{X,k} m_{X,k}$. The mean is then $m_{X,k} = \overrightarrow{W}_{X,k} W_{X,k} m_{X,k}$. c.f. [19]

The total number of matrix inversions depends on the number of mean values of variables, which are needed for the application, e.g. which parameters are identified with EM. Already nine matrix inversions per time step are needed for the Kalman filter/smoother following the concept described above and if only the mean of $X_k$ is derived as result of the Kalman filter. Another way to reduce the number of matrix inversions further, would be to use additional message passing update rules introduced in [19]. With these rules combinations without any matrix inversion are possible. But no numerically stable implementation with these rules was found in the literature.

An important role is also played by the initialization of the forward messages on the left ($k = 1$) and the backward messages on the right ($k = N$) end of the Kalman factor graph. In general a generic message with zero information can be used for both cases, i.e. $\overrightarrow{W}_{X',1} = 0$ and $\overrightarrow{W}_{X',N} = 0$; $\overleftarrow{W}_{X',N} m_{X',N} = 0$ and $\overleftarrow{W}_{X',N} = 0$. To improve the numerical robustness, a very small information matrix could be used instead, otherwise depending on the noise level in the
system, the initialized variance equal to zero can lead to numerical issues. A special case is
the linear time invariant Kalman factor graph, where also the variance of the steady state
Kalman filter could be used for the forward initialization [22]. The backward message would
still need to be initialized by the generic message above. All these initializations were suc-
cessfully tested with the mentioned restrictions. The initialization with generic messages is
finally used for the bilinear implementation in this thesis.

5.2.2 Implementing EM-Messages

Three steps are related to the implementation of EM-messages.

1. Calculating the EM-message ($e^{\eta \theta}$).
2. Passing messages over equality nodes.
3. Marginalizing of messages on the $\Theta'$ edge.

For the notation see the example of an EM factor graph in Figure 4.3. The first step, calcu-
lating the EM-message, can lead to some numerical issues, especially as marginalized local
variables are used in the EM-node and as mentioned in the last section the marginalization
can be rather troublesome due to the inversion of the information matrix. Especially for sys-
tems with low noise levels, it can result in badly conditioned information matrices. Another
sometimes numerically hard step is to calculate the cross-correlation between some variables
on the graph, as it is needed e.g. for the EM-message to identify the matrix parameter of a
multiplication node, see Equation (47). For the cross-correlation several inversions of infor-
mation matrices are needed, which as mentioned can be a badly conditioned problem. In real
world data tests this was actually one of the limiting steps so far.

The next step is the messages passing over the equality nodes. (Other nodes for EM-messages
are not discussed here.) If parameterized as Gaussian, the equality node message passing is
rather simple, especially as EM-messages are in general parameterized with weighted mean
and information matrix. Also for the non-Gaussian messages introduced in this thesis the
update rules for equality nodes are straight forward and will not be discussed further.
Finally it remains the marginalization on a $\Theta'$ edge. This can again be a numerically challeng-
ing step, as it involves inversions of information matrices, but there seems to be no obvious
way to circumvent this step.

5.2.3 Implementing MIMO Systems

In general the implementation of multiple input multiple output systems (MIMO) is as de-
scribed above, but for the combined update rules of Gaussian messages (Table 3.2) exists
sometimes a way to replace the matrix inversion with a scalar inversion. If the multiple inputs
are known, i.e. variance is zero, or are independent, i.e. diagonal variance matrix, the calcu-
lation of the summation node with the states (Messages $\tilde{\eta}^\theta$, $\tilde{\eta}^\theta$) can be simplified in terms of
matrix inversions. Instead of adding the input vector, e.g. $u_k$ as a vector to the state, each
vector element (input, scalar) is added separately. If this is done with help of the combined
message update rule (Table 3.2(2)), only scalar inversions are needed. The inverted term in
these update rules has the same dimensions as the message on the edge with the multiplier
matrix $B$. On factor graphs this can be interpreted as having multiple inputs in a row and
adding them separately. Figure 5.2 illustrates the changed input section of the graph.
If e.g. noise inputs are independent of each other, meaning that they have variance in diagonal form, then this approach can also be used to add the noise. For the implementation of the special case that the input is known, i.e. the variance of the input $u$ is zero, the combined update rule used for the improved MIMO Implementation (Table 3.2.2) would fail. This is caused by the inversion of the variance, which is not defined in this case. A minor noise level has to be assumed, several orders of magnitudes smaller than the general noise level in the data (the influence is in this way negligible), to apply the introduced MIMO implementation. This is a way to avoid problems with badly conditioned matrices for this special case of a known input.

