Modeling and Control of Static and Dynamic Systems with Bayesian Networks

Logos Verlag Berlin 2004

Submitted to the

Technische Fakultät der
Universität Erlangen–Nürnberg

in partial fulfillment of the requirements for
the degree of

DOKTOR–INGENIEUR

from

Rainer Deventer

Erlangen — 2004

As dissertation accepted from the
Technische Fakultät der
Universität Erlangen–Nürnberg

Date of submission:	27.10.2003
Date of graduation:	13.1.2004
Dean:	Prof. Dr. rer. nat. A. Winnacker
Reviewer:	Prof. Dr.-Ing. H. Niemann
	Prof. Dr.-Ing. W. Schweiger

Acknowledgments

This thesis was written during my employment at the chair for pattern recognition as a member of the special research center 396 'Robust, shortened process sequences for lightweight sheet parts'. I would like to thank everyone who contributed to the success of this thesis.

Particularly I would like to thank Prof. Niemann for stimulating discussions and for his constructive criticism. I also wish to express my gratitude to Prof. Schweiger who acts as second reviewer. Both reviewers supported me very much by writing quickly their reviews. Furthermore I would like to thank Prof. Ehrenstein and Prof. Geiger for the good cooperation within the special research center.

Also the good working atmosphere at the institute for pattern recognition was very helpful. Prof. Denzler provided me with helpful comments and looked carefully through the manuscript. Additionally he volunteered to write the preface for this thesis. Mr. Popp, the technician at the institute, assisted me in all questions concerning the hardware.

Additionally I am indebted to my friends and colleagues Mr. Martino Celeghini, Prof. Joachim Denzler, Mr. Luis Fernández Díaz, Dr. Dorthe Fischer, Dr. Oliver Kreis, and Ms. Alexandra Weichlein for carefully reading my thesis.

Finally I have to thank my parents who supported me during my complete development. I severely regret that it was not granted to my mother to see me finishing my doctorate.

Contents

Preface

Todays modern manufacturing processes face shorter production cycles, demand for more flexibility and higher quality of the products. This results in an increased need for new methods in the area of modeling and controlling of process chains, that allows self-adaption as well as integration of expert knowledge.

This is the focus of the book at hand. It is one of the first books that merges techniques from classical control theory and modern, probabilistically motivated artificial intelligence to develop new methods for modeling and adaptive control of dynamic processes. The key element is a Bayesian network, that allows the explicit modeling of dependencies between events. Thus, expert knowledge can be easily integrated. However, in this framework also automatic detection of the dependencies is possible. This makes Bayesian networks perfectly suited for modeling and control of manufacturing processes: during the first design expert knowledge can be used, while in service the parameter can be adapted online without user intervention.

The book gives a comprehensive introduction to Bayesian models as well as to control theory. A link is made between classical methods for describing and handling linear and non-linear dynamic systems at the one side and dynamic Bayesian networks on the other side. Although the book describes two applications from manufacturing technology to prove the applicability, the theory is developed in a general way. First, the reader clearly gets used to linear and non-linear dynamic systems before the relation is shown to dynamic Bayesian networks. Well known non-linear units, like saturation or hysteresis, are discussed in the following. The results of modeling hydroforming and injection moulding show the benefits of the approach.

Prof. Joachim Denzler

Chapter 1

Introduction

1.1 Problem presentation

Manufacturing technology has to face several challenges. The time to market is getting shorter [sfb95; GL99] on the one side, but a high quality is of great importance, as an unsatisfied costumer might change to the competitor [Pfe93] and tell his opinion a lot of other people [Pfe93]. An additional problem for countries like Germany are high wages [sfb95], so that there is a high need for automation. The collaborative research center SFB 396 tries to master this challenge by several measures, e.g. by experiments with new materials or combination of new materials, or the integration of several process-steps. To achieve the goals also modeling plays an important role [sfb95]. This is not only valid in the scope of the collaborative research center, but also for manufacturing technology in general [Lan96].

A model is a simplified image of reality which abstracts, depending on the intended application, from unimportant details. In [Lan96] three different groups of models are distinguished. *Process models* are a quantitative mapping of continuous processes. A second group are *control models*, describing the relationship between control devices and the technical processes. *Information technical models* are used, if automation with a process computer is intended, and they represent the automation tasks.

Within the scope of this thesis only process models are discussed. Process models can further be characterized as either static, e.g. used for the calculation of suitable settings for input variables, or dynamic, if the course of the measured variables is of interest. An example for a static process model is the modeling of the distribution of the forces during hydroforming, with the aim to guarantee an equal distribution at all points of measurement.

Depending on the planned application, different means for modeling are used, e.g. mathe-

matical models or Petri nets [Lan96].

In automation also statistical methods are used, e.g. control charts [Pfe98; GL99]. Control charts record the output variables, representing the quality of the process. Additionally upper and lower thresholds are defined. When measurements are beyond those thresholds an action is required. In this thesis also a statistical approach is suggested, based on Bayesian networks. But in contrast to control charts the suggested approach aims at automatic control, the input variables are directly calculated using the statistical model.

Bayesian networks represent the distribution of multiple, discrete or continuous, random variables. For a better overview, the variables are displayed in an acyclic, directed graph. The broad applicability of Bayesian networks can be attributed to several advantageous features.

- There are a lot of training algorithms available, both for the structure and the parameter of Bayesian networks [Bun94; Jor99; Mur02; RS97; GH94; FK03].
- Efficient inference algorithms are developed in the last 15 years [LS88; Lau92; LSK01; Pea88].
- It is possible to work with missing measurements and hidden variables [Mur98b].

To apply Bayesian networks to automation, there are a number of requirements, listed shortly in the following paragraphs. Of course, most of these requirements are not only specific for Bayesian networks, but for modeling in general.

First, the accuracy of the model is required; i.e., the deviation between reality and the predictions of the models should be minimized.

The second desired feature is the ability to generalization. That means the model must be able to make predictions also for yet unpresented settings of the in- or output variables.

In many cases the structure of the model is also subject of modeling [Lan96]. In dependency on the required application other demands may occur.

Sometimes a number of experiments is executed, to find out a suitable setting for the input variables. To reduce costs, usually test-plans are used. The aim is to reduce the number of experiments with a small impact on the knowledge gained by the experiments. This leads to a low number of training sets which complicates a statistical analysis, particularly with discrete Bayesian networks.

If the model is to be applied in quality management, there is a need for a permanent adaptation as additional data are collected during the production process.

When dynamic models are applied in control, other requirements have to be met.

- As a controller should act without supervision of an operator, a robust model and controller is needed. An oscillation has to be avoided in all cases.
- The calculation of the manipulated variables has to be done in real-time. The required response-time depends on the controlled system.
- Failure of sensors should be compensated, so that stability of control is not affected.
- An on-line adaptation of model-parameters is desirable, to keep track with a changing environment.

In the next two sections the state of the art in control theory is discussed, and a brief overview about applications of Bayesian networks is given. In section 1.4 the contribution of this thesis to the field of modeling and control is sketched.

1.2 State of the art in modeling and control

In section 1.1 it is mentioned, that intelligent modeling and control plays an important role for manufacturing processes and industrial production. As the largest part of this thesis considers this problem from the point of view of Bayesian networks, this section deals with alternative approaches, so that a more comprehensive picture is given. The approaches of traditional control are omitted, they are discussed in chapter 3.

In the following different approaches from artificial intelligence are discussed mainly from the point of view of modeling manufacturing processes and of industrial control. In the last decades rule-based systems, neural networks, fuzzy control, evolutionary systems, and statistical process control are frequently applied approaches in automatic process control [JdS99; FJdS99].

Rule-based systems use a knowledge base, with several rules, to define suitable control actions. In comparison to the other algorithms, they play a smaller role in industrial control. Major problems are the knowledge acquisition, needed to develop the knowledge base. An additional drawback is, that usually a set of rules is not suited to generate numerical control signals, needed for control purposes. One of the advantages is, that it is easy to generate an explanation for the suggested control action.

Neural networks are typically divided into an input layer, several hidden layers, and an output layer. Typically each node in a layer is connected to each node in the succeeding layer. Of course there are several exceptions, like Boltzmann machines [Dev94] and Hopfield nets [RM86], which are fully connected. Each layer consists of several nodes. The behavior of the complete net is

governed by the weights of the connections between the nodes and the activation function, which maps the weighted input of the parent nodes to the output. During the training process, the weights of a neural network are adapted, while the type of nodes, the structure, and the activation function are kept constant. For the training process two different scenarios are distinguished, supervised and unsupervised learning.

During the unsupervised learning process only the input is given. It is the task of the training process, to detect frequently occurring patterns. One of the most popular representative is the *Kohonen network*, which consists of one in- and output layer. During the training process the weights of the winner-neuron, whose weights are closest to the input, are adapted, so that its weights get closer to the presented input. Additionally, the weights of its neighbors are changed in the same way, depending on the distance to the winner neuron. At the end of the training process neighbored neurons react on similar inputs.

In supervised learning the input and the desired output are presented to the net. A well-known example is the backpropagation algorithm. In this case the weights are adapted in order to minimize the error between actual and required output.

For the purpose of control one should keep in mind that usually the direction of inference is fixed. Thus it is impossible to train a neural network, so that it simulates the behavior of a dynamic system and put the desired value at the output nodes to get the manipulated value. In [WSdS99] different methods for neural control are discussed.

The simplest one is *supervised control*. Here the neural network is trained, so that it copies the behavior of an existing controller. After the training is finished the neural network is used instead of the controller, used for the training.

In the approach of ”*direct inverse control*” the network “is trained to learn the inverse dynamics of the system” [WSdS99]. That means that the systems output is used as the input of the neural network, which tries to predict the system’s input which has led to the observed output. To correct the weights the predicted input is compared with the actual one, so that supervised learning is possible. After the training process the neural network acts as controller; i.e., the desired value is used as input of the neural network, the output of the net is used as input signal for the dynamic system.

In ”*neural adaptive control*” the neural network is trained to learn the parameters of a plant and adapt a controller based on this information. In contrast to supervised control and to direct inverse control the neural network does not act as controller, but to improve the performance of an existing controller [WSdS99]. When only information about success or failure are available reinforcement learning could be used.

The great advantage of neural networks is the ability to cope also with new situations and to learn nonlinear functions with an arbitrary precision. A drawback is that a neural network is a kind of black box, so that there is nearly no possibility to interpret the training results.

Another approach is Fuzzy Logic, based on the theory of Fuzzy sets, suggested by Lotfi Zadeh in the late sixties [Zad65]. In principle *Fuzzy Control* [Pre92; HR91] is based on a number of control rules like

If 'Pressure' = 'high' and 'Temperature' = 'high' then 'Cooling temperature' = 'low',

which are given either by a knowledge engineer or are trained in combination with a neural network. In comparison to a rule based system, the predicates of the preconditions are not evaluated in a boolean manner, but are mapped to an interval [0 1], representing the degree, a precondition is fulfilled. After the degree of truth is assigned to each predicate, the boolean operators are applied. Typically the result of the 'and'-operator is mapped to the minimum of both truth-degrees, the 'or'-operator is mapped to the maximum of both operands. In this way a degree of truth is assigned to the conclusion. To combine the results of several rules, making predictions for the same variable, *defuzzyfication* has to be applied to all results. A method regularly applied, is to use the center of gravity as final result. Fuzzy control can be combined with neural networks to train the fuzzy rules or with a rule based system to enable it to generate numerical solutions.

When control is regarded as an optimization problem *evolutionary algorithms* can be applied [Nom99]. An example might be the assignment of jobs to different manufacturing systems. The main idea of evolutionary algorithms is to code the solution, e.g. the algorithm used for control, in so called chromosomes, e.g. as strings or as trees with operators and variables as leaves and nodes. At the beginning a lot of solutions are collected in an initial population. Afterwards the quality of the solution is judged by a fitness function. To generate a new population, new chromosomes are generated using the old population, where chromosomes, representing a good solution are used more frequently. The new chromosomes are changed with a low probability, e.g. by flipping a bit in the chromosomes. This imitates the process of mutation. After the new population is generated, the iteration of evaluation and generation of new chromosomes begins once again. The idea is that the overall quality of the solutions increases, as better chromosomes are preferred during reproduction. A monotonous increase of the quality can be guaranteed, when the best chromosome is kept in the population, i.e. the elitist strategy is applied. Evolutionary algorithms are applied, when the fitness function can not be differentiated, i.e. the application of gradient descent is impossible.

Statistical methods are seldom used. Chapter 7 of [Lu96] discusses the application of statistical process control, based on statistical process charts. The idea is to apply principal component

analysis on both the domain of in- and output variables, to reduce the dimension, and afterwards learn the dependency between in- and output variables. Thus a prediction of the quality of the output is possible, and it is possible to trigger the change of the input variables in real time, when the process is out of control.

This section shows, that there are a lot of possibilities for intelligent process control, Bayesian networks are usually not mentioned in the literature about industrial control. The next section will give a coarse overview about the application of Bayesian networks to show, that in principle the preconditions for the application of Bayesian networks in control are given.

1.3 Bayesian networks and their application

In the last section it was discussed that there are already a lot of means to deal with control problems. This section shows that Bayesian networks are applied in different domains, e.g. medical diagnosis, user-modeling, and data-mining. Some applications are time critical [HB95; HRSB92]. As a lot of training algorithms are available the most important prerequisites for self-adaptive control are given.

In general Bayesian networks can be regarded as a mean to represent the relation between several random variables in a directed graph. The nodes in that graph represent the random variables. The arcs between the nodes stand for the dependency of the nodes.

When a distribution of discrete random variables has to be modeled, the effort grows exponentially with the number of used nodes. This has led to the conclusion, that it is an intractable task to develop an expert system, based on probability theory [Jen96]. To avoid this exponential growth of complexity, the inference process in a Bayesian network is based on local distributions of one random variable, depending on its parents. This measurement makes the statistical inference tractable and reduces also the number of parameters. In a decision theoretic framework influence diagrams [Zha98a; Zha98b; Jen01], i.e. Bayesian networks with additional nodes to represent actions and their utility, are used. When time dependent relations have to be represented dynamic Bayesian networks can be used. A deeper introduction in Bayesian network and dynamic Bayesian network is given in chapter 2.

The most popular application of Bayesian networks seems to be the Office-Assistant, delivered the first time as part of the Office 97 package. The assistant was developed within the framework of the Lumière project [HJBHR98; HH98], starting 1993. The aim of the project was to find out the goal and the needs of the user, based on the state of the program, past actions, and on a possible query to the help system. In a prototype it was even tried to save the

estimated competency of the user in the registry, which should be used as an additional source of information. As the aims of the user vary within time, a Dynamic Bayesian network is used for the representation of the user's goal [HJBHR98; Had99]. The past actions are transformed by the Lumière Events Language into predicates, which can be regarded as random variables and modeled in a Bayesian network. Thus the program can identify pattern like plan-less search and a pause which might indicate the need for help. Using a Bayesian network a probability distribution of the user's need is calculated. In a prototype a icon popped up, when a threshold is exceeded. In the final version this knowledge is used to improve answering queries to the system by the identification of the user's goal.

From the historical view the first real-world applications are medical expert systems, e.g. the MUNIN system [Kit02; AJA+89] for "electromyographic diagnosis of the muscle and nerve system"[Lau01], the pathfinder system [HHN91] for "diagnosis of lymph vertex pathology" [Lau01], which was later commercialized as the Intellipath system [Jen96]. Additional examples are the probabilistic reformulation [SMH+91] of the INTERNIST-1/QMR knowledge base and the Child system at the Great Ormond Street Hospital in London for the diagnosis of heart diseases [Jen96].

The MUNIN system, developed at the University of Aalborg, is used for the diagnosis of 22 different diseases [Kit02] with 186 symptoms, and has the ability to detect several diseases at the same time. It is structured in 12 units, representing different muscles and nerves, with 20 - 150 nodes each. Each unit is structured in three layers, where the first two layers represent the pathological variables, and the last layer the variables for diagnosis.

In comparison to the MUNIN system, the PATHFINDER system has a simpler structure. There is one central variable with more than 60 states, representing the different diagnoses. Thus the system is not able to represent more than one disease at the same time. As parents of the diagnosis variable there are 130 information variables, sometimes linked with each other [Jen96]. First attempts of user modeling are also found in the PATHFINDER system [HJBHR98] in order to adapt the questions and answers of the expert system to the competence of the user.

The Child system helps the pathologists at the hotline of the Great Ormond Street Hospital, to decide whether a blue baby should be transported to a special hospital or not. The structure and probabilities was initialized by experts, and later refined by existing cases. As a result the system could compete with experts in that domain [Jen96].