Figure 5.2: MIMO Implementation of a summation node. Graph (a) shows the original summation node for a MIMO input. Graph (b) is the improved MIMO implementation, where $b_{i,k}$ represents the $i$-th column of $B$. This improvement can be used, if $u$ has a diagonal variance matrix.
6 Results

The results presented in this section will give a proof of concept for the new methods of the bilinear extension to EM using factor graphs, introduced in Section 4.3, and the identification of variance matrix with EM using factor graphs, see Section 4.4. Throughout this section one basic model is used to generate the identification data. It is introduced in Appendix A. All the data used for the identification is generated based on these system parameters unless stated differently. A known input signal (generated using a Gaussian generator to ensure sufficient excitation of the system) is used for $u_k$. The actual noise sequences were generated based on the specified normalized Gaussian noise levels for each result.

Normalized Gaussian noise in this context means that if e.g. the process noise is 100% and every other excitation (input) of the system is 0, then the output has a standard deviation of 1. Mean of the noise signal is in any case zero and the variance is set to satisfy the percentage of the normalized Gaussian noise level. The numerical normalization gains were determined empirically for the model parameters defined in Appendix A.

The same normalization procedure was used for the input signal $u_k$, this was necessary as the analysis of the regular normalization for a bilinear system was beyond the scope of this thesis. In this sense the normalization factor for the input was determined empirically to have a steady state output of one if the input $u$ is one. Due to this normalization is a mean squared error of 0.01 a 1% relative error.

6.1 Identifying the Bilinear Term, Matrices $A$ and $F$

In this Section the identification of the two matrices $A$ and $F$ is presented. The identification of other parameters together with $A$ and $F$ is discussed in Section 6.3. First some exemplary results are shown, then the convergence with different noise levels is discussed. The factor graph used for this identification is shown in Figure 6.1. It is based on the Kalman filter/smoother graph introduced in Section 3.3 and it uses the identification concept for the bilinear matrices $A$ and $F$ explained in Section 4.3. This graph can be used for the bilinear system because in the identification process the input $u_k$ is always known and therefore the bilinear structure can be represented as a linear, time variant system. This transformation was discussed in Section 4.3.

For the identification a simple schedule which alternates between a Kalman message passing run (solid line in the factor graph) and a EM message passing run (dashed line in the factor graph) for the parameter matrix $[A \ F]$ is used.
Figure 6.1: This factor graph is used for the identification of the bilinear parameter matrices $A$ and $F$. It is the combination of the Kalman filter/smoother factor graph (Figure 3.4), the factor graph for the EM identification of a multiplication node (Figure 4.5) and the factor graph components of the bilinear extension (Figure 4.6).
6.1 Identifying the Bilinear Term, Matrices $A$ and $F$

6.1.1 Exemplary Results - Low/High Noise Level

The exemplary results are shown for two cases using the basic model with two different levels of noise to generate the identification data. Beside the properties of the basic model (Appendix A) the two cases are specified in the following way:

**Low Noise Level Case**
- Normalized Gaussian process noise is 1% - $V$
- Normalized Gaussian measurement noise is 1% - $W$

**High Noise Level Case**
- Normalized Gaussian process noise is 10% - $V$
- Normalized Gaussian measurement noise is 5% - $W$

For this identification a data set of 1500 time steps was used and the iteration horizon was chosen to show a converged result for both noise level cases. In total 1000 iteration steps were performed.

The identification was initialized with random matrices for $A$ and $F$, which in the first few iterations even lead to an initially unstable output behavior. In the upcoming paragraphs the result for both cases is discussed based on different figures. Note: The presented example is an especially hard problem, as the initial parameter set does not converge directly to a stable system behavior. With a purely stable system behavior over all iterations a converged parameter set would be achieved with less iterations. The presented example was chosen to visualize the nice property of the EM algorithm to converge quickly from a far off initial parameter set, usually without getting stuck in a local minimum on the way.

**Convergence**  Plots concerning the mean squared error (MSE) convergence are shown in Figure 6.2, one for each noise case. The error shown is the difference between the generated and the simulated output. For the identification set the simulation was done under the assumption that all noise inputs, which led to the generated identification data set, are known. Due to this setup the mean squared output error (MSE) is independent from the specific noise values in the simulation data set and compares only the behavior of the system based on the parameters $A$ and $F$. For the validation set a different input set ($u_k, v_k$ and $w_k$) was used, but again the same for output generation and simulation. The validation shows how good the identified system is for another data set than the identification set.

It is eye catching how quickly EM reduces the initial error. After less then 42 iterations the initially unstable system is in both cases identified to a stable system with an error below 1% to the measured output. For the higher noise level case the 1% margin is already reached at iteration 24. This quicker convergence for the high level noise case is a clue showing that the performance of EM strongly depends on the noise levels in the system, Section 6.1.2 elaborates further on that phenomena. In general it can be said that after the fast convergence at the beginning it is usually slowing down dramatically. This is typical for EM, cf. [12], which is only strong in the identification of the general dynamics. For the identification of the details the convergence is very slow or in the higher noise level case even stops. There the error stagnates on the level of $10^{-3}$, but compared to the noise in the system this level of identification error is already quite a good result.

The ripple, which is apparent in the first iterations, is a result of the local EM approach, but its concrete origin is not studied further in this thesis.
Figure 6.2: Convergence of the mean squared error of the output for two data sets. The solid line stands for the identification data set; the dashed line stands for a validation data set.
Figure 6.3: Step response for the two noise cases of the identification of the matrices $A$ and $F$. 
Figure 6.4: Impulse response for the two noise cases of the identification of the matrices $A$ and $F$. Only the noise free signals are shown.
Step and Impulse Response. In this paragraph the identified matrices $A$ and $F$ are tested with a step and an impulse response. All other parameters are assumed to be known and are set to their correct value. The Figures 6.3 and 6.4 show the corresponding results for both noise cases introduced earlier. Of course step and impulse response are directly connected, but for better readability both plots are given. As an example judging only from the impulse responses would lead to the conclusion that the identification is flawless in both cases, but the step response shows, that at least for the high level noise case the steady state gain is not identified properly. For these simulations the noise was set to zero. This improves the visibility of the dynamics. To give an impression of how much noise was in the original identification data, the noisy step output for the original system is shown in the step response plot (dotted line). The distinction between noise cases has to be made anyway, because the identification data, used to find the tested system matrices $A$ and $F$, was disturbed by noise of the different levels. The identified results correspond to the results presented above with the MSE plots. For both noise cases the impulse response of the identified system shows a very close behavior to the original system. In the low noise level case almost no differences to the original system are visible, for the high noise level case the differences are slightly more visible, but the identification still seems to be good. This shows that the fast dynamics are identified well. For the step responses the result looks a bit different. In these plots a deviation of the steady state is visible. Especially for the high level noise case the steady state is $2 - 3\%$ off. There are two possible explanations for this problem either a non-satisfying deterministic input signal was used, which does not cover the whole bandwidth and range of the nonlinear system dynamics (e.g. zero mean input signal), or a deficiency of the identification algorithm itself. Even tough the first is more likely it was not possible to verify it in the scope of this thesis.