Another domain of application is the modeling of technical systems. A system, based on a Bayesian model of a technical process is the Vista system [HB95; HRSB92], "which has been used for several years at NASA Mission Control Center in Houston" [Had99]. Its aim is to

decide in real-time which information is displayed to the operators in the control center. To do so a model of the propulsion system, the example used by the Vista-project, is developed including all types of possible errors, e.g. failure of sensor data. When the Bayesian model detects an abnormal situation, the most probable causes are displayed to the user together with the necessary sensor data, to deal with the situation. The utility of the information is measured by the expected value of revealed information, a measure used to judge the utility of new information. The decision, which information is displayed to the user, is based on an influence diagram, where the utility of each action is measured be the expected value of this information.

The intended application in this thesis has to react in real-time. Examples for systems reacting in real-time are of course the Vista system. Another example is object detection and tracking in images of an infrared camera, supported by Bayesian networks[Pan98]. Also in control Bayesian networks are discussed. Welch [WS99] discusses sorting of contaminated waste with a hybrid Bayesian network. To achieve real-time properties only parts with changed evidence are updated.

This section has only shown the most prominent examples, other applications are Data-Mining [Hec97], trouble-shooting, e.g. for printer [BH96; SJK00], and surveillance of an unmanned underwater vehicle [Had99]. A rich source of additional applications is found in [CAC95; HMW95; Kit02; Lau01; Had99]. Haddawy [Had99] and F. V. Jensen [Jen01] offer a list of available toolboxes for modeling Bayesian networks.

1.4 Contribution of the thesis

As seen in section 1.1, static and dynamic process models are distinguished. Applicable models are developed in both domains.

Static models For static models, the technique of piecewise *linear approximation* is evolved. The main idea is to represent some input parameters both by a discrete node and a continuous node. The discrete nodes are used to implement a kind of skeleton of the desired function. The number of states, being proportional to the number of points in the skeleton, depends on the required accuracy and is restricted by the available training data. This approach has several advantages.

- From the practical point of view no special software is required.
- The idea is based on the general idea of function approximation by (multiple) Taylor series, it is not only applicable in the domain of modeling manufacturing processes. It is expected that this technique is also applicable, e.g. in the field of data-mining.

- Both discrete and continuous nodes are implemented directly. Thus no discretization is required. Therefore loss of information, caused by discretization, is avoided.

- As shown in section 5.1 each node has a special meaning. That is, it is easy to incorporate a-priori knowledge into the model. For example it is easy to deduce initial parameters for the Bayesian network. Also the interpretation of the training results is easy, so that the learned parameters can be used to gain insight in the modeled processes.

The concept of modeling a function or a manufacturing process by linear approximation by multiple Taylor series is applied to different manufacturing processes, e.g. to preforming and calibration of hydroforming, and to injection moulding. In all cases a great accuracy of the model is demonstrated. A comparison with the standard deviation of the different processes shows that a large part of the prediction error is due to scattering of the data.

Also the *ability to generalization* of Bayesian models is satisfactorily shown. This is particularly of importance, because test-plans that lead to missing configurations are frequently used in manufacturing technology. If the observation of a missing configuration is a prerequisite for the training of the model generalization fails. The thesis discusses the relationship between test-plans and the structure of the Bayesian networks. It is suggested that for each set of discrete nodes, being parent of an arbitrary node, all possible settings of the parent nodes have to be observed. This criterion is applied with great success to the modeling of laser beam welding.

The development of Bayesian networks for different manufacturing processes shows that Bayesian networks provide a suitable mean to build accurate models, which make sensible predictions even for yet unpresented inputs.

Dynamic models The second focus of the thesis is the modeling of dynamic, in most of the cases linear, processes. A framework is developed, to use dynamic Bayesian networks as controller. The main idea is to estimate the state of the system using information about former in- and output signals. Using the desired value as additional source of information, the Bayesian network is able to calculate the required input signals, which lead to the desired output. By comparison between the predicted and the observed output, the disturbance variable is estimated. Assuming, that the disturbance value changes slowly in comparison to the sampling period, the estimation of the disturbance variable is propagated to the future and can therefore be included in the calculation of the input signal.

To ensure a *broad applicability* a general model (state space model), well-known in control theory, is used as controlled system to test the new approach. This model can easily be transformed into two different structures for the dynamic Bayesian network. First, the state space

approach, where the information about the state of the system is stored in hidden state nodes, can be used directly to deduce the structure.

In the second approach (difference equation model) the information about the state of the system is gathered by access to former in- and output nodes. A comparison between the state-space approach and the difference equation approach shows that the latter is more stable in all our experiments. The reason is that the lower number of hidden nodes leads to better training results.

Thus the primary precondition, the *stability* of a controller is fulfilled. In a second step the provided accuracy, depending on the number of time-slices and therefore on the time required for inference, is tested. It turns out, that the number of time-slices can be severely reduced. Reducing the number of time-slices leads only to a minor reduction of the quality in terms of the steady state error and the sum of the squared error.

For hybrid dynamic Bayesian networks inference time is proportional to $k^{n_{\text{past}}+n_{\text{future}}}$ with k as the number of configurations per timeslice and $n_{\text{past}} + n_{\text{past}}$ as the number of time-slices used to represent past and future. Thus this result is of great importance also for hybrid, dynamic Bayesian networks. Despite the encouraging results inference time remains a great problem before hybrid Bayesian networks might be applied for control purposes.

1.5 Overview

This thesis is situated at the intersection of different domains. First the thesis can be seen from the point of view of the intended application, i.e. the modeling of manufacturing processes and the control of dynamic systems. On the other hand the used algorithms are from the domain of mathematics or computer science. As it is the aim that this thesis is understandable for both engineers and computer-scientists an introduction is given in all domains. Chapter 2 deals with an introduction to Bayesian networks. A focus of this chapter are hybrid Bayesian networks, as they are needed to model nonlinear processes. Additionally, the inference process used for hybrid Bayesian networks, is also used for Dynamic Bayesian networks.

Chapter 3 introduces the most important points from control theory. Major theme is the state space description, including a short discussion of normal forms. Also the description of dynamic systems by difference equations is explained, as this theory provides the background of our models. Additional points are the setting of controller parameter's, to provide us with means, to compare Bayesian controller with traditional ones.

The experiments with dynamic systems are discussed in chapters 4 and 5. The former dis-

cusses the application of Bayesian controllers to linear dynamic systems and the comparison of state space and difference equation model.

The latter discusses the research concerning modeling and control of non-linear systems. The concept of linear approximation by multiple Taylor series is introduced. In the second part of chapter 5 the control of nonlinear systems is discussed.

Chapter 6 introduces briefly the modeled manufacturing processes. This chapter provides only a short discussion of the parameters. The physical background is omitted.

The models developed for the manufacturing processes that are introduced in chapter 6, are provided in chapter 7. Similar techniques to those, discussed in chapter 5, are used. An additional requirement, discussed in chapter 8, is real-time. Here the measures which can be taken to react in real-time, mainly the dependency on the number of time slices, is examined. The thesis finishes with an overview about the results and suggestions for future work in chapter 9.

Chapter 2

An introduction to Bayesian networks

2.1 Preliminaries

It is our aim to model technical processes using statistical means. This section will shortly introduce the terminology of probability theory, i.e. random variables and their dependencies are discussed. A deeper introduction is given in [Bre69; Bre73].

Manufacturing processes depend on many different parameters, e.g. welding depends on the power and the velocity of the laser beam. The quality of the joint might be measured by the tensile strength, i.e. the force needed to divide the two blanks. In our experiments a force F in an interval [0 N , 5200 N] as set of possible outcomes Ω was measured.

In a first case the engineer might be only interested whether the tensile strength exceeds a threshold $F_{\min}$, i.e. a mapping

$$F_1(\omega^r) = \begin{cases} 1 & \text{if } \omega^r \leq F_{\min} \\ 2 & \text{otherwise} \end{cases} \tag{2.1}$$

from the outcome of an experiment ω^r to a finite set $\{1\,2\}$ is used. In this case F_1 is a *discrete random variable* as $F_1(\omega^r)$ has only a finite number of possible values.

In a second case the tensile strength itself is of importance, i.e. the identity is used as mapping $F_2(\omega^r) = \omega^r$. In this case the domain of the mapping F_2 is infinite, F_2 is a *continuous random variable* . In the following discrete random variables are denoted by X, continuous ones by Y or Z.

It is possible to assign a probability $P(X(\omega^r) = x)$, abbreviated by $P(x)$, to the result $X(\omega^r)$. For continuous random variables a distribution $p(Y(\omega^r) = y)$, or shorter $p(y)$, has to be used, as

it is not possible to assign a positive probability to a single event y.

When modeling technical processes usually more than one random variable is involved, i.e. dealing with probabilities $P(X_1, X_2, \cdots, X_{n_N})$ is essential. Sets of random variables $\boldsymbol{X} = \{X_1, X_2, \cdots, X_{n_N}\}$ are denoted by calligraphic characters, thus $P(X_1, X_2, \cdots, X_{n_N})$ is abbreviated by $P(\boldsymbol{X})$.

A *probability table*, which assigns a probability to each event $(X_1 = x_1, X_2 = x_2, \cdots, X_{n_N} = x_{n_N})$, will grow exponentially with the number of random variables n_N. To factorize the probability $P(X_1, X_2, \cdots, X_{n_N})$ the *conditional probability*

$$P(X_1|X_2) = \frac{P(X_1, X_2)}{P(X_2)} \tag{2.2}$$

might be used to rewrite $P(X_1, X_2, \cdots, X_{n_N})$ to

$$P(X_1, X_2, \cdots, X_n) = P(X_1) \prod_{i=2}^{n_N} P(X_i|X_{i-1}, \cdots, X_1) \ , \tag{2.3}$$

which is known as the *chain rule*. Sometimes $P(\boldsymbol{X}_1|\boldsymbol{X}_2, \boldsymbol{X}_3) = P(\boldsymbol{X}_1|\boldsymbol{X}_2)$ holds, that is the state of the random variables $\boldsymbol{X}_3$ does not matter, provided that the state of $\boldsymbol{X}_2$ is known. In this case $\boldsymbol{X}_1$ and $\boldsymbol{X}_3$ are called *conditionally independent* given $\boldsymbol{X}_2$, denoted by $\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_3|\boldsymbol{X}_2$. When $\boldsymbol{X}_2$ is empty, $\boldsymbol{X}_1$ and $\boldsymbol{X}_3$ are called *independent*.

Conditional independency can be used, to rewrite the chain rule to

$$P(X_1, X_2, \cdots, X_{n_N}) = P(X_1) \prod_{i=2}^{n_N} P(X_i|\mathbb{P}(X_i)) \ , \tag{2.4}$$

where $\mathbb{P}(X_i) \subseteq \{X_1, X_2, \cdots, X_{i-1}\}$ are called the *parents* of X_i. Variables which are not in the set of parents $\mathbb{P}(X_i)$ are assumed to be conditionally independent of X_i. When $\mathbb{P}(X_i)$ is a true subset of $\{X_1, X_2, \cdots, X_{i-1}\}$, the conditional probability table for $P(X_i|\mathbb{P}(X_i))$ has less entries than $P(X_i|X_1, X_2, \cdots, X_{i-1})$.

As an example a snapshot of a family's life is regarded. First, the season $S \in \{$'spr','sum' , 'fal','win'$\}$ which has an influence on the state of the heating $H \in \{$'on','off'$\}$, and on problems starting the car ($S_{\mathrm{P}} \in \{$'yes','no'$\}$), is observed. The heating is only switched on, when the family is not absent ($A \in \{$'yes','no'$\}$. The last random variable is the cost of energy $E_{\mathrm{c}} \in \{$'low','$medium$','$high$'$\}$. The probability distribution

$$P(S, A, H, S_{\mathrm{P}}, E_{\mathrm{c}}) = P(S)P(A|S)P(H|S, A)P(S_{\mathrm{P}}|S, A, H)P(E_{\mathrm{c}}|S, A, H, S_{\mathrm{P}}) \tag{2.5}$$

$P(A)$	
$A =$'yes'	0.15
$A =$'no'	0.85

Table 2.1: Probabilities for node Absent

$P(S)$	
$S =$'spr'	0.25
$S =$'sum'	0.25
$S =$'fal'	0.25
$S =$'win'	0.25

Table 2.2: Probabilities for node Season

$P(E_c\|H)$	$H =$'on'	$H =$'off'
$E_c =$'low'	0.1	0.9
$E_c =$'med'	0.7	0.05
$E_c =$'$high$'	0.2	0.05

Table 2.3: Conditional probabilities for costs of energy E_c

$P(H =$'on'$\|S, A)$	$A =$'yes'	$A =$'no'
$S =$'spr'	0.1	0.3
$S =$'sum'	0.01	0.05
$S =$'fal'	0.1	0.3
$S =$'win'	0.2	0.99

Table 2.4: Conditional probabilities for node Heating

$P(S_P)$	$P(S_P =$'yes'$\|S)$
$S =$'spr'	0.05
$S =$'sum'	0.05
$S =$'fal'	0.05
$S =$'win'	0.3

Table 2.5: Conditional probabilities for node starting problems S_P

might be simplified to

$$P(S, A, H, S_P, E_c) = P(S)P(A)P(H|S, A)P(S_P|S)P(E_c|H) \tag{2.6}$$

which reflects the assumption that, e.g., the probability of absence does not depend on the season, and that the costs of energy are independent from the season and of being absent, provided that the state of the heating is given.

The probabilities for this example can be defined as in tables 2.1 – 2.5. They will be used later in this chapter to illustrate the inference algorithm for Bayesian networks. Table 2.4 shows only the probability for $P(H =$'on'$|S, A)$. As the probabilities sum to one $P(H =$'off'$|S, A) = 1 - P(H =$'on'$|S, A)$. The probabilities for $S_P =$'no' are calculated similarly.

2.2 Definition of Bayesian networks with discrete variables

To illustrate conditional independency, a directed acyclic graph (DAG) with edges pointing from the parents of $\mathbb{P}(X_i)$ to X_i is used.

The DAG representing the independencies of equation (2.6) is pictured in figure 2.1. These considerations result in the definition of a *Bayesian network.* Bayesian networks (BNs) are a compact graphical representation of a probability distribution, and exhibit the conditional inde-

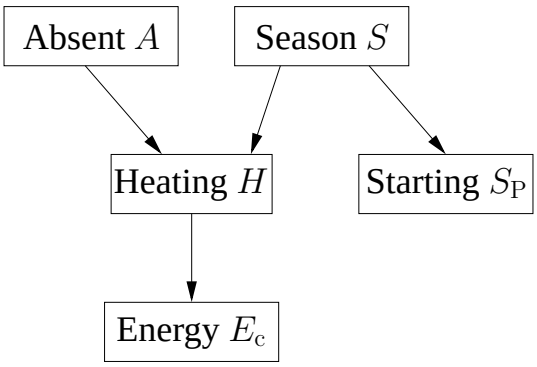


Figure 2.1: Graphical representation of the example

Figure 2.2: Two different Bayesian networks for the same distribution

pendency of the distribution's random variables.

Let $\{X_1, \ldots, X_{n_N}\}$ be a set of random variables, each of which takes values in some domain $\mathrm{Dom}(X_i)$. A Bayesian network over $X_1, \ldots, X_{n_N}$ consists of two components: a directed acyclic graph $\mathcal{G}(\boldsymbol{V}_\mathcal{G}, \boldsymbol{E}_\mathcal{G})$, with $\boldsymbol{V}_\mathcal{G} = \{X_1, \ldots, X_n\}$ as its set of vertices and $\boldsymbol{E}_\mathcal{G} = \{X_i \to X_j | X_j \in \mathbb{P}(X_i)\}$ as set of edges. To each node X_i a *conditional probability distribution (CPD)* $P(X_i \mid \mathbb{P}(X_i))$ is assigned (see [SGS01], page 13).

Note that a Bayesian network defines a unique probability distribution, but not vice versa. A very simple example is the representation of $P(X_1, X_2)$ in figure 2.2, which is equal to $P(X_1)P(X_2|X_1)$ or to $P(X_2)P(X_1|X_2)$. But usually there is an edge from X_1 to X_2 if changing X_1 has an influence on X_2, i.e. X_1 may be regarded as causing an effect on X_2. A detailed discussion about *isomorphic Bayesian networks*, i.e. BNs representing the same probability distribution, can be found in [HG95].

The next question is how changes in the evidence, e.g. observing that $S = \mathit{'win'}$, changes the probability distribution. Before introducing an algorithm for the recalculation of the probability distribution when new evidence is observed, a quantitative discussion is given to gain a more intuitive insight. This discussion (compare [Jen96; Jen01; SGS01]) is based on different types of connections in a BN.