6.1.2 Convergence Depending on Noise Levels

There are two properties of convergence, which will be discussed briefly in connection with the noise levels in the system. First is the identification output error (MSE) for the converged solution and second is the number of iterations needed to achieve a certain error level. The study of these topics in this thesis is limited to the extent of some examples. Figure 6.5 illustrates the relevant results for this section. All identification runs for these results were solved under almost the same conditions as the results in Section 6.1. As before only the matrices $A$ and $F$ were identified, but the noise levels of the identification data were altered as illustrated. For a better readability, the initial parameter set was chosen this time to be a stable system, but was once found randomly (for all simulations the same). The unstable initial parameter set as used before has the consequence to increase the occurrence of numerical issues and would therefore give a less coherent picture for these plots. Note: The noise itself is not reflected directly in the MSE, as the same input sequence ($u, v$ and $w$) was used for the generation and the simulation of the data sets.

In Figure 6.5a it is illustrated how the finally achieved MSE is related to the noise levels of the identification data. Two lines can be drawn to limit the area of good MSE results. The first is the line, which is orthogonal to the axis of process noise ($\sigma_v$). It shows that the MSE increase proportional to the logarithmic scale of process noise. In other words for an increasing process noise level the identification gets harder up to a level, where the identification is useless or the algorithm even diverges (outside of the figure, for $\sigma_v \gg 100\%$). The second line which can be identified from the plot gives an affine line for increasing measurement
(a) Final mean squared error of the output (MSE) for identifications with different noise levels. The gaps in the 3D surface are noise levels, where the algorithm diverged or was unstable.

(b) Number of iterations for a maximum identification error (MSE) of 0.1%. Each bar represents one simulation, every Simulation with 0 iterations did not reach an error level below 0.1%.

Figure 6.5: Convergence study for different noise levels.
6.2 Variance Matrix Identification

noise ($\sigma_w$) and decreasing process noise ($\sigma_v$). It can be seen in the noise level space as a
diagonal threshold for useful and useless MSE levels. On the bad side of this threshold no
identification is possible, because the identification stagnates at the initial MSE level (red
area in the plot, at an MSE level of $10^{-0.1}$). This can be explained by looking at the lower
bound function in Figure 2.1. For very low process noise level the identification with EM gets
impossible, because the lower bound function gets too narrow. In the limit, for zero noise, the
lower bound function would be a Dirac function and no maximization for the M-step would
be possible anymore. The measurement noise $\sigma_w$ makes the whole situation just worse at
higher process noise levels already, because it obfuscates the information in the data. This is
why the threshold line runs diagonal at the left end of the plot. In general it can be said that
the EM algorithm using factor graphs can be used for the whole area inside the green/blue
shading. Outside of this area other identification methods have to be employed. As a remark
it is to say, that for noise levels outside of the range of the plots in Figure 6.5a the probability
of numerical problems increases. The too high or too low noise levels cause badly conditioned
information or variance matrices, which cannot be inverted, this makes the algorithm to fail.
The plot (b) in Figure 6.5 shows the number of iterations needed to achieve an MSE below
0.1% for different noise levels. The results used for this plot are the same as for the plot
(a) and therefore they can be compared directly. Each column on this plot represents one
identification result for the specified noise levels. Note that all results which never achieved
an MSE level below 0.1% are marked with white squares. In this plot the same two lines can
be drawn as for the other plot. The borders between the results, which reached the desired
MSE level and the ones which did not reach it, draw these lines. On top of that it can be seen
that the further away the identification run is from the diagonal threshold line mentioned for
the other plot, the smaller is the number of iterations needed to achieve the desired error
level. This means for higher process noise levels the algorithm converges faster, of course at
a certain point the final error level gets too high. The same behavior can also be seen in the
second noise case of Section 6.1. One possible explanation is that the lower bound function
defined for EM gets wider for more uncertainty in the system. Therefore the lower bound
function allows bigger iteration steps for the maximizer. To support the understanding the
reader is again referred to Figure 2.1, where the lower bound function is visualized.

6.2 Variance Matrix Identification

With this section the approach proposed in Section 4.4 is discussed. The multivariate Gauss-
sian variance matrix, e.g. of the process noise $v$, is identified based on the Kalman filter/smoothen factor graph to calculate the local variables needed for the EM part. The
complete factor graph for this section is shown in Figure 6.6. It is used to identify the vari-
ance of the Gaussian process noise. The same could be done for the measurement noise or
other Gaussian noise node functions in a factor graph.