If there is a directed path $X_\mathrm{s} \to \cdots \to X_i \to \cdots \to X_\mathrm{e}$ from an arbitrary start node X_s to an end node X_e, this is called a *serial connection* between X_s and X_e. Observing X_s has of course

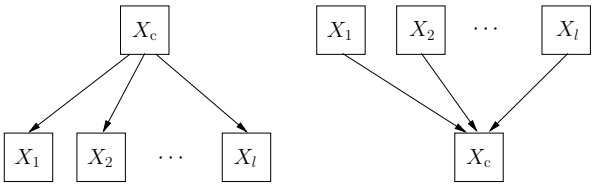


Figure 2.3: Diverging connection (left) and collider (right)

an influence on the probability distribution of X_e, unless there is an evidence for X_i. In our example the observation that the season is winter would increase the probability of high energy costs, unless it is notified, that the heating is switched off. A common cause X_c for different events $X_1 \cdots X_l$ is usually modeled by a *diverging connection*. (Confer left hand side of figure 2.3). For example an observation that the heating is on results in a higher probability for winter, which in turn leads to a higher probability for starting problems with the car. This example shows that a diverging connection also enables the flow of information, i.e. an observation for one node leads to a changed probability distribution of the other node.

The last type of connection, called *collider* (see right hand side of figure 2.3), is used to model random variables $X_1 \cdots X_l$, all causing an effect on X_c. If X_c is not observed, changes in the probability distribution of $X_i, 1 \leq i \leq l$ have no effect on the probability of $X_j, 1 \leq j \leq l, i \neq j$. For example observing that the family is absent ($A = 'yes'$), has no effect on the season S.

The three types of connections can now be used to characterize the information flow in a Bayesian network.

In a serial connection $X_s \rightarrow X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_l \rightarrow X_e$ the information flow might be blocked by instantiating X_i, so that changes of X_s no longer have an influence on X_e. Diverging connections $X_i \leftarrow X_c \rightarrow X_j$ show a similar behavior. Here the information flow between X_i and X_j can be blocked by instantiating the common source X_c.

A collider $X_i \rightarrow X_c \leftarrow X_j$ shows a different behavior. When no information is given about X_c, changes in X_i have no influence on X_j. Contrarily, when evidence is given for X_c or some of its descendents the information flow between X_i and X_j is enabled. These considerations lead to the definition of d-separation.

Two distinct variables X_s and X_e in a causal network are called *d-separated* if, for all paths between X_s and X_e, there is an intermediate variable X_i (distinct from X_s and X_e) such that either

- the connection is serial or diverging and X_i is instantiated or
- the connection is converging and neither X_i nor any of X_i's descendants have received

evidence.

If X_s and X_e are not d-separated, we call them *d-connected*.

If the Bayesian network is readily defined, i.e. the DAG and the conditional probabilities are determined, there are several tasks for which the Bayesian network might be used. The first one is called *marginalization*. Here the user is not interested in a distribution of all random variables $\boldsymbol{X} = \boldsymbol{X}_\mathrm{p1} \dot{\cup} \boldsymbol{X}_\mathrm{p2}$, but only in a part $\boldsymbol{X}_\mathrm{p1} \subset \boldsymbol{X}$ of it. Thus it is necessary to calculate

$$P(\boldsymbol{X}_\mathrm{p1}) = \sum_{\boldsymbol{X}_\mathrm{p2}} P(\boldsymbol{X}_\mathrm{p1}, \boldsymbol{X}_\mathrm{p2}) \ ; \tag{2.7}$$

i.e., to sum over all the variables $\boldsymbol{X}_\mathrm{p2}$ which are not in the marginal distribution. A similar operation exists for continuous random variables, the only difference is that in this case summation is replaced by integration.

A second frequently occurring question is: How is the probability distribution $P(\boldsymbol{X})$ changed, if $X_i = x_i$ is known, i.e. how is $P(\boldsymbol{X}|X_i = x_i)$ calculated? The next section introduces the frequently used junction tree inference algorithm which is one way of efficiently calculating the requested distributions.

2.2.1 Junction tree algorithm

To use the Bayesian network, e.g. in an expert system or in a controller, an efficient inference machine is necessary to calculate marginal distributions, include evidence, or to calculate instantiations of the random variables which lead to a maximal probability. One algorithm for the propagation of evidence is introduced in [Pea88], but in most of the cases the *junction tree* algorithm is used as described e.g. in [LS88] or [Jen96], where inference tasks are done in a hypergraph called *junction tree*. The junction tree is generated from the DAG in several steps.

Moralization and triangulation

As a first step, all nodes with a common child are connected. At the same time all directions in the original DAG are dropped. The resulting graph is called a *moral graph*. The moral graph resulting from figure 2.1 is given in figure 2.4. As a result the link between the nodes Absent and Season is added.

Next the moralized graph is triangulated. As the triangulation is applied to an undirected graph (directions are dropped during moralization) $\mathcal{G}(\boldsymbol{V}_\mathcal{G}, \boldsymbol{E}_\mathcal{G})$, the edges in $\boldsymbol{E}_\mathcal{G}$ are denoted by $X_i \text{—} X_j$.

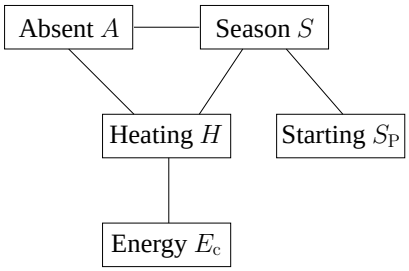


Figure 2.4: Moral graph of figure 2.1

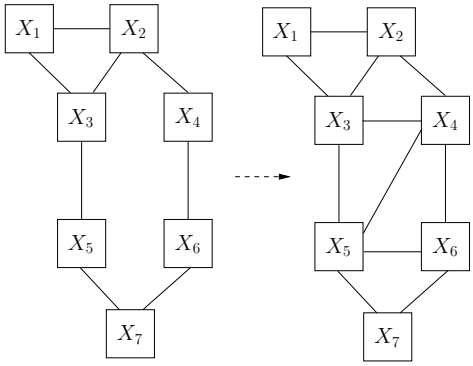


Figure 2.5: Triangulation of a graph

A undirected graph $\mathcal{G}(\boldsymbol{V}_{\mathcal{G}}, \boldsymbol{E}_{\mathcal{G}})$ is called *triangulated* if any cycle X_1 — X_2 — $\cdots$ — X_{l-1} — X_1 of length $l > 3$ has at least one *chord*, i.e. a link X_i — X_j between two non-consecutive nodes X_i and X_j.

If the required chords are not already in the set of edges, they are added, in order to get a triangulated graph. Figure 2.4 is a trivial example for a triangulated graph, as the longest cycle is of length 3. A more complex example is given in figure 2.5 which contains the cycle X_2—X_3—X_5— X_7—X_6—X_4—X_2 which has no chord. To get a triangulated graph, the links X_3—X_4, X_4—X_5, and X_5—X_6 can be added.

The process of moralizing a graph is unique, whereas the triangulation is not. It is of advantage to add as few links as possible in order to obtain the triangulated graph, as the number of links has a major influence on the time complexity of the inference process. More information about triangulation and its time-complexity is given in [Kjæ90]. A more thorough introduction into graph theory with respect to Bayesian networks is found in chapter 1 of [Lau96].

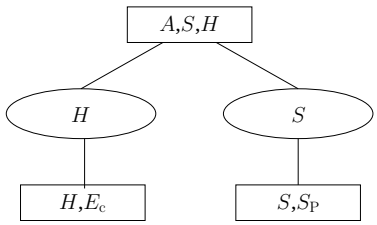


Figure 2.6: Junction tree for example in figure 2.5

Junction tree

It is now possible to identify subgraphs, where all nodes are pairwise linked for example subgraph $\mathcal{G}(\{A, S, H\}, \{A—S, S—H, H—A\})$ in figure 2.4. Maximal subgraphs with this property are called *cliques* and are used as nodes in a hypergraph.

This hypergraph is organized in a special way, so that all nodes $\boldsymbol{X}_i$ on a path $\boldsymbol{X}_\mathrm{s} — \boldsymbol{X}_1 — \boldsymbol{X}_2 — \cdots — \boldsymbol{X}_l — \boldsymbol{X}_\mathrm{e}$ between the start hypernode $\boldsymbol{X}_\mathrm{s}$ and the end hypernode $\boldsymbol{X}_\mathrm{e}$ contain the nodes of the intersection between $\boldsymbol{X}_\mathrm{s}$ and $\boldsymbol{X}_\mathrm{e}$, formally $\boldsymbol{X}_i \supseteq (\boldsymbol{X}_\mathrm{s} \cap \boldsymbol{X}_\mathrm{e})$. This property is also known as the *running intersection property* . The resulting tree is called a *join tree*. The join tree of our example contains three cliques, $\{A, S, H\}, \{H, E_\mathrm{c}\}$, and $\{S, S_\mathrm{P}\}$. According to [Jen01], there is always a way to organize the cliques of a triangulated graph into a join tree.

For inference purposes additional nodes containing the random variables in the intersection of two neighbored nodes are added. These additional nodes are called *separators*. The join tree should be used for the inference process of Bayesian networks, thus a mean is missing to calculate the distributions for random variables of the join tree. To enable the calculation of distributions, tables are attached to each clique and separator of the join tree, similar to the conditional probability tables of a Bayesian network. These tables are called *potentials*, denoted by ϕ, e.g. the potential of a clique $\mathbb{C}$ is denoted by $\phi_\mathbb{C}$. In comparison to probabilities, the entries of a potential do not sum to 1. Only after message passing, discussed later in this section, these potentials may be used for the calculation of probabilities.

The domain $\mathrm{Dom}(\phi)$ of a potential ϕ is the set of random variables being represented by the potential. The resulting structure of a join tree, including the separators, together with the potentials for each node and separator, is called a junction tree. Figure 2.6 shows the junction tree of the example in figure 2.4. The rectangles are used for cliques, the ellipses for separators.

Next, the mathematical properties of potentials are discussed. Afterwards the initialization of the junction tree is discussed. As a result of the initialization the junction tree represents the same distribution as a Bayesian network.

	Season	
$\phi_{\{A,S,H\}}$	'spr'	'sum'
Absent = 'yes'	$(3.75 \cdot 10^{-3}, 0.03375)$	$(3.75 \cdot 10^{-4}, 0.037125)$
Absent = 'no'	$(0.06375, 0.14875)$	$(0.010625, 0.2018754)$
	'fal'	'win'
Absent = 'yes'	$(3.75 \cdot 10^{-3}, 0.03375)$	$(7.5 \cdot 10^{-3}, 0.03)$
Absent = 'no'	$(0.06375, 0.14875)$	$(0.210375, 2.125 \cdot 10^{-3})$

Table 2.6: Potential after initialization

Potentials

As starting point a simple example, which will later on be used to initialize the node $\{S, H, A\}$, is given in table 2.6. Each entry of the table is two dimensional representing the values for Heating = 'on' and Heating = 'off'. To be able to use potentials for the inference process in Bayesian networks, it is necessary to define multiplication, division and marginalization for potentials. Multiplication of two potentials ϕ_1 and ϕ_2 is done by piecewise multiplication of the entries. If $\mathrm{Dom}(\phi_1) = \boldsymbol{X}_1 \cup \boldsymbol{X}_2$ and $\mathrm{Dom}(\phi_2) = \boldsymbol{X}_2 \cup \boldsymbol{X}_3$ the product of ϕ_1 and ϕ_2 is defined as

$$\phi_1\phi_2(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = \phi_1(\boldsymbol{x}_1, \boldsymbol{x}_2)\phi_2(\boldsymbol{x}_2, \boldsymbol{x}_3) \ . \tag{2.8}$$

Division is defined in the same way as piecewise division of the table entries.

As an example let us take the probability for Absent as ϕ_1 and the conditional probability table for Heating as ϕ_2 (see tables 2.1 and 2.4). The resulting potential $\phi_3 = \phi_1\phi_2$ is given in table 2.7. The two entries in parenthesis represent the values for H ='on' and 'off'.The potential ϕ_3('on','spr','yes') $= \phi_1$('yes')ϕ_2('on','spr') $= 0.15 \cdot 0.1 = 0.015$. In a similar way ϕ_3('off','spr','yes') $= 0.15 \cdot 0.9 = 0.135$ is calculated. After multiplication with the potential $\phi_{\{\mathrm{S}\}}$ for the season table 2.6 is obtained.

Marginalization of a potential for $\boldsymbol{X} = X_1 \cup \mathbb{S}$ to a potential for $\mathbb{S}$, also called *projection*,

$$\phi^{\downarrow\mathbb{S}}(\mathbb{s}) = \sum_{\boldsymbol{x}_1 \in \mathrm{Dom}(\boldsymbol{X}_1)} \phi(\boldsymbol{x}_1, \mathbb{s}) \tag{2.9}$$

is defined similar to marginalization over probability tables. A complete definition of an algebra using potentials is given e.g. in [Jen96] or [Jen01]. Next the representation of probabilities by a junction tree is discussed.

	Season	
$\phi_3(H,S,A)$	'spr'	'sum'
$A =$'yes'	(0.015,0.135)	$(1.5\,10^{-3}, 0.1485)$
$A =$'no'	(0.255,0.595)	(0.0425, 0.8075)
	'fal'	'win '
$A =$'yes'	(0.015,0.135)	(0.03,0.12)
$A =$'no'	(0.255,0.595)	$(0.8415, 8.5 \cdot 10^{-3})$

Table 2.7: Potential resulting from multiplication of the conditional probabilities for Absent and Heating

Representation of probabilities by a junction tree

So far the graphical representation of the junction tree and the mathematical properties of potentials are defined. The missing link is, how the junction tree together with the potentials defines the probability distribution. The aim is that at all times the quotient of the product of all clique potentials by the product of the separator potentials is equal to the probability distribution of the Bayesian network, as expressed in 2.10.

$$P(X_1, \cdots, X_{n_N}) = \frac{\prod_{\mathbb{C}\in\mathbb{C}_J} \phi_{\mathbb{C}}}{\prod_{\mathbb{S}\in\mathbb{S}_J} \phi_{\mathbb{S}}} \tag{2.10}$$

This property is ensured during initialization and is never changed throughout the complete inference process. Another desirable property, to be reached at the end of the inference process is the *global consistency*

$$\phi_{\mathbb{C}_i}^{\downarrow \boldsymbol{X}} = \phi_{\mathbb{C}_j}^{\downarrow \boldsymbol{X}} \quad , \tag{2.11}$$

which means that the result of calculating the marginal potential is independent of the used potential.

To guarantee equation (2.10) a potential of 1, that is a potential with each table-entry equal to 1, is assigned to each clique and separator. Afterwards each conditional probability $P(X_i|\mathbb{P}(X_i))$ is regarded as a potential $\phi_{\mathbb{F}(i)}$ and multiplied with a potential $\phi_{\mathbb{C}}$ with the domain $\mathbb{F}(i) \subseteq \mathrm{Dom}(\mathbb{C})$. The set $\mathbb{F}(i) = \mathbb{P}(X_i) \cup \{X_i\}$ denotes the family of node X_i and contains the node itself together with all its parents.

Usually it is said that a variable X_i is assigned to a node in the junction tree, i.e. to a clique

$\mathbb{C}$ of random variables. This results in

$$P(X_1, \cdots, X_{n_N}) = \prod_{i=1}^{n_N} P(X_1|\mathbb{P}(X_i)) \tag{2.12}$$

$$= \prod_{i=1}^{n_N} \phi_{\mathbb{F}(i)} \tag{2.13}$$

$$= \prod_{\mathbb{C}\in\mathbb{C}_\mathrm{J}} \phi_{\mathbb{C}} \tag{2.14}$$

$$= \frac{\prod_{\mathbb{C}\in\mathbb{C}_\mathrm{J}} \phi_{\mathbb{C}}}{\prod_{\mathbb{S}\in\mathbb{S}_\mathrm{J}} \phi_{\mathbb{S}}} \ . \tag{2.15}$$

The last equation holds, as all separators $\mathbb{S}$ are initialized to one.

Here it is important to notice that several potentials $\phi_{\mathbb{F}(i)}$ may be assigned to the same node in the junction tree, e.g. A, S, and H may be all assigned to the clique $\{A, S, H\}$ in the junction tree. This assignment results in the potential of table 2.6. But only the assignment of H to the clique $\{A, S, H\}$ is obligatory.

Direct after initialization the property of equation (2.11) is not given. For example $\phi^{\downarrow\{H\}}_{\{H,E_\mathrm{c}\}}$ is 1, as no information about the state of the heating is assigned to that clique.

Message passing

To ensure consistency of the junction tree, messages are passed between the cliques of the junction tree. This results in a recalculation of the potentials.

A clique $\mathbb{C}_j$ is said to absorb knowledge from a clique $\mathbb{C}_i$, if the separator $\mathbb{S}_{ij}$ between $\mathbb{C}_i$ and $\mathbb{C}_j$ gets as new potential ϕ^*

$$\phi^*_{\mathbb{S}_{ij}} = \phi^{\downarrow\mathbb{S}_{ij}}_{\mathbb{C}_i} \tag{2.16}$$

the marginal of $\mathbb{C}_i$. Afterwards the clique $\mathbb{C}_j$ is multiplied with the quotient of the new and the old separator

$$\phi^*_{\mathbb{C}_j} = \phi_{\mathbb{C}_j} \frac{\phi^*_{\mathbb{S}_{ij}}}{\phi_{\mathbb{S}_{ij}}} \ . \tag{2.17}$$

After $\mathbb{C}_j$ has absorbed knowledge from $\mathbb{C}_i$, equation (2.10) still holds, as

$$\frac{\prod_{\mathbb{C}\in\mathbb{C}_\mathrm{J}} \phi^*_{\mathbb{C}}}{\prod_{\mathbb{S}\in\mathbb{S}_\mathrm{J}} \phi^*_{\mathbb{S}}} = \frac{\left(\prod_{\mathbb{C}\in(\mathbb{C}_\mathrm{J}\setminus\{\mathbb{C}_j\})} \phi_{\mathbb{C}}\right)}{\left(\prod_{\mathbb{S}\in(\mathbb{S}_\mathrm{J}\setminus\{\mathbb{S}_{ij}\})} \phi_{\mathbb{S}}\right)} \frac{\phi^*_{\mathbb{C}_j}}{\phi^*_{\mathbb{S}_{ij}}} \ ; \tag{2.18}$$

i.e. only the clique $\mathbb{C}_j$ and separator $\mathbb{S}_{ij}$ have changed. Including the assignments of equation (2.16) and (2.17) leads to

$$\frac{\left(\prod_{\mathbb{C}\in(\mathbb{C}_\mathrm{J}\setminus\{\mathbb{C}_j\})}\phi_\mathbb{C}\right)}{\left(\prod_{\mathbb{S}\in(\mathbb{S}_\mathrm{J}\setminus\{\mathbb{S}_{ij}\})}\phi_\mathbb{S}\right)}\frac{\phi^*_{\mathbb{C}_j}}{\phi_{\mathbb{S}^*_{ij}}} = \frac{\left(\prod_{\mathbb{C}\in(\mathbb{C}_\mathrm{J}\setminus\{\mathbb{C}_j\})}\phi_\mathbb{C}\right)}{\left(\prod_{\mathbb{S}\in(\mathbb{S}_\mathrm{J}\setminus\{\mathbb{S}_{ij}\})}\phi_\mathbb{S}\right)}\frac{\phi_{\mathbb{C}_j}\phi^*_{\mathbb{S}_{ij}}}{\phi_{\mathbb{S}_{ij}}\phi^*_{\mathbb{S}_{ij}}} \tag{2.19}$$

$$= \frac{\left(\prod_{\mathbb{C}\in(\mathbb{C}_\mathrm{J}\setminus\{\mathbb{C}_j\})}\phi_\mathbb{C}\right)}{\left(\prod_{\mathbb{S}\in(\mathbb{S}_\mathrm{J}\setminus\{\mathbb{S}_{ij}\})}\phi_\mathbb{S}\right)}\frac{\phi_{\mathbb{C}_j}}{\phi_{\mathbb{S}_{ij}}} \quad . \tag{2.20}$$

The first round of message passing is called *collectEvidence*. During collectEvidence the parent cliques $\mathbb{C}_\mathrm{p}$ absorb knowledge from their children $\mathbb{C}_\mathrm{ch}$. A parent clique is only allowed to absorb knowledge from its child, if this child has finished its knowledge absorption. Thus the leaves of the junction tree are not changed during collectEvidence. The root node is the last one to be updated as it has to wait until all of its children have finished knowledge absorption.

In a second phase *distributeEvidence* the children absorb knowledge from their parents. This phase is similar to collectEvidence, but the messages are sent in the other direction. After collectEvidence and distributeEvidence are finished, it is guaranteed that the junction tree is globally consistent. That is for any two potentials ϕ_i and ϕ_j which share common variables $\mathbb{S}$, marginalization results in the same potential

$$\phi^{\downarrow\mathbb{S}}_{\mathbb{C}_i} = \phi^{\downarrow\mathbb{S}}_{\mathbb{C}_j} \tag{2.21}$$

for $\mathbb{S}$.

In our example the potential $\phi_{\{H,E_\mathrm{c}\}}$ is initialized with the conditional probability table of E_c; i.e., $\phi_{\{H,E_\mathrm{c}\}} = P(E_\mathrm{c}|H)$. The potential $\phi_{\{S,S_\mathrm{P}\}}$ is initialized with $P(S_\mathrm{P}|S)$. The potential $\phi_{\{A,S,H\}}$, which is used as root, gets its first value from the product of $P(A), P(S)$, and $P(H|S,A)$. When collectEvidence is called, $\phi^{\downarrow\{H\}}_{\{H,E_\mathrm{c}\}}$ and $\phi^{\downarrow\{S\}}_{\{S,S_\mathrm{P}\}}$ are calculated. As both are equal to 1, i.e. a table with all entries equal to 1, nothing changes. During distribute evidence $\phi^{\downarrow\{H\}}_{\{A,S,H\}}$ is calculated. The result, at the end of distributeEvidence, is summarized in table 2.8 which is used to update the separator potential $\phi_{\{H\}}$ and the clique potential $\phi_{\{H,E_\mathrm{c}\}}$. Similar calculations are done for the other clique potential $\phi_{\{S,S_\mathrm{P}\}}$.

Introduction of evidences

Another frequently occurring task is the calculation of marginal probabilities given new evidences. Usually *hard* and *soft evidence* are distinguished. Hard evidence means the knowledge that $X = x$, and soft evidence means the exclusion of some states; i.e., $X \in \{x_i, x_j, x_k \cdots\} \subset$

	$\phi^{\downarrow\{H\}}_{\{A,S,H\}}$
H ='on'	0.363875
H ='off'	0.636125

Table 2.8: Potential $\phi^{\downarrow\{H\}}_{\{A,S,H\}}$ after distributeEvidence

$\mathrm{Dom}(X)$. Both types are handled in a similar way. A clique with $X \in \mathbb{C}_i$ is selected from the junction tree and all positions in the potential table of $\phi_{\mathbb{C}_i}$, being not consistent with the evidence, are set to zero. When evidence is entered, the root node calls collectEvidence. After this phase is finished, distributeEvidence is called. At the end of message passing the junction tree is consistent again. Marginal probabilities with respect to the new evidence e

$$P(\boldsymbol{X}, e) = \sum_{\mathbb{C}\setminus\boldsymbol{X}} \phi_{\mathbb{C}} \tag{2.22}$$

result from marginalizing using arbitrary cliques $\mathbb{C} \supseteq \boldsymbol{X}$. Of course the same holds for a separator $\mathbb{S} \supseteq \boldsymbol{X}$.

The message passing scheme in junction trees may be improved in different ways. Shenoy and Shafer[She97] save the division when calculating the new potentials. The main difference to the junction tree algorithm is that the Shenoy-Shafer algorithm sends messages that do not include the part of the potential caused by the receiver.

2.2.2 Learning algorithms for Bayesian networks

Up to now, it was assumed that the conditional probabilities $P(X_i|\mathbb{P}(X_i))$ are given, and only questions concerning the calculation of marginals and the probability of special configurations are discussed. But in reality typically only the domain knowledge of the modeled application and a lot of data are given. That is, neither the structure, nor the conditional probabilities are given. The former is not within the scope of the thesis, for a discussion see e.g. [FMR98; CH92; HGC95; HG95].

When learning the parameters of a distribution, e.g. the conditional probabilities of the Bayesian network of figure 2.1, it is of advantage, if all nodes are observed. But usually incomplete data occur frequently during model development. Additionally the usage of hidden nodes, which do not represent an existing value, is sometimes helpful in order to reduce the number of parameters. When learning the distribution, it is assumed that the unobserved values

are missing at random, i.e. that no additional information is given by the fact that some variables are unobserved. This assumption is meaningful for the technical context of this thesis. A short discussion of the different types of missing data is given in [CDLS99] or [RS97]. The data of the modeled processes also contain continuous variables. Thus, it is necessary to use a training method which is able to deal also with continuous values, ideally with discrete and continuous values at the same time.

In a statistical approach, it is supposed that the type of the distribution, e.g. Gaussian or Dirichletian, is given and that only the parameters $\boldsymbol{\theta}$ of the distribution are trained. The parameters of the distribution are regarded as an additional random variable, the probability of a special configuration $\boldsymbol{x}$ given the parameters $\boldsymbol{\theta}$ is therefore denoted by $P(\boldsymbol{x}|\boldsymbol{\theta})$.

For learning, two different approaches can be used. The first one is the *maximum likelihood* estimation. The aim is to maximize the (logarithmic) likelihood of the observations $P(\boldsymbol{x}^j)$. When $n_c(\boldsymbol{x})$ denotes, how often a configuration $\boldsymbol{x}$ is observed, the likelihood L'

$$L'(\boldsymbol{\theta}) = \prod_{j=1}^{N} P(\boldsymbol{x}^j) = \prod_{\boldsymbol{x}} P(\boldsymbol{x})^{n_c(\boldsymbol{x})} \tag{2.23}$$

is defined as the product of the probability of the N observations. More often the log-likelihood

$$L(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} n_c(\boldsymbol{x}^j) log(P(\boldsymbol{x}^j)) \tag{2.24}$$

is used.

The second approach is the Bayesian approach which is characterized by the given a priori distribution $p(\boldsymbol{\theta})$ of the parameters. The a-posteriori distribution $p(\boldsymbol{\theta}|\boldsymbol{x}^1, \cdots, \boldsymbol{x}^N)$ which incorporates the observations is calculated. Usually so called conjugate priors [Bun94] are used, so that the a-posteriori distribution $p(\boldsymbol{\theta}|\boldsymbol{x}^1, \cdots, \boldsymbol{x}^N)$ is of the same family as the a-priori distribution. For discrete Bayesian networks a Dirichlet distribution or for binary random variables a Beta distribution [Rin97] may be used.

In the following the EM algorithm [DLR77; ST95] will be discussed. This algorithm is based on the maximum likelihood principle. It is frequently used for training with missing data, and it is able to deal with discrete and continuous data at the same time [Mur98a; MLP99]. It can be even used for the estimation of a suitable structure of a dynamic Bayesian network [FMR98].

Observation $\boldsymbol{x}^i$	A	S	H	S_{P}	E_{c}
$\boldsymbol{x}^1$	'yes'	'sum'	'on'	'yes'	'med'
$\boldsymbol{x}^2$	'yes'	'sum'	'on'	'no'	'med'
$\boldsymbol{x}^3$	'yes'	'sum'	'off'	'no'	'low'
$\boldsymbol{x}^4$	'no'	'sum'	'off'	'no'	'low'
$\boldsymbol{x}^5$	'no'	'sum'	'off'	'no'	'low'

Table 2.9: Possible observations for the Bayesian network of figure 2.1

In a Bayesian network the probability $P(\boldsymbol{x})$ is factorized by the chain rule to

$$P(\boldsymbol{x}) = \prod_{i=1}^{n_N} P(\boldsymbol{x}_{(i)}|\boldsymbol{x}_{(\mathbb{P}(X_i))}) \tag{2.25}$$

where $\boldsymbol{x}_{(\mathbb{P}(X_i))}$ denotes the configuration of the parent nodes $\mathbb{P}(X_i)$ within the configuration $\boldsymbol{x}$. Similarly $\boldsymbol{x}_{(\mathbb{F}(X_i))}$ denotes the configuration of the family and $\boldsymbol{x}_{(i)}$ the instantiation of the i-th node within $\boldsymbol{x}$.

In table 2.9 five possible observations for the Bayesian network in figure 2.1 are listed. Using the observation $\boldsymbol{x}^1$ of table 2.9 the configuration $\boldsymbol{x}^1_{(\mathbb{P}(H))}$ = ('yes','sum'). Similarly the configuration $\boldsymbol{x}^1_{(\mathbb{F}(H))}$ is equal to ('on','yes','sum') and $\boldsymbol{x}^1_{(H)}$ = ('on'). Using the factorization of equation (2.25), the log-likelihood of equation (2.24) is rewritten to

$$\begin{aligned} \sum_{\boldsymbol{x}} n_c(\boldsymbol{x}) \log(P(\boldsymbol{x})) &= \sum_{\boldsymbol{x}} n_c(\boldsymbol{x}) \log(\prod_{i=1}^{n_N} P(\boldsymbol{x}_{(i)}|\boldsymbol{x}_{(\mathbb{P}(X_i))})) & (2.26) \\ &= \sum_{\boldsymbol{x}} \sum_{i=1}^{n_N} n_c(\boldsymbol{x}) \log(P(\boldsymbol{x}_{(i)}|\boldsymbol{x}_{(\mathbb{P}(X_i))})) \ . & (2.27) \end{aligned}$$

To restrict the computation of the log-likelihood to local factors $n_c(\boldsymbol{x}_{(\mathbb{F}(X_i))})$ and $P(\boldsymbol{x}_{(i)}|\boldsymbol{x}_{(\mathbb{P}(X_i))})$ equation (2.27) is adapted to

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n_N} \sum_{\boldsymbol{x}_{(\mathbb{F}(X_i))}} n_c(\boldsymbol{x}_{(\mathbb{F}(X_i))}) \log(P(\boldsymbol{x}_{(i)}|\boldsymbol{x}_{(\mathbb{P}(X_i))})) \ . \tag{2.28}$$

In equation (2.27) the term $P(H =$'on'$|A =$'yes'$, S =$'sum'$)$ occurs once for observation $\boldsymbol{x}^1$ and once for observation $\boldsymbol{x}^2$. In equation (2.28) the configuration $H =$'on'$, A =$'yes'$, S =$'sum' counts twice as it is observed twice.

A distribution of discrete random variables is determined by the conditional probabilities

$$P(X_i = x_{ij} | \mathbb{P}(X_i) = \mathbb{p}(X_i)) = \theta_j^{i,\mathbb{p}(X_i)} \quad , \tag{2.29}$$

i.e. $\theta_j^{i,\mathbb{p}(X_i)}$ denotes the probability, that the i-th random variable X_i is instantiated with the j-th value $x_{ij} \in \{x_{i1}, x_{i2}, \cdots, x_{in_i}\} = \mathrm{Dom}(X_i)$ of its domain, and that their parents are instantiated with $\mathbb{p}(X_i)$. Now the log-likelihood

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n_N} \sum_{\boldsymbol{x}_{(\mathbb{F}(X_i))}} n_c(\boldsymbol{x}_{\mathbb{F}(X_i)}) \log(\theta_{\boldsymbol{x}_{(i)}}^{i,\boldsymbol{x}_{(\mathbb{P}(X_i))}}) \quad . \tag{2.30}$$

can be expressed in terms of its parameters θ. Assuming that the parameter of different random variables or for different parent configurations are unlinked (global respectively local meta independence)[CDLS99] $\theta_{\boldsymbol{x}_{(i)}}^{i,\boldsymbol{x}_{(\mathbb{P}(X_i))}}$ is maximized by

$$\hat{\theta}_{\boldsymbol{x}_{(i)}}^{i,\boldsymbol{x}_{(\mathbb{P}(X_i))}} = \frac{n_c(\boldsymbol{x}_{(\mathbb{F}(X_i))})}{n_c(\boldsymbol{x}_{(\mathbb{P}(X_i))})} \quad . \tag{2.31}$$

For the Bayesian network of figure 2.1, table 2.9 lists five observations. Using equation (2.31) results in

$$\begin{aligned} \theta_{'off'}^{H,\{A='yes',S='sum'\}} &= \frac{1}{3} \\ \theta_{'on'}^{H,\{A='yes',S='sum'\}} &= \frac{2}{3} \end{aligned}$$

If unobserved variables $\boldsymbol{u}$ have to be taken into account each configuration

$$\boldsymbol{x} = \boldsymbol{u}\boldsymbol{o} \tag{2.32}$$

consists of an unobserved part $\boldsymbol{u}$ and an observed part $\boldsymbol{o}$. Thus $\log(P(\boldsymbol{o}, \boldsymbol{u}|\boldsymbol{\theta}))$ has to be maximized. Now things become more complicated as an estimation for $\boldsymbol{u}$ depends on $\boldsymbol{\theta}$ and vice versa. If unobserved variables occur, the EM algorithm can be used. It employs two different steps. In the first step (E-step) an estimation $\boldsymbol{\theta}^{(k)}$ from the k-th iteration is used to calculate expected values for the missing values $\boldsymbol{u}$.