In general this method would allow to identify full variance matrices, i.e. of dependent multi-
variate noise signals, but in the implemented framework of this thesis this could not be tested.
It was successfully tested for the identification of variance matrices in diagonal form. In gen-
eral it is to say that even though the process noise in the system is technically independent
(variance in diagonal form), the identification leads to a variance matrix with off diagonal
elements. This appearing dependency seems purely numerical and is from the perspective of
the original system parameters a numerical issue. If a prior knowledge about the structure
of the variance matrix, e.g. diagonal (independent noise processes), is available, a constraint
can be introduced. This constraint leads to an improved behavior and the original variance
matrix is more likely to be found. A basic alternating schedule was used for this identification, i.e. for each iteration one Kalman filter-smoother message passing and one EM using factor graphs message passing is done. For this schedule the proposed constraint is applied to the resulting variance after each EM message passing and before the variance is used in the next Kalman filter-smoother message passing run.

The approach was tested and converged with and without constraint in a small number of iterations ($15 - 20$). With the constraint enforcing a diagonal form, the resulting variance matrix was less than 10% of the original noise variance. Without constraint the identification has too many degrees of freedom and the variance includes all the numerical dependencies between the noise signals. This is an issue which also appears in standard EM. It was for example an issue for a problem of factor analysis in [14].

\[
\begin{align*}
A_k &= X_k' + NW_k + Z_k, \\
C_k &= Y_k + NW_k + W_k, \\
A_{k+1} &= X_{k+1}' + NW_k + Z_{k+1}, \\
C_{k+1} &= Y_{k+1} + NW_k + W_{k+1}.
\end{align*}
\]

Figure 6.6: Factor graph for EM to identify a multivariate variance matrix of a Gaussian distribution. (Figure 4.6).

### 6.3 Combined Identification

The identification methods used in the last two sections can also be combined to identify the bilinear matrices $A$ and $F$ together with the variance of the process noise $V$. Further the end goal would be to identify all parameters of the model together with one factor graph and all needed EM connections. The biggest obstacle for this goal is that every parameter identification (EM-connections) introduces new cycles into the factor graph. In Section 3.1 it was explained that factor graphs with cycles are not guaranteed to converge and that, to achieve the convergence, possibly special update schedules can be needed. In this thesis the focus was not on such update schedules, therefore the combined identification is limited to
6.3 Combined Identification

simple examples.
The combination of the two identifications explained in the last two sections converges on a basic alternating schedule. This means for each iteration one Kalman filter/smooother message passing run and one EM message passing run are done. While the EM message passing is done for the identification $A, F$ and $V_f$ at the same time. For this combined identification the factor graphs from the Figures 6.1 and 6.6 are combined into one factor graph, meaning one central Kalman factor graph and EM-connections for the identification of the bilinear term and the process noise variance matrix are used. This combined factor graph already includes several cycles, which would allow different update schedule concepts. Using the same problem setup as in Section 6.1 and adding the identification of the process noise variance, it converges with a basic schedule, despite all the cycles in the graph. In Figure 6.7 the exemplary MSE convergence for the high level noise case including the variance identification is shown. The variance identification is not directly visible in the MSE plot, but as it can be seen by comparing it to the Figure 6.2b, the convergence of the error is slightly different. This is the case as the variance identification has an influence on the Kalman filter/smooother run on the factor graph and therefore influences the identification of the matrices $A$ and $F$. Surprisingly the convergence of the MSE at the beginning is even better, but on the other hand, additional peaks are visible in the MSE plot. These are possibly the result of the uncertainty of the variance identification. The initial value for the variance was chosen to be above the real variance, due to this the noise in the estimated data from the Kalman filter is higher and therefore the identification convergence is improved, cf. Section 6.1.2. The variance identification itself converges thanks to the constraint to a diagonal matrix with a margin of several percent to the correct value.