This missing values are now used to calculate the estimated counts

$$\widetilde{n_c}(\boldsymbol{x}_{(\mathbb{F}(X_i))}) = E[n_c(\boldsymbol{x}_{(\mathbb{F}(X_i))}) | \boldsymbol{\theta}^{(k)}, \boldsymbol{o}^1 \cdots \boldsymbol{o}^N] \tag{2.33}$$

of $\boldsymbol{x}_{(\mathbb{F}(X_i))}$ given the N observations $\boldsymbol{o}^i$. The expected counts can be calculated

$$\widetilde{n_c}(\boldsymbol{x}_{(\mathbb{F}(X_i))}) = \sum_{j=1}^{N} P(X_i = \boldsymbol{x}_{(i)}, \mathbb{P}(X_i) = \boldsymbol{x}_{(\mathbb{P}(X_i))} | \boldsymbol{\theta}^{(k)}, \boldsymbol{o}^j) \tag{2.34}$$

using the probabilities $P(X_i = \boldsymbol{x}_{(i)}, \mathbb{P}(X_i) = \boldsymbol{x}_{(\mathbb{P}(X_i))} | \boldsymbol{\theta}^{(k)}, \boldsymbol{o}^j)$ that can easily be computed using the junction tree algorithm or other inference algorithms for Bayesian networks. The next step of the EM-algorithm is the maximization step. Similar to equation (2.31) the new parameters are now estimated by

$$\hat{\boldsymbol{\theta}}_{\boldsymbol{x}_{(i)}}^{i,\boldsymbol{x}_{(\mathbb{P}(X_i))}} = \frac{\widetilde{n_c}\big(\boldsymbol{x}_{(\mathbb{F}(X_i))}\big)}{\widetilde{n_c}\big(\boldsymbol{x}_{(\mathbb{P}(X_i))}\big)} \ , \tag{2.35}$$

the counts are simply replaced by the estimated counts. The new parameters are now used in a new expectation maximization loop. It is proven that the estimation of $\boldsymbol{\theta}^{(k)}$ converges, but not necessarily to a global maximum. Thus it is advantageous to use a-priori information, instead of starting with an arbitrary estimation for $\boldsymbol{\theta}^{(0)}$, to get a good initialization.

So far only inference and training of discrete Bayesian networks were discussed. The next step will be to add continuous variables to the Bayesian network.

2.3 Hybrid Bayesian networks

The data from the engineers do not only consist of discrete variables, like the type of blank, but also continuous variables like temperature or pressure. One possibility to cope with this situation is to find a discretization of continuous variables, e.g. by vector quantization. Of course, discretization does not only result in a loss of information. Additionally, there is no mean to make predictions for values between discrete values. Thus it seems of advantage to enhance the Bayesian network so that it can cope directly with continuous random variables.

To enable an analytical calculation of means and variances two restrictions apply. It is supposed that there are only linear dependencies between the continuous variables and that the continuous variables are normally distributed.

The restriction to linear dependencies is overcome by using both, a discrete and continuous node, for a continuous random variable, where the discrete node is triggered by the continuous on. In section 2.3.1 the distribution of a so called hybrid Bayesian network is defined. For inference there are two possible algorithms. The first one, introduced 1992 by S. L. Lauritzen [Lau92], uses two different representation schemes for the distributions. Switching between those representations involves a matrix inversion and is therefore numerically instable. This

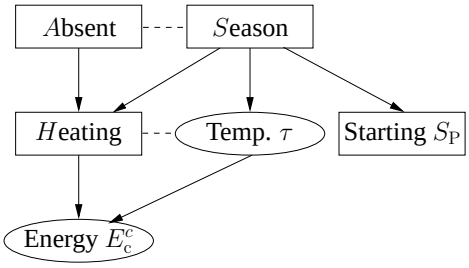


Figure 2.7: Example for a hybrid Bayesian network

drawback is fixed by the second approach, described by the same author in [LJ99].

2.3.1 Definition of hybrid Bayesian networks

Similar to discrete Bayesian networks, also hybrid Bayesian networks are defined using a DAG. The set of nodes $\boldsymbol{V}_{\mathcal{G}}$

$$\boldsymbol{V}_{\mathcal{G}} = \boldsymbol{\Delta}_{\mathcal{G}} \cup \boldsymbol{\Gamma}_{\mathcal{G}} \tag{2.36}$$

contains the random variables which can be partitioned in discrete and continuous random variables $\boldsymbol{\Delta}_{\mathcal{G}}$ respectively $\boldsymbol{\Gamma}_{\mathcal{G}}$. Once again the conditional independencies are characterized by the structure of the DAG. Usually, it is assumed that discrete variables have no continuous parents, an exception is the 'variational approximation' introduced in [Mur99]. Lerner [LSK01] suggests to expand the inference algorithm, so that also Softmax nodes, representing a distribution of a discrete random variable that depends on one or more continuous parents, can be handled.

The probability of discrete nodes can therefore be characterized by a conditional probability table. Continuous nodes Y are assumed to be normally distributed, i.e. a Gaussian distribution is defined for each configuration $\boldsymbol{x}$ of the discrete parents $\mathbb{P}(Y) \cap \boldsymbol{\Delta}_{\mathcal{G}}$. These normal distributions are defined by their mean and variance $\gamma(\boldsymbol{x})$. The mean of the CG (conditional Gaussian) distribution

$$p(y \mid \boldsymbol{x}, \boldsymbol{z}) = \mathcal{N}(\alpha(\boldsymbol{x}) + \boldsymbol{\beta}(\boldsymbol{x})^T \boldsymbol{z}, \gamma(\boldsymbol{x})) \tag{2.37}$$

depends on an offset $\alpha(\boldsymbol{x})$, the evidence $\boldsymbol{z}$ given for the continuous parents and a weight vector $\boldsymbol{\beta}(\boldsymbol{x})$.

As an example let us consider the energy costs E_c^c as a continuous variable. The energy costs depend on the temperature τ. The Season is regarded as parent of the temperature τ. The Bayesian network is depicted in figure 2.7. To distinguish discrete nodes from continuous nodes, the discrete nodes are drawn as rectangle or square, the continuous nodes are drawn as

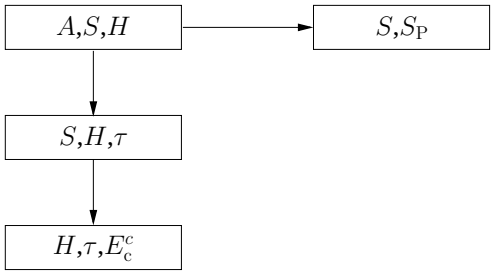


Figure 2.8: Junction tree for the example depicted in figure 2.7

$p(\tau\|S)$	$\alpha(S)$	$\gamma(S)$
$S =$'spr'	12	5
$S =$'sum'	20	5
$S =$'fal'	12	5
$S =$'win'	5	5

Table 2.10: Distribution for node τ

$p(E_c^c\|H,\tau)$	$\alpha(H)$	$\beta(H)$	$\gamma(H)$
$H =$'off'	300	-0.2	100
$H =$'on'	500	-20	100

Table 2.11: Distribution for node E_c^c

ellipse or circle. The dashed lines are not part of the Bayesian network. They are added during moralization. No additional links are added during triangulation, which results in the junction tree depicted in figure 2.8. The conditional probabilities are defined as before in tables 2.1 – 2.5. For the new continuous nodes the parameters in tables 2.10 and 2.11 are used. They are selected so that the model reflects a sensible behavior, e.g. the mean temperature in summer is higher than the mean temperature in winter.

The most frequently used inference algorithms for hybrid Bayesian networks are both based on a junction tree. The first steps towards a junction tree, moralization of the BN and triangulation, are nearly identical to the steps for discrete BNs. The only difference is that the triangulated graph is not allowed to contain a continuous path $X_1 - \cdots - Y_i - \cdots - X_2$ between two non-neighbored, discrete nodes X_1 and X_2. A good overview about triangulation algorithms for discrete BNs is given in [Kjæ90], the proceeding for hybrid BNs is described in [Ole93] and [JJD94]. Using the triangulated graph a junction tree is calculated which has as special property a strong root. The strong root is important for marginalization during message passing.

A clique $\mathbb{C}_{\boldsymbol{R}}$ in a junction tree is a *strong root* if any pair $\mathbb{C}_{\boldsymbol{A}}, \mathbb{C}_{\boldsymbol{B}}$ of neighbors on the tree with $\mathbb{C}_{\boldsymbol{A}}$ closer to $\mathbb{C}_{\boldsymbol{R}}$ than $\mathbb{C}_{\boldsymbol{B}}$ satisfies

$$(\mathbb{C}_{\boldsymbol{B}} \setminus \mathbb{C}_{\boldsymbol{A}}) \subseteq \boldsymbol{\Gamma}_{\mathcal{G}} \ \vee \ (\mathbb{C}_{\boldsymbol{B}} \cap \mathbb{C}_{\boldsymbol{A}}) \subseteq \boldsymbol{\Delta}_{\mathcal{G}}. \tag{2.38}$$

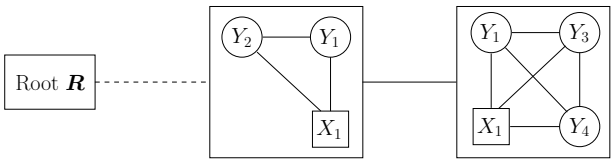


Figure 2.9: Two cliques with a hybrid separator

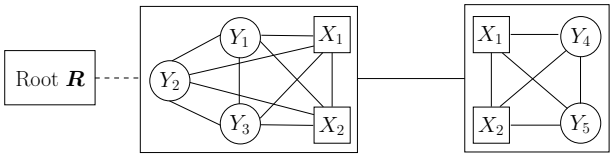


Figure 2.10: Two cliques with a discrete separator

According to Leimer [Lei89] the cliques of a decomposable marked graph can always be transformed in a junction tree with at least one strong root.

Neighboring cliques which might occur in a junction tree are depicted in figure 2.9 and 2.10.

In figure 2.8 the clique $\{A, S, H\}$ can be used as strong root. The separator between $\{A, S, H\}$ and $\{S, S_{\mathrm{P}}\}$ is a subset of $\boldsymbol{\Delta}_{\mathcal{G}}$. When $\{S, H, \tau\} = \mathbb{C}_{\boldsymbol{A}}$ and $\{H, \tau, E_{\mathrm{c}}^c\} = \mathbb{C}_{\boldsymbol{B}}$, the set difference $\mathbb{C}_{\boldsymbol{B}} \setminus \mathbb{C}_{\boldsymbol{A}} = \{H, \tau, E_{\mathrm{c}}^c\} \setminus \{S, H, \tau\} = \{E_{\mathrm{c}}^c\} \subset \boldsymbol{\Gamma}_{\mathcal{G}}$.

Using a junction tree with strong root guarantees that the separator potentials calculated during collectEvidence are always CG-potentials. When calculating the separator potentials in the other direction usually a CG potential with the same mean and dispersion is used instead.

After the construction of the junction tree, each node of the BN is assigned to one clique in the junction tree. That means the potential of the node is multiplied with the clique's potential. The representation of the potential depends on the used inference algorithm. The first one is described in [Lau92]. It works similarly to the algorithm for discrete BNs; i.e., the joint distribution is calculated by a division of the clique potentials by the separator potentials. This algorithm uses two different potential representations at the same time. The transformation from one representation scheme to the other includes the calculation of an inverse matrix. Sometimes this leads to numerical instability. To avoid numerical instability, Lauritzen introduced a second inference algorithm in [LJ99] which distinguishes between so called head and tail variables. A potential is proportional to a distribution of the head variables given the tail variables.

2.3.2 Inference in hybrid Bayesian networks

Moment and canonical characteristic

Up to now the strong root of the junction tree is discussed. The junction tree is used by both algorithms. In analogy to discrete BNs, each variable X is assigned to a clique $\mathbb{C}$, so that $\mathbb{F}(X)) \subseteq \mathbb{C}$. The potential of each variable is represented by the *moment characteristic*; i.e. by its mean $\boldsymbol{\xi}(\boldsymbol{x})$, the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{x})$, and the probability of the configurations $P(\boldsymbol{x})$. The distribution of continuous nodes $\boldsymbol{Y}$, given the configuration $\boldsymbol{x}$, is equal to

$$p(\boldsymbol{Y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{\xi}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x})) \tag{2.39}$$

whenever $P(\boldsymbol{x}) > 0$. Beside the moment characteristic it is possible to represent a CG-distribution by the *canonical characteristic*, i.e. by a potential

$$\phi(\boldsymbol{x}, \boldsymbol{y}) = \chi(\boldsymbol{x}) \exp(g(\boldsymbol{x}) + \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{y} - \boldsymbol{y}^T \boldsymbol{K}(\boldsymbol{x}) \boldsymbol{y}/2) \tag{2.40}$$

where $\chi(\boldsymbol{x})$ denotes an indicator function which is equal to 1 iff $P(\boldsymbol{x}) > 0$, and $g(\boldsymbol{x})$ is a real number. The length of the vector $\boldsymbol{h}$ is equal to the length of $\boldsymbol{y}$. The symmetric matrix $\boldsymbol{K}$ is the inverse of the covariance $\boldsymbol{\Sigma}$. Sometimes $\chi(\boldsymbol{x})$ is omitted and $P(\boldsymbol{x}) > 0$ is required instead. As both representation schemes are used for different operations during inference process, a transformation

$$\begin{aligned}
P(\boldsymbol{x}) &\propto (\det(\boldsymbol{\Sigma}(\boldsymbol{x})))^{\frac{1}{2}} \exp(g(\boldsymbol{x}) + \boldsymbol{h}(\boldsymbol{x})^T \boldsymbol{\Sigma}(\boldsymbol{x}) \boldsymbol{h}(\boldsymbol{x})/2) && (2.41)\\
\boldsymbol{\xi}(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x})^{-1} \boldsymbol{h}(\boldsymbol{x}) && (2.42)\\
\boldsymbol{\Sigma}(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x})^{-1} && (2.43)\\
\boldsymbol{K}(\boldsymbol{x}) &= \boldsymbol{\Sigma}(\boldsymbol{x})^{-1} && (2.44)\\
\boldsymbol{h}(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x}) \boldsymbol{\xi}(\boldsymbol{x}) && (2.45)\\
g(\boldsymbol{x}) &= \log(P(\boldsymbol{x})) + \frac{\log(\det(\boldsymbol{K}(\boldsymbol{x}))) - \mid \Gamma \mid \log(2\pi) - \boldsymbol{\xi}(\boldsymbol{x})^T \boldsymbol{K}(\boldsymbol{x}) \boldsymbol{\xi}(\boldsymbol{x})}{2} && (2.46)
\end{aligned}$$

between the moment and the canonical characteristic has to be defined. The number of continuous nodes in the potential is denoted by $\mid \Gamma \mid$.

Initialization

The canonical characteristic is used for initialization

$$g(\boldsymbol{x}) = -\frac{\alpha(\boldsymbol{x})^2}{2\gamma(\boldsymbol{x})} - \frac{\log(2\pi\gamma(\boldsymbol{x}))}{2} \tag{2.47}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \frac{\alpha(\boldsymbol{x})}{\gamma(\boldsymbol{x})} \begin{pmatrix} 1 \\ -\boldsymbol{\beta}(\boldsymbol{x}) \end{pmatrix} \tag{2.48}$$

$$\boldsymbol{K}(\boldsymbol{x}) = \frac{1}{\gamma(\boldsymbol{x})} \begin{pmatrix} 1 & -\boldsymbol{\beta}(\boldsymbol{x})^T \\ -\boldsymbol{\beta}(\boldsymbol{x}) & -\boldsymbol{\beta}(\boldsymbol{x})\boldsymbol{\beta}(\boldsymbol{x})^T \end{pmatrix} \tag{2.49}$$

using the parameters given by the definition of the BN.