![MSE Identification and Validation Data](image)

**Figure 6.7:** Convergence plot for the MSE of the output; result of the combined identification of $A, F$ and $V_f$.

If additionally the matrix $C$ is added to the identification problem, the convergence for this example was no longer given. The identification of the matrix $C$ is done with simple matrix
multiplication EM-messages as derived in Section 4.2.2 and adds another connection over EM-edges. Therefore it increases the complexity of the cycles in the graph. For this more complex identification schedules have to be elaborated, but as stated this is beyond the scope of this thesis.
7 Summary and Outlook

Summary It was the objective of this thesis to find an extension for the known concept "EM using factor graphs" towards the identification of bilinear systems. The need of an adequate system identification for bilinear systems was motivated primarily by insufficient thermal models for building control. The main advantage of the approach of EM using factor graphs is the modularity and the graphical representation of factor graphs. Another advantage of factor graphs is that all the calculations on the graph are done locally. This fact allows a distributed implementation of the factor graph calculations without losing the modularity and the simplicity of the approach. At the same time the expectation maximization is known to be a robust identification algorithm, used for many statistical applications.

The bilinear extension presented in Section 4.3 is an answer to this objective, which maintains the nice properties of the underlying concepts. It is a modular extension and can be combined with any existing EM using factor graphs approach, without changing more than the local environment of the bilinear part in the factor graph. For the proposed approach no new node function had to be introduced. Previously known structures, such as the linear multiplication node, the equality node and as a central part the EM-message for the identification of a linear multiplication node parameter, are reused. This guarantees that it is compatible with other EM using factor graphs concepts.

In Section 5 the general implementation of factor graphs was discussed, which is also valid for the implementation of the bilinear concepts.

In an exemplary setup the implementation of the new concept was tested with a numerical example. These results are summarized in Section 6, where studies related to the various noise levels in data sets are shown. As a limiting factor the studies in this thesis base only on these exemplary results. In these results it was found that the convergence of the algorithm is strongly connected to the uncertainty in the data sets. If the uncertainty is too low the algorithm does not converge and for people, not familiar with EM, it is even more surprising that the algorithm converges faster if more uncertainty is present in the data set. This behavior was also demonstrated by the results of this thesis.

Finally the method was tested to combine it with the identification of other model parameters on the same factor graph. The limits of such combinations were found quickly and this brought up the question about the appropriate schedules for the identification with such factor graphs.

Furthermore it is to mention that the bilinear concept is not limited to the standard structure presented in this thesis. The bilinear identification approach can be combined with any imaginable factor graph structure that is applicable for the usage with the introduced concept of message passing rules. This could be in the form of additional inputs or a second level of unknown states, the general frame work is explicitly not limited to a state space model and the calculation rules stay the same as long as the standard node functions are used. It will be interesting to see if this novel way, of treating bilinear models in factor graphs, also finds its way in other factor graphs algorithms beyond EM for system identification.

A side product of this thesis was an EM approach for the identification of Gaussian multivariate variance matrices. It was presented in Section 4.4 and is an extension to the concept of scalar Gaussian variance identification presented in [16, Section 5.6]. The tests of this method were limited to the special case of diagonal variance matrices. For this case it was found that additional constraints are needed to achieve satisfiable results.
**Conclusion**  The proposed bilinear approach is feasible for the identification of the bilinear parts of the desired model structure. The implementation shows that it is applicable in the separate identification of the bilinear matrices, and it shows an acceptable region of good convergence in terms of the data uncertainty levels. The method shows a fast convergence in the first few iterations especially from far off initial parameter sets (large region of attraction). On the other hand the convergence for the last few percents of accuracy is relatively slow. In terms of robustness did the identification of more than the bilinear parameters not satisfy the expectations for an EM approach. The reduced robustness is most probably related to appropriate message passing schedules, which are beyond the scope of this thesis.

**Outlook**  One of the next steps of research connected to this bilinear extension is to test it with real world measured data and to verify the convergence findings. Further the question of applicable message passing schedules for the identification with other parameters on the same factor graphs has to be answered. Ideally a widely applicable concept to find such feasible schedules would be found.