During the initialization of the junction tree in figure 2.8 the random variable E_c^c has to be assigned to the clique $\{H, \tau, E_c^c\}$. This clique is the only one which contains the parents $\mathbb{P}(E_c^c) = \{H, \tau\}$. The node τ has to be assigned to the clique $\{S, H, \tau\}$. For the initialization of $\phi_{\{H,\tau,E_c^c\}}$ the distribution $\mathcal{N}(E_c^c|H,\tau)$ is transformed in a potential. Using the parameters of table 2.11 results in the following canonical characteristic for the potential $\phi_{\{H,\tau,E_c^c\}}$.

$$\begin{aligned} g(H='off') &= -\frac{300^2}{2\cdot 100} - \frac{\log(2\pi\cdot 100)}{2} \\ &= -450 - 3.22 = -453.22 \\ \boldsymbol{h}(H='off') &= \frac{300}{100}\begin{pmatrix} 1 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 3 \\ 0.6 \end{pmatrix} \\ \boldsymbol{K}(H='off') &= \frac{1}{100}\begin{pmatrix} 1 & 0.2 \\ 0.2 & -0.04 \end{pmatrix} \\ &= \frac{1}{100}\begin{pmatrix} 0.01 & 0.002 \\ 0.002 & -4\cdot 10^{-4} \end{pmatrix} \end{aligned}$$

Provided the domains $\mathrm{Dom}(\phi_1) = \mathrm{Dom}(\phi_2)$ of two arbitrary potentials ϕ_1 and ϕ_2 are equal, the canonical characteristic can be used directly for the multiplication and division of potentials

$$(g_1, \boldsymbol{h}_1, \boldsymbol{K}_1) \times (g_2, \boldsymbol{h}_2, \boldsymbol{K}_2) = (g_1 + g_2, \boldsymbol{h}_1 + \boldsymbol{h}_2, \boldsymbol{K}_1 + \boldsymbol{K}_2) \ . \tag{2.50}$$

Usually the precondition of equal domains is not given, thus an *expansion* of the potentials from

e.g. $\phi_{\{\boldsymbol{X}_1,\boldsymbol{Y}_1\}}(\boldsymbol{x}_1,\boldsymbol{y}_1)$ to $\phi_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}_1,\boldsymbol{Y}_2\}}(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}_1,\boldsymbol{y}_2)$

$$g_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}_1,\boldsymbol{Y}_2\}}(\boldsymbol{x}_1,\boldsymbol{x}_2) = g_{\{\boldsymbol{X}_1,\boldsymbol{Y}_1\}}(\boldsymbol{x}_1) \tag{2.51}$$

$$\boldsymbol{h}_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}_1,\boldsymbol{Y}_2\}}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \begin{pmatrix} \boldsymbol{h}_{\{\boldsymbol{X}_1,\boldsymbol{Y}_1\}}(\boldsymbol{x}_1) \\ 0 \end{pmatrix} \tag{2.52}$$

$$\boldsymbol{K}_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}_1,\boldsymbol{Y}_2\}}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \begin{pmatrix} \boldsymbol{K}_{\{\boldsymbol{X}_1,\boldsymbol{Y}_1\}}(\boldsymbol{x}_1,\boldsymbol{x}_2) & 0 \\ 0 & 0 \end{pmatrix} \tag{2.53}$$

may be necessary to guarantee equal domains. Using expansion and multiplication, an initial potential of all cliques in the junction tree is calculated. The separators are initialized with $\phi_{\mathbb{S}} = 1$, i.e. $\phi_{\mathbb{S}} = (0,0,0)$.

In the junction tree depicted in figure 2.8 the three variables A, S, and H are assigned to the clique $\{A,S,H\}$. This clique has to distinguish 16 different characteristics $(g,\boldsymbol{h},\boldsymbol{K})$, one for each configuration. For the configuration A ='yes',S ='spr', H ='on', the new potential $\phi^*_{\{A,S,H\}}$ is

$$\begin{aligned} \phi^*_{\{A,S,H\}} &= \phi_{\{A\}}(A='yes') \times \phi_{\{S\}}(S='spr') \times \\ &\quad \phi_{\{A,S,H\}}(A='yes', S='spr', H='on') \\ &= (-1.8971,0,0) \times (-1.3863,0,0) \times (-2.3036,0,0) \\ &= (-5.586,0,0)\ . \end{aligned}$$

For the other cliques no multiplication is necessary during initialization.

Marginalization and message passing

Directly after the initialization, message passing is triggered. For message passing equations (2.16) and (2.17) are used, but with a changed potential definition in comparison to the discrete case. According to equation (2.16) marginals have to be calculated during message passing. Marginalization of a CG-distribution over continuous variables results in a CG-distribution. For marginalization over a discrete variable, a sum of multiple Gaussians with different weights has to be calculated. The resulting potential is not necessarily a CG-potential. E.g. figure 2.11 shows a mixture of two Gaussians.

Marginalization of a CG-potential is only guaranteed to result in a CG-potential, if marginalizing is carried out over a continuous variable. The marginal $\phi^{\downarrow\{\boldsymbol{X},\boldsymbol{Y}_1\}}_{\{\boldsymbol{X},\boldsymbol{Y}_1,\boldsymbol{Y}_2\}} = (g',\boldsymbol{h}',\boldsymbol{K}')$ of

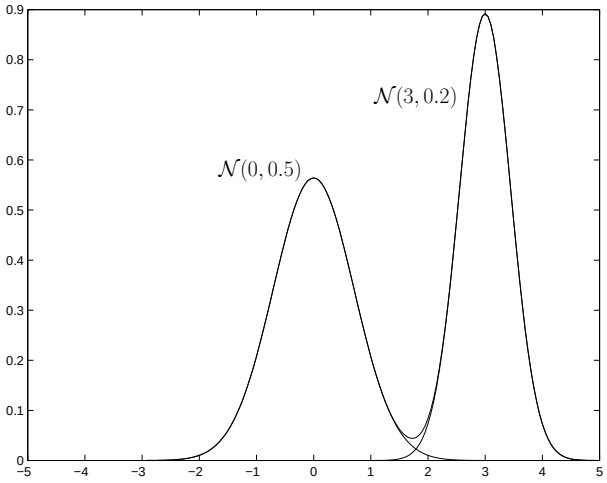


Figure 2.11: Mixture of two Gaussians

$\phi_{\{\boldsymbol{X},\boldsymbol{Y}_1,\boldsymbol{Y}_2\}} = (g, \boldsymbol{h}, \boldsymbol{K})$ with

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \end{pmatrix} \qquad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{11}\boldsymbol{K}_{12} \\ \boldsymbol{K}_{21}\boldsymbol{K}_{22} \end{pmatrix}$$

$$g'(\boldsymbol{x}) = g(\boldsymbol{x}) + \frac{\mid \boldsymbol{Y}_1 \mid \log(2\pi) - \log(\det(\boldsymbol{K}_{11}(\boldsymbol{x}))) + \boldsymbol{h}_1^T(\boldsymbol{x})\boldsymbol{K}_{11}(\boldsymbol{x})^{-1}\boldsymbol{h}_1(\boldsymbol{x})}{2} \quad (2.54)$$

$$\boldsymbol{h}'(\boldsymbol{x}) = \boldsymbol{h}_2(\boldsymbol{x}) - \boldsymbol{K}_{21}(\boldsymbol{x})\boldsymbol{K}_{11}^{-1}(\boldsymbol{x})\boldsymbol{h}_1(\boldsymbol{x}) \quad (2.55)$$

$$\boldsymbol{K}'(\boldsymbol{x}) = \boldsymbol{K}_{22}(\boldsymbol{x}) - \boldsymbol{K}_{21}(\boldsymbol{x})\boldsymbol{K}_{11}^{-1}(\boldsymbol{x})\boldsymbol{K}_{12}(\boldsymbol{x}) \quad (2.56)$$

is a so called *strong marginal* in contrary to the *weak marginal* where only mean and dispersion of the overall distribution are preserved.

Marginalization over discrete variables results in a CG-potential only if there are no continuous variables in the domain of the potential or if these continuous random variables are removed by marginalization first.

When sending messages in the direction of the strong root, a strong marginalization takes place. Calling distributeEvidence from the strong root, a weak marginalization from $\phi_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}\}} =$

$(P, \boldsymbol{\xi}, \boldsymbol{\Sigma})$ to $\phi^{\downarrow\{\boldsymbol{X}_1,\boldsymbol{Y}\}}_{\{\boldsymbol{X}_1,\boldsymbol{X}_2,\boldsymbol{Y}\}} = (P', \boldsymbol{\xi}', \boldsymbol{\Sigma}')$

$$P'(\boldsymbol{x}_1) = \sum_{\boldsymbol{x}_2} P(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{2.57}$$

$$\boldsymbol{\xi}'(\boldsymbol{x}_1) = \sum_{\boldsymbol{x}_2} \frac{\boldsymbol{\xi}(\boldsymbol{x}_1, \boldsymbol{x}_2) P(\boldsymbol{x}_1, \boldsymbol{x}_2)}{P'(\boldsymbol{x}_1)} \tag{2.58}$$

$$\boldsymbol{\Sigma}(\boldsymbol{x}_1) = \frac{\sum_{\boldsymbol{x}_2} \boldsymbol{\Sigma}(\boldsymbol{x}_1, \boldsymbol{x}_2) P(\boldsymbol{x}_1, \boldsymbol{x}_2)}{P'(\boldsymbol{x}_1)} + \frac{\sum_{\boldsymbol{x}_2} (\boldsymbol{\xi}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \boldsymbol{\xi}'(\boldsymbol{x}_1))^T (\boldsymbol{\xi}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \boldsymbol{\xi}'(\boldsymbol{x}_1)) P(\boldsymbol{x}_1, \boldsymbol{x}_2)}{P'(\boldsymbol{x}_1)} \tag{2.59}$$

may take place, where only the mean and dispersion of the CG-distribution are preserved. After sending messages in both directions, the junction tree is consistent, i.e. for two neighboring cliques $\mathbb{C}_1$ and $\mathbb{C}_2$ with separator $\mathbb{S}$

$$\phi^{\downarrow\mathbb{S}}_{\mathbb{C}_1} \approx \phi^{\downarrow\mathbb{S}}_{\mathbb{C}_2} \tag{2.60}$$

holds, where $\approx$ means that the first two moments are identical. After message passing is finished, the cliques represent the true marginals for discrete variables, for the hybrid case it is only guaranteed that mean and dispersion of a clique $\phi_{\mathbb{C}}$

$$\phi_{\mathbb{C}} \approx \phi^{\downarrow\mathbb{C}} \tag{2.61}$$

are equal to the marginal of the joint potential ϕ.

To get the junction tree in figure 2.8 globally consistent, collectEvidence is called. First the marginal $\phi^*_{\{H,\tau\}} = \phi^{\downarrow\{H,\tau\}}_{\{H,\tau,E^c_\mathrm{c}\}}$ is calculated and assigned to the separator. Directly afterwards the new separator potential $\phi^*_{\{H,\tau\}}$ is multiplied with $\phi_{\{S,H,\tau\}}$

$$\phi^*_{\{S,H,\tau\}} = \phi_{\{S,H,\tau\}} \times \phi^*_{\{H,\tau\}} \ . \tag{2.62}$$

The division by the old separator potential is omitted here as this potential is initialized to one. Afterwards the strong root $\{A, S, H\}$ is updated. First the new separator potentials

$$\phi^*_{\{S,H\}} = \phi^{*\downarrow\{S,H\}}_{\{S,H,\tau\}} \tag{2.63}$$

$$\phi^*_{\{S\}} = \phi^{\downarrow\{S\}}_{\{S,S_\mathrm{P}\}} \tag{2.64}$$

are calculated. Then the strong root gets the new potential

$$\phi^*_{\{A,S,H\}} = \phi_{\{A,S,H\}} \times \phi^*_{\{S,H\}} \times \phi^*_{\{S\}} \ . \tag{2.65}$$

Once again the division by the old separator potential is omitted, as they are equal to 1 directly after initialization. All the calculated marginals are strong marginals. Considering the property of the strong root it can be seen that during collectEvidence marginalization always leads to strong marginals. During distribute evidence weak marginals occur, when $\phi^{\downarrow\{H,\tau\}}_{\{S,H,\tau\}}$ is calculated. Here marginalization over the discrete variable Season is necessary. In this case a mixture of eight Gaussians is approximated by a mixture of two Gaussians. An equality of the potentials is therefore not possible.

Introduction of evidences

The last task is the introduction of evidences. Discrete evidences $X = x$ are entered by multiplication of the potential with an indicator function so that the potential is zero for impossible configurations. In contrast to discrete evidences, for continuous evidences $Y = y$ all cliques $\mathbb{C}$ with $\{Y\} \subseteq \mathbb{C}$ have to be changed. Suppose that a potential $\phi = (g, \boldsymbol{h}, \boldsymbol{K})$ with

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_1(\boldsymbol{x}) \\ \boldsymbol{h}_Y(\boldsymbol{x}) \end{pmatrix} \qquad \boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{1Y} \\ \boldsymbol{K}_{Y1} & \boldsymbol{K}_{YY} \end{pmatrix}$$

is simplified, by entering $Y = y$. The new potential $\phi' = (g', \boldsymbol{h}', \boldsymbol{K}')$ is calculated by removing the components for Y by

$$g'(\boldsymbol{x}) = g(\boldsymbol{x}) + \boldsymbol{h}_Y(\boldsymbol{x})y - \frac{\boldsymbol{K}_{YY}(\boldsymbol{x})y^2}{2} \tag{2.66}$$

$$\boldsymbol{h}'(\boldsymbol{x}) = \boldsymbol{h}_1(\boldsymbol{x}) - y\boldsymbol{K}_{Y1}(\boldsymbol{x}) \tag{2.67}$$

$$\boldsymbol{K}'(\boldsymbol{x}) = \boldsymbol{K}_{11}(\boldsymbol{x}). \tag{2.68}$$

Of course the introduction of evidences leads to a smaller domain of ϕ'. After entering the evidence, the junction tree is inconsistent. Thus the functions collectEvidence and distributeEvidence must be called again to come to a consistent junction tree. After message passing the clique potentials are proportional to the marginalized joint distribution.

Unfortunately it turned out that the frequent transformation between moment and canonical characteristic is numerical unstable due to the matrix inversion. This drawback is avoided in the inference algorithm introduced in [LJ99]. The main idea is to partition the set of continuous

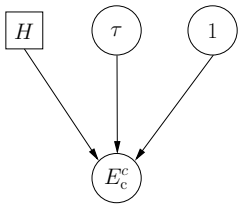


Figure 2.12: Equivalent network for the training of the node E_c^c

variables into *head* and *tail*, so that a potential represents a CG distribution of the head variables depending on the tail variables.

2.3.3 Learning the parameters of a hybrid Bayesian network

The aim is comparable to the aim in section 2.2.2, i.e. to learn the parameters of the modeled distribution. For discrete nodes conditional probability tables are learned, the technique is the same, if no arcs $X \rightarrow Y$ from discrete to continuous nodes are allowed. For the continuous nodes the mean α, the weight vector $\boldsymbol{\beta}$, and the variance γ are the parameters to be estimated. If global and local independence of the parameters are supposed, it is sufficient to maximize the parameters of each family independently. Of course, for the estimation of the instantiation, the complete Bayesian network has to be used. When assuming that every continuous node Y has at least one continuous parent $\boldsymbol{Z}$ whose last element is 1, the mean α can be modeled by the last column of the weight vector $\boldsymbol{\beta}$.

As an example see figure 2.12 which shows an equivalent network for node E_c^c, where the mean α is replaced by the weight of the link $1 \rightarrow E_c^c$.

The distribution of node Y with discrete parents X and continuous parents $\boldsymbol{Z}$ becomes

$$p(y|\boldsymbol{x},\boldsymbol{z}) = \frac{1}{\sqrt{(2\pi)\gamma}} \exp(\frac{1}{2}(y-\boldsymbol{\beta}(\boldsymbol{x})^T\boldsymbol{z})\gamma^{-1}(y-\boldsymbol{\beta}(\boldsymbol{x})^T\boldsymbol{z}) \ , \tag{2.69}$$

where γ is the variance of the normal distribution. For the more general case of a multidimensional normal distribution see [Mur98a]. For the purpose of the thesis one-dimensional distributions are sufficient.

Using an indicator function $\delta_{\boldsymbol{x}}(\boldsymbol{x}_1)$ being 1 iff $\boldsymbol{x}_1 = \boldsymbol{x}$ and 0 otherwise, the part of the

	ESS	Remark
$d_{\boldsymbol{x}}$	$\sum_{j=1}^{N} \delta_{\boldsymbol{x}}^j$ with $\delta_{\boldsymbol{x}}^j = E[\delta_{\boldsymbol{x}}(\boldsymbol{x}^j)\|\boldsymbol{o}^j]$	
$S_{YY,\boldsymbol{x}}$	$\sum_{j=1}^{N} \delta_{\boldsymbol{x}}^j E_{\boldsymbol{x}}[(y^j)^2\|\boldsymbol{o}^j]$	
$S_{ZY,\boldsymbol{x}}$	$\sum_{j=1}^{N} \delta_{\boldsymbol{x}}^j E_{\boldsymbol{x}}[\boldsymbol{z}^j y^j\|\boldsymbol{o}^j]$	Only if Y has continuous parents
$S_{ZZ^T,\boldsymbol{x}}$	$\sum_{j=1}^{N} \delta_{\boldsymbol{x}}^j E_{\boldsymbol{x}}[\boldsymbol{z}^j(\boldsymbol{z}^j)^T\|\boldsymbol{o}^j]$	Only if Y has continuous parents
$S_{Y,\boldsymbol{x}}$	$\sum_{j=1}^{N} \delta_{\boldsymbol{x}} E_x[y\|\boldsymbol{o}^j]$	Only if Y has no continuous parents

Table 2.12: Essential Sufficient Statistics for training of hybrid Bayesian networks

log-likelihood depending on the parameters of Y is

$$L = \log(\prod_{j=1}^{N} \prod_{\boldsymbol{x}} [p(y^j|\boldsymbol{z}^j, \boldsymbol{x}^j, \boldsymbol{o}^j)]^{\delta_{\boldsymbol{x}}(\boldsymbol{x}^j)}) \ . \tag{2.70}$$

The vector $\boldsymbol{o}$ denotes the N observations, but all random variables y, $\boldsymbol{z}$ and $\boldsymbol{x}$ may be unobserved. Once again the problem is that an estimation of the parameters requires the knowledge of the instantiation of the unobserved variables and vice versa. For the solution the EM-algorithm might be used. It starts with an arbitrary parameter-set $\boldsymbol{\theta}^{(0)}$. This is used to calculate the estimated log-likelihood

$$L = -\frac{1}{2} \sum_{j=1}^{N} E[\sum_{\boldsymbol{x}} \delta_{\boldsymbol{x}(\boldsymbol{x}^j)} \log(\gamma(\boldsymbol{x})) + \delta_{\boldsymbol{x}}(\boldsymbol{x}^j)(y^j - \boldsymbol{\beta}(\boldsymbol{x})^T \boldsymbol{z}^j)\gamma(\boldsymbol{x})^{-1}(y^j - \boldsymbol{\beta}(\boldsymbol{x})^T \boldsymbol{z}^j)|\boldsymbol{o}^j]. \tag{2.71}$$

To keep track of the N training examples, the *essential sufficient statistics*(ESS), listed in table 2.12 are used.