Another issue is that alternative identification methods for the two matrices $B$ and $D$ of the basic model structure need to be incorporated for a combined approach including all parameters. This is due to the fact that EM is not feasible for parameters, which enter the model only through known non-stochastic signals. In the case of these matrices $B$ and $D$ the problem is that they are multiplied with the known input $u$. The missing uncertainty in $u$ lets the lower bound (see Figure 2.1) for the identification of these matrices become a Dirac function for which EM is not applicable. Therefore another identification method needs to be used for these matrices.

Another field of research would be to combine EM using factor graphs with other identification methods. This could improve the convergence for the last few percents of accuracy. One possible approach could be to use a method based on steepest descent. In the literature concepts exist to do steepest descent methods using factor graphs, cf. [5]. The jointly used frame work o factor graphs could simplify the handover between the first and the second part of the identification process.

Further a more detailed study of the convergence of EM using factor graphs in general could be used to analyze its competitive position against other methods. In this context it is not to forget that factor graphs have a clear advantage against traditional methods in terms of modular flexibility, purely local calculations and graphical representation.

The already acceptable range of good convergence could possibly be extended by the use of a concept called Hybrid EM using factor graphs, which was introduced in [6, Chapter 4]. It would possibly make EM using factor graphs feasible for data sets with even less uncertainty. If further analysis of the convergence gives satisfying results, an online implementation of the identification could be a further development of the approach. This would be especially interesting for systems with fast changing dynamics, as the algorithm has shown to converge quickly from far of initial conditions. The modularity would in this case allow to change the model structure on the running system without changing the basic equations as for such changes only the schedule for the message passing would need to change.

A further topic of research for the identification of Gaussian multivariate variance matrices is the detailed verification of the method for fully occupied variance matrices.
A Appendix: Numerical Example for a Bilinear Model used for the Results

For all the results in this thesis one numerical example of a bilinear model was used. The model structure is

\[
x_{k+1} = A \cdot x_k + F \cdot (u_k \otimes x_k) + B \cdot u_k + v_k
\]
\[
z_k = C \cdot x_k + G \cdot (u_k \otimes x_k) + w_k.
\]

The model parameters and variables are defined as follows:

- \(x_k \in \mathbb{R}^{n_x}\) System state for time \(k\)
- \(u_k \in \mathbb{R}^{n_u}\) Control input for time \(k\), deterministic
- \(z_k \in \mathbb{R}^{n_z}\) System output for time \(k\)
- \(v_k \sim \mathcal{N}(0,V)\) Gaussian Process Noise
- \(w_k \sim \mathcal{N}(0,W)\) Gaussian Measurement Noise
- \(n_x = 2\) Number of states
- \(n_u = 1\) Number of deterministic inputs
- \(n_z = 1\) Number of outputs
- \(A = \begin{bmatrix} 0.5 & 1.1 \\ 0.2 & 0.3 \end{bmatrix}\)
- \(F = \begin{bmatrix} 0.6 & 0.05 \\ 0.1 & 0.2 \end{bmatrix}\)
- \(B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\)
- \(C = \begin{bmatrix} 1 & 1 \end{bmatrix}\)
- \(G = \begin{bmatrix} 0 & 0 \end{bmatrix}\)
- \(u_k \sim \mathcal{N}(0,(u_{norm} \cdot I)^2)\) Deterministic normalized Gaussian input, \(u_{norm} = 0.57\)
- \(v_k \sim \mathcal{N}(0,(\sigma_V \cdot v_{norm} \cdot I)^2)\) Normalized Gaussian process noise, \(v_{norm} = 0.275\)
- \(w_k \sim \mathcal{N}(0,(\sigma_W \cdot w_{norm} \cdot I)^2)\) Normalized Gaussian measurement noise, \(w_{norm} = 1\)
- \(\sigma_V\) Process noise level in percent \(\propto\) standard deviation
- \(\sigma_W\) Measurement noise level in percent \(\propto\) standard deviation

The parameters of the noise processes are defined for each problem separately.
References