The index $\boldsymbol{x}$ in $E_{\boldsymbol{x}}$ is used to denote the expectation given that $\boldsymbol{X} = \boldsymbol{x}$. The values used by these formulas are calculated by entering $\boldsymbol{o}$ as evidence and then using the junction tree algorithm to calculate the missing values.

Using the statistics the new estimations

$$\hat{\boldsymbol{\beta}}(\boldsymbol{x})^T = S_{ZY,\boldsymbol{x}}^T S_{ZZ^T,\boldsymbol{x}}^{-1} \tag{2.72}$$

$$\hat{\gamma}(\boldsymbol{x}) = \frac{S_{YY,\boldsymbol{x}}}{d_{\boldsymbol{x}}} - \frac{\hat{\boldsymbol{\beta}}(\boldsymbol{x})^T S_{ZZ^T,\boldsymbol{x}}}{d_{\boldsymbol{x}}} \tag{2.73}$$

$$\hat{\alpha}(\boldsymbol{x}) = \frac{S_{Y,\boldsymbol{x}}}{d_{\boldsymbol{x}}} \tag{2.74}$$

H	τ	E_{c}^c	1
$'on'$	10	493	1
$'on'$	13	457	1
$'off'$	27	107	1
$'off'$	19	209	1
$'off'$	22	153	1

Table 2.13: Numbers used for the training of the parameters of node E_{c}^c

are calculated. The Bayesian network is used to calculate the expected values of the unobserved random variables given the parameter of the last M-step. These expected values are used to calculate the ESS. When all training cases are included in the ESS, the parameters are maximized using equations (2.72) to (2.74), where the last one is used for continuous random variables without continuous parents. For a discussion of the multidimensional case see [Mur98a].

As an example we regard the training of E_{c}^c, with $\mathbb{F}(E_{\mathrm{c}}^c) = \{H, \tau, E_{\mathrm{c}}^c\}$. To simplify the example, we assume that all variables are observed so that we can ignore the other random variables of the example and one EM-step is sufficient to train the parameters. As training cases the values in table 2.13 are observed.

The observations of table 2.13 results in the following statistics

$$\begin{aligned}
d_{H='on'} &= 2 \\
d_{H='off'} &= 3 \\
S_{YY,H='on'} &= 493^2 + 457^2 = 451898 \\
S_{ZY,H='on'} &= [10 \quad 1]493 + [13 \quad 1]457 = [10871 \quad 950] \\
S_{ZZ^T,H='on'} &= \begin{bmatrix} 10 \\ 1 \end{bmatrix} \begin{bmatrix} 10 & 1 \end{bmatrix} + \begin{bmatrix} 13 \\ 1 \end{bmatrix} \begin{bmatrix} 13 & 1 \end{bmatrix} = \begin{bmatrix} 269 & 23 \\ 23 & 2 \end{bmatrix} .
\end{aligned}$$

This ESS results in the following new parameters for the node E_{c}^c

$$\begin{aligned}
\hat{\boldsymbol{\beta}}^T &= [-12 \quad 613] \\
\hat{\gamma} &= 0 \ .
\end{aligned}$$

The results is obviously correct as there are two parameters for two examples so that a variance equal to zero meets expectation.

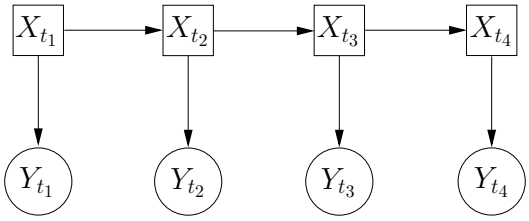


Figure 2.13: A Hidden Markov Model as an example of a Dynamic Bayesian network

2.4 Dynamic Bayesian networks

The networks discussed in the previous sections had one common trait, namely there is only one random variable for each measurement. This is sufficient for some of the engineering processes discussed in this thesis, but for many purposes it is not. Particularly for the analysis of time series, for medical purposes (e.g. the regular measurement of a patients fever), or for dynamic processes described by differential equations, it is necessary to represent the measured data at different points in time. So information about the past can be used for the prediction of the future.

The main idea is to represent each point in time by a separate Bayesian network, called time-slices. These time-slices are linked together by temporal edges (cf. [Kjæ92] [Kjæ93]) or inter slice connections. Usually the Markov assumption holds

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{t-1} \perp\!\!\!\perp \boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_{t+k} | \boldsymbol{x}_t, \tag{2.75}$$

i.e. the future is conditionally independent of the past given the present. The nodes X_t having links to neighbored time slices are called the *interface* nodes. Kjærulff defines the interface as all nodes with incoming links from previous time slices [Kjæ92], whereas K. P. Murphy [Mur02] distinguishes between forward and backward interface where the forward interface of a time-slice contains all nodes with an edge to the next time-slice. In the backward interface all nodes with an incoming edge from the previous time-slice are included.

As an example the HMM in figure 2.13 can be regarded. Here the discrete state nodes X_t are in the forward interface of time slice t.

Usually the nodes representing the same random variable at different points in time, e.g. the nodes Y_{t_2}, Y_{t_3}, and Y_{t_4} have the same parameters. An exception are sometimes the nodes of the first time-slice. In figure 2.13 X_{t_1} has no incoming links, thus the node represents a probability $P(X_{t_1})$ which is in most of the cases different from $P(X_t|X_{t-1}), t_2 \leq t \leq t_4$. So the first

time-slice represents the a-priori distribution, the other time-slices the distribution of the random variables at time t, given the information at time $t-1$. This leads to the following definition of a DBN, taken from [Mur02]

A DBN is defined to be a pair (BN_1,$BN_{\rightarrow}$, where BN_1 is a BN which defines the prior and $BN_{\rightarrow}$ is a two slice temporal Bayes net, which defines $P(X_t|X_{t-1})$ by means of a DAG as follows

$$P(X_t|X_{t-1}) = \prod_{i=1}^{n_N} P(X_{i,t}|\mathbb{P}(X_{i,t})) \ , \tag{2.76}$$

where $X_{i,t}$ is the i-th node at time t. In this definition X is either a discrete or a continuous random variable.

There are a lot of different inference algorithms. The simplest one, used for the experiments discussed in chapter 4, is to unroll the DBN to $t_{\max}$ time-slices, so that it becomes a hybrid Bayesian network as discussed in section 2.3.1 with $n_{ts}t_{\max}$ nodes, where n_{ts} denotes the number of nodes in each time slice. E.g. figure 2.13 can be regarded as a DBN with $n_{ts} = 2$ unrolled to 4 time-slices. The advantage is that the same inference algorithm as discussed in section 2.3.1 can be applied. The disadvantage is the fixed number of time slices.

To deal with a varying number of time slices Kjærulff [Kjæ92; Kjæ93] suggests a reorganization of the junction tree. Murphy discusses several inference algorithms in his thesis [Mur02], one of them is the interface algorithm. The first step is the organization in a junction tree with the restriction that all interface nodes are in one clique. This can be forced by adding additional links to the moralized graph. After triangulation there is one clique which separates the time-slices from each other. For example see figure 2.14 which shows the junction tree resulting from figure 2.13.

Now each time slice is represented by a junction tree, which contains additional nodes from the previous time-slice. This junction tree contains exactly one clique which is linked to the previous time slice and one clique linked to the next time slice. As figure 2.14 shows, the cliques to the previous and to the next time slice might collapse to one clique.

The interface algorithm starts by calling collectEvidence for the clique containing the nodes of the forward interface, first for the junction tree J_1, which represents the nodes of the first time slice. Afterwards the cliques with the forward interface in J_2 to $J_{t_{\max}}$ call collectEvidence. This results in an absorption of knowledge from the previous time slice.

In the backward pass, the forward interface cliques are calling distribute evidence, starting in the junction tree $J_{t_{\max}}$. This step includes the distribution of the knowledge to the previous time-slice.

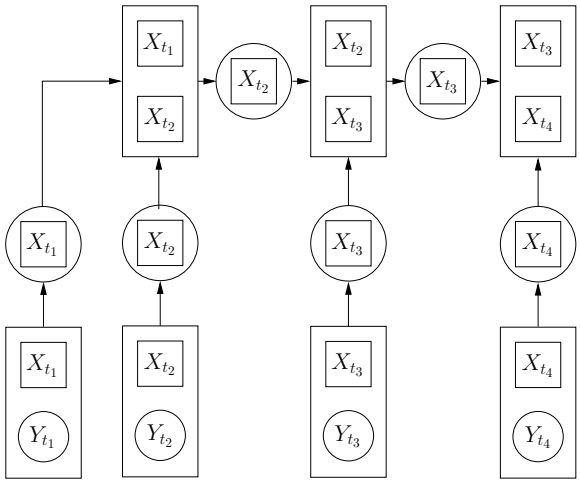


Figure 2.14: Junction tree of the HMM in figure 2.13

In our example first collectEvidence is called by the cliques $\{X_{t_1}, X_{t_2}\}, \{X_{t_2}, X_{t_3}\}$ and finally by $\{X_{t_3}, X_{t_4}\}$. This finishes the forward pass. During the backward pass distributeEvidence is called by $\{X_{t_3}, X_{t_4}\}, \{X_{t_2}, X_{t_3}\}$ and at the end by $\{X_{t_1}, X_{t_2}\}$.

In the discussed example all separators are discrete ones, thus all calculated marginals are strong. Usually this can not be guaranteed so that week marginals might occur in both the forward and the backward pass.

In the next section the basics of control theory are introduced. After the introduction to control theory is finished, the application of dynamic Bayesian networks as controller is discussed.

Chapter 3

Control of dynamic systems

The overall aim is to study the question whether Bayesian networks can be used for control purposes. To show the limits and preconditions for the usage of Bayesian controller, this chapter starts with a characterization of dynamic systems. Afterwards linear systems, together with the theory of Kalman Filters, are discussed. The knowledge about linear systems leads to a general structure of Bayesian networks and suitable parameter settings. For non-linear units, prototypical models are provided for some frequently occurring transfer units. For a discussion of non-linear systems see section 3.2. In section 3.3 traditional control methods and criteria to judge the quality of control are introduced. These criteria allow a comparison between traditional and Bayesian controller.

Generally each dynamic system, denoted as transfer unit φ, maps an input function $u(t)$ to an output function $y^m(t)$

$$y^m(t) = \varphi[u(t)]. \tag{3.1}$$

Firstly, we restrict ourself to *time invariant* systems. In time invariant systems a delayed input signal $u(t - t_0)$ leads to the same, delayed output signal

$$y^m(t - t_0) = \varphi[u(t - t_0)]. \tag{3.2}$$

Even if equation (3.2) does not hold exactly, it is a common approach in parameter estimation to assume time invariance. In practice this leads to satisfying results, if changes of the plant are much slower than the adaptation of the controller (cf. [SL91]). Thus, from now on it is assumed that dynamic systems are time invariant.

The second property of dynamic systems, which simplifies analysis and control, is *linearity*.

A transfer unit φ is linear if and only if for arbitrary constants c_1 and c_2

$$\varphi[c_1 u_1(t) + c_2 u_2(t)] = c_1 \varphi[u_1(t)] + c_2 \varphi[u_2(t)] \tag{3.3}$$

holds.

In the next two sections the descriptions of linear and non-linear systems, as they are usually used in control theory, are discussed. The aim of the discussion is to guarantee that the new approach, introduced in this thesis, is applicable to a wide range of systems. The main idea is to model typical linear and non-linear systems, and show how these systems respectively the developed models may be combined to more complex systems.

The knowledge about typical non-linear units also demonstrates the theoretical limits of Bayesian models for the modeling of dynamic processes. Of course these limits are not identical with the limits of a stochastic approach in general, as Bayesian networks are restricted to a mixture of Gaussians. Using particle filters [TL98; IB98] may open a complete new range of applications.

3.1 Description of linear dynamic systems

This section discusses two different approaches for the mathematical description of linear, dynamic systems. Both of them correspond to a special Bayesian network. The first discussed description is the state-space description, which is also used for multiple input multiple output systems (MIMO). The description by difference equation leads to Bayesian networks with less hidden nodes and thus shows a better training behavior.

A dynamic system may be regarded as a black box with several input and output signals u and y^m respectively. We distinguish the output of the model y^m and the observed output q, where the latter includes also the influences of the disturbance variable z^d.

The current output of the dynamic system does not depend solely on the input signal, but also on an internal state $\boldsymbol{X}^s$ or on former in- and output signals. Linear, time invariant systems with one-dimensional in- and output, are regularly described by differential equations

$$\sum_{i=0}^{n} a_i^c \frac{d^i y^m(t)}{dt^i} = \sum_{j=0}^{m} b_j^c \frac{d^j u(t)}{dt^j} \quad . \tag{3.4}$$

Only systems with $m \leq n$ are physical realizable. As a simple example a car with mass M, accelerated by a force F and slowed down by friction and a spring, is discussed. The friction is

proportional to the product of a constant b and the velocity $v = \frac{ds}{dt}$, the excursion s of the spring causes a force ks. Thus

$$M\frac{d^2s(t)}{dt^2} + b\frac{ds}{dt} + ks = F(t) \tag{3.5}$$

holds. By substitution $x_1 = \frac{ds}{dt}$ and $x_2 = s$ the example can be transformed to a system of differential equations of first order

$$\begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{b}{M} & -\frac{k}{M} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{M} \\ 0 \end{bmatrix} F(t) \tag{3.6}$$

$$s(t) = x_2(t), \tag{3.7}$$

or generally

$$\frac{d\boldsymbol{x}^s(t)}{dt} = \boldsymbol{A}\boldsymbol{x}^s + \boldsymbol{B}\boldsymbol{u}(t) \tag{3.8}$$

$$\boldsymbol{y}^m(t) = \boldsymbol{C}\boldsymbol{x}^s + \boldsymbol{D}\boldsymbol{u}(t) \ . \tag{3.9}$$

Equations (3.8) and (3.9) are called the state-space description of a dynamic system. The matrix $\boldsymbol{A}$ at the left hand side of equation (3.8) describes the trajectory of the state $\boldsymbol{x}^s$. For a homogeneous system, i.e. $\boldsymbol{u}(t) = 0$, the description of the system by the transfer matrix $\boldsymbol{A}$ and the current state $\boldsymbol{x}_0^s$ is sufficient to calculate the state of the system in the future. The influence of the input $\boldsymbol{u}$ on the state $\boldsymbol{x}^s$ is described by the input matrix $\boldsymbol{B}$. The first equation of the state-space description describes therefore the state transition influenced by the input $\boldsymbol{u}$.

The modeled output of a system depends in many cases only on the state $\boldsymbol{x}^s$ as described by the output matrix $\boldsymbol{C}$, for jump Markov systems it depends also on the input $\boldsymbol{u}$.

The state-space description is also used to describe multiple-input multiple-output (MIMO) systems. In the general case $\boldsymbol{A}$ is a $n \times n$ matrix, where n is equal to the order of the system, the state vector $\boldsymbol{x}^s$ is $n \times 1$ vector. The $n \times r$ input matrix $\boldsymbol{B}$ describes the reaction of the r dimensional input. The $o \times n$-dimensional output matrix $\boldsymbol{C}$ depends on the dimension o of the output. The feedthrough matrix $\boldsymbol{D}$ is of dimension $o \times r$.

In reality all dynamic systems are exposed to disturbing influences $\boldsymbol{z}^{d'}$. For linear systems it is possible to add the system's reaction to all disturbance variables to one variable $\boldsymbol{z}^d$ which has an effect on the output of the system. Thus the observed output $\boldsymbol{q}$ is the sum of the model's output and the disturbance variable.

$$\boldsymbol{q}(t) = \boldsymbol{y}^m(t) + \boldsymbol{z}^d(t) \tag{3.10}$$

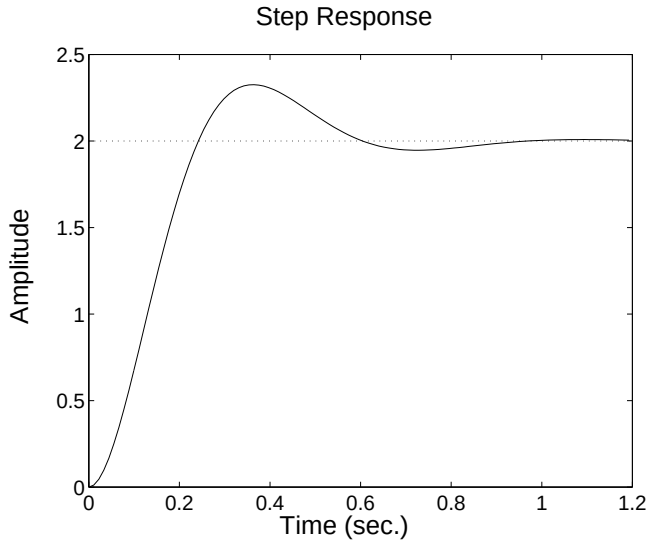


Figure 3.1: Step response of a second order system

The dynamic system is displayed as dashed box in figure 3.7. The manipulation reaction describes the response of the system to the input signal $\boldsymbol{u}(t)$, e.g. the reaction to the impulse function. Mathematically the manipulation reaction can be described by the state-space description.

The disturbance reaction defines the reaction on, usually uncontrolled, environmental influences $\boldsymbol{z}^d(t)$. It is the task of a controller to reduce the effect of a disturbance as fast as possible.

To get a good impression of a dynamic system, it is helpful to regard the response to different input signals, e.g. the *step function*

$$\sigma(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases} \tag{3.11}$$

or the *impulse function*

$$u(t) = \begin{cases} \frac{1}{\delta} & \text{if } 0 \leq t < \delta \\ 0 & \text{if } t \geq \delta \end{cases} , \tag{3.12}$$

which can approximately be regarded as a very short impulse of duration δ. Figure 3.1 displays the step response of a system described by $2u(t) = y^m + 0.1\frac{dy^m}{dt} + 0.01\frac{d^2y^m}{dt^2}$ which is one of our test systems. The impulse response of the same system is shown in figure 3.2. A second order system may overshoot and needs a long time to converge to a new value. In this thesis we will show how to calculate an input signal, so that overshooting is avoided and the output settles quickly to its new value. This method is based on Kalman filters as described in section 3.1.2.

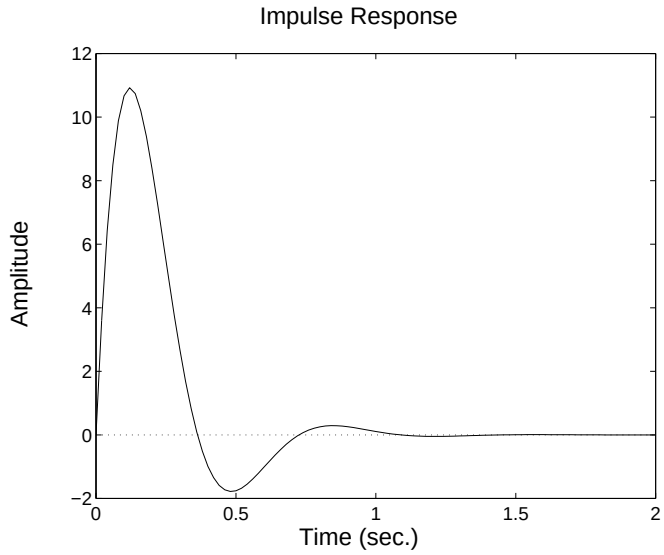


Figure 3.2: Impulse response of a second order system

The example of a second order system is now used to discuss the *natural angular frequency* and the damping D of a dynamic system. Generally, a second order delay element is described by

$$K_s u(t) = T_2^2 \frac{d^2 y^m(t)}{dt^2} + T_1 \frac{dy^m(t)}{dt} + y^m(t) \quad . \tag{3.13}$$

The factor K_s describes the gain of the system, that is the quotient of the output by a constant input signal. T_1 and T_2 are two time constants. The ratio of T_1 and T_2 describes the *damping* ratio [LW00; Unb97a]

$$D = \frac{T_1}{2T_2} \quad . \tag{3.14}$$

The *natural angular frequency*

$$\omega_0 = \frac{1}{T_2} \tag{3.15}$$

describes the oscillation behavior. For an undamped system, i.e. $D = 0$, the *resonance frequency*

$$\omega_r = \omega_0 \sqrt{1 - 2D^2} \tag{3.16}$$

is equal to the natural angular frequency.

According to the damping five different systems are distinguished. They show different step responses.

Overdamped systems with $D > 1$ converge to their new output without any overshoot. There

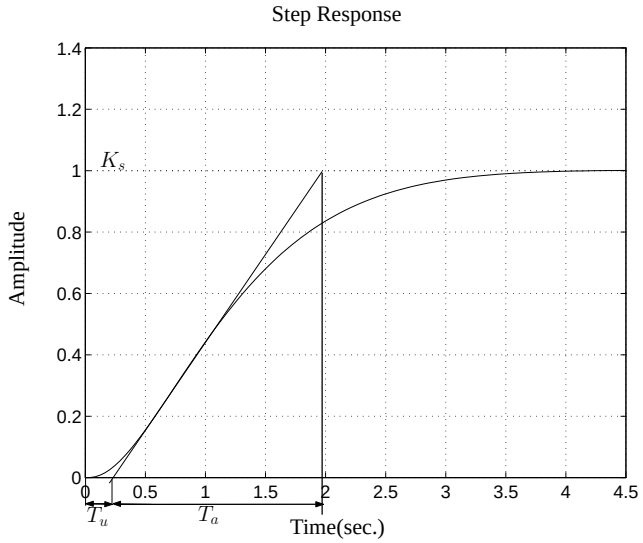


Figure 3.3: Calculation of delay time T_u and raise time T_a

are different parameters to characterize overdamped systems. E.g. the step response is characterized by the gain K_s, the delay time T_u, and the rise time T_a. The time constants T_u and T_a are calculated as depicted in figure 3.3. In a simple approach by Ziegler and Nichols they are used to determine the parameters of a controller.

Critically damped systems with D = 1 show a faster increase of the output than systems with higher damping. If the damping D is between 0 and 1, an overshoot together with an oscillation with decreasing amplitude is observed, as shown in figure 3.1. When a damping of 0 is reached the amplitude of the oscillation keeps constant. Further decreasing of the damping to a value between -1 and 0 leads to an unstable system whose step response has an increasing amplitude.

For the more general case of n-th order system the *Laplace transformation* is helpful which maps a function $f(t)$ to

$$F(s) = \int_0^\infty f(t) \exp(-st) dt \ . \tag{3.17}$$

The quotient $G(s)$ of the Laplace transformed out- and input signal $Y^m(s)$ respectively $U(s)$

$$G(s) = \frac{Y^m(s)}{U(s)} \tag{3.18}$$

is called *transfer function* of the dynamic system. Linear single-input single-output(SISO) sys-

tems are described by

$$G(s) = \frac{b_0^c + b_1^c s + \cdots b_m^c s^m}{a_0^c + a_1^c s + \cdots + a_n^c s^n} = \frac{B(s)}{A(s)} \tag{3.19}$$

the quotient of two polynomials $A(s)$ and $B(s)$.

Sometimes the described system reacts with a delay T_d, called *dead-time*, to changes of the input. This is modeled by an additional term e^{-T_d}, so that

$$G(s) = \frac{B(s)}{A(s)} e^{-T_d} \quad . \tag{3.20}$$

Please note that $A(s)$ and $B(s)$ are used for the description of the system, whereas $Y^m(s)$ and $U(s)$ denote the Laplace-transformed in- and output signals. To get an impression of the behavior of the dynamic system the homogeneous system, i.e. $u(t) = 0$, is regarded. In the Laplace transformed this leads to $U(s) = 0$ and thus to

$$A(s) = a_0^c + a_1^c s + \cdots a_n^c s^n = 0 \quad , \tag{3.21}$$

usually called the *characteristic equation*.

If all zeros s_k with $A(s_k)$ = 0 are in the left half s-plane, i.e. the real part $Re(s_k) < 0$, the system is stable. If at least one s_k has a real part $Re(s_k) > 0$ or if a multiple pole is placed on the imaginary axis, i.e. $Re(s_k) = 0$, the system is unstable.

In equations (3.8) and (3.9) $\boldsymbol{A}$ is a $n \times n$ matrix, where n denotes the order of the differential equation. For single input single output systems, i.e. both y^m and u are scalars instead of vectors, $\boldsymbol{B}$ and $\boldsymbol{C}$ are vectors of length n and $\boldsymbol{D}$ is a scalar. Thus, there are $n^2 + 2n + 1$ parameters in the state space description. Comparing the number of parameters in the state space description with the maximal number of parameters in differential equation (3.4) leads to the conclusion that there are many possible state space descriptions for the same differential equation. Reducing the number of parameters would result in a smaller search space of possible state space descriptions, which means a more robust and effective learning process. That is exactly what is done by normal forms.

3.1.1 Normal forms

As mentioned in the last paragraph of section 3.1, there is no unique state space description for a given differential equation. As a simple example imagine a dynamic system with gain K_s. To model such a system the amplification can either take place between the input $\boldsymbol{u}$ and the state $\boldsymbol{x}^s$

or between the state $\boldsymbol{x}^s$ and the output $\boldsymbol{y}^m$.

Control theory distinguishes several normal forms, usually the form of the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ makes the difference. As this chapter is no introduction to control theory, only the *observable canonical form* is discussed. The reason to use the observable canonical form is of practical nature and will be explained later.

By a transformation process, defined e.g. in chapter 1.6 of [Unb97b], the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ take the form

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & 0 & -a_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & & 1 & 0 & -a_{n-2} \\ 0 & 0 & \cdots & & 0 & 1 & -a_{n-1} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} \tag{3.22}$$

$$\boldsymbol{C} = [0\,0\,0\cdots 1] \qquad \boldsymbol{D} = b_n \ .$$

It is important to note that the introduction of normal forms is neither a restriction of the dynamic system, nor is the input/output-behavior of the system changed. Only the internal state of the system is concerned. Using the observable canonical form the number of free parameters is reduced to $2n + 1$. This number is equal to the number of parameters in the differential equation (3.4), provided that $n = m$ and a_n is normalized to 1. So no redundancy is left.

Up to now only the description of systems in continuous time space is discussed. The next section 3.1.2 describes the transformation to discrete time domain. This transformation is also suitable to deduce the weights of a DBN, modeling the system.

3.1.2 Kalman filter

For digital control the output signals $q(t)$ are not measured for all points in time, usually the measurements are taken regularly with a fixed sampling period ΔT. The same is valid for the input signal $\boldsymbol{u}(t)$, which is only calculated for special points in time. Between two point t and $t + 1$ it is assumed that $\boldsymbol{u}_t$ is constant, i.e. $\boldsymbol{u}(\tau) = \boldsymbol{u}_t$ for $t \leq \tau < t + 1$. Linear time discrete systems are described in the same way as continuous time systems, only the coefficients have to be changed. The coefficients are deduced by comparison to Kalman filters, a well-known method for tracking and prediction of stationary, linear systems. It can be used to calculate the parameters of time discrete systems as described in section 3.1, equations (3.8) and (3.9).

Furthermore Kalman filters are a special case of DBNs, so the results obtained for Kalman filter, e.g. in [Gel94], may be used without any changes.

Kalman filters represent a time discrete system whose state transition

$$\boldsymbol{x}_{t+1}^{s} = \boldsymbol{A}_{\mathrm{BN}}\boldsymbol{x}_{t}^{s} + \boldsymbol{B}_{\mathrm{BN}}\boldsymbol{u}_{t} \tag{3.23}$$

is described by the matrix $\boldsymbol{A}_{\mathrm{BN}}$. In this equation t is used as index to denote a time discrete system. The index 'BN' is used, as the transfer matrices are different for time discrete systems.

To calculate $\boldsymbol{A}_{\mathrm{BN}}$, a solution of differential equation (3.8) is needed, as $\boldsymbol{A}_{\mathrm{BN}}$ has to integrate all the state transitions between t and $t+1$. As long as no input is present, equation (3.8) is solved by

$$\boldsymbol{x}^{s}(t) = \boldsymbol{x}^{s}(t_0)\boldsymbol{\Phi}(t, t_0) \tag{3.24}$$

where

$$\boldsymbol{\Phi}(t, t_0) = \sum_{i=0}^{\infty} \boldsymbol{A}^{i}\frac{(t-t_0)^{i}}{i!} \tag{3.25}$$

is called *transfer matrix*. Assuming $t = t_{k+1}$ and $t_0 = t_k$ with a constant time difference $\Delta T = t_{k+1} - t_k$ the matrix

$$\boldsymbol{A}_{\mathrm{BN}} = \boldsymbol{\Phi}(t_{k+1}, t_k) \tag{3.26}$$

depends only on the time difference ΔT and $\boldsymbol{A}$. Thus $\boldsymbol{A}_{\mathrm{BN}}$ is constant for all k; i.e., the parameters of a DBN, modeling the dynamic system, are independent from the time slice.

Taking into account the influence of the input on the state leads to

$$\boldsymbol{B}_{\mathrm{BN}}\boldsymbol{u}_{t} = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}(t_{k+1}, \tau)\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{3.27}$$

which simplifies to

$$\boldsymbol{B}_{\mathrm{BN}} = \Delta T \sum_{i=0}^{\infty} \frac{\boldsymbol{A}^{i}\Delta T^{i}}{(i+1)!}\boldsymbol{B} \tag{3.28}$$

if only systems with a constant sampling period $\Delta T = t_{k+1} - t_k$ and a constant input $\boldsymbol{u}_t = \boldsymbol{u}(\tau)$ within one time-slice are considered. To build a DBN which incorporates equations (3.26) and (3.28), $\boldsymbol{B}_{\mathrm{BN}}$ is used as weight matrix between the input nodes and the state nodes as shown in figure 3.4. Matrix $\boldsymbol{\Phi}(\Delta T)$ describes the transition from one state to the next and is therefore used as weight matrix for the inter slice connection between two states in neighboring time slices. This

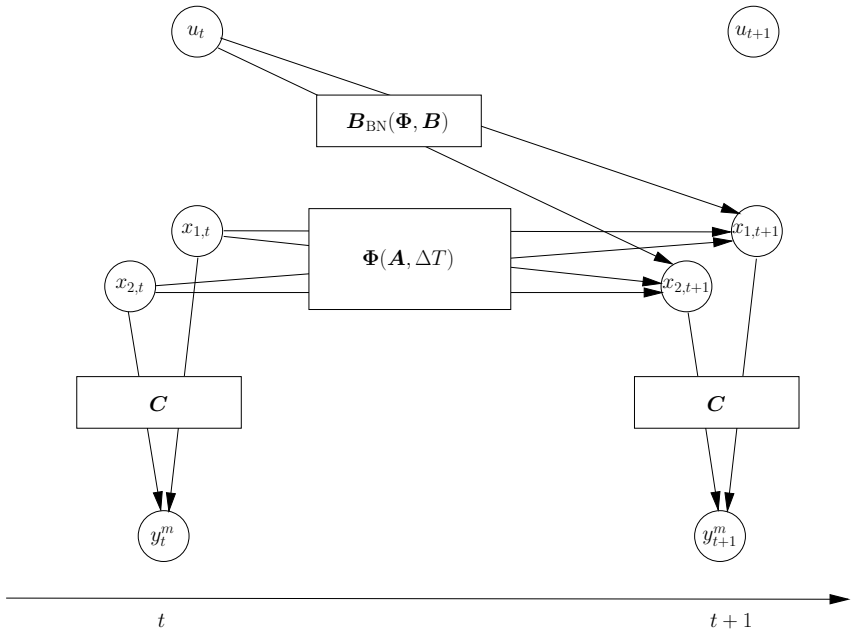


Figure 3.4: Bayesian network used for modeling the state transition

means that the state at time $t+1$ is calculated by

$$\boldsymbol{x}_{t+1}^s = [\boldsymbol{\Phi}(\boldsymbol{\Delta T})\ \boldsymbol{B}_{\mathrm{BN}}] \cdot \begin{bmatrix} \boldsymbol{x}_t^s \\ \boldsymbol{u}_t \end{bmatrix}. \tag{3.29}$$

In a BN the mean $\boldsymbol{\xi}$ of the normal distribution is equal to $\boldsymbol{\xi} = \boldsymbol{\alpha} + \boldsymbol{\beta z}$. In contrast to equation (2.37), describing the one-dimensional case, $\boldsymbol{\alpha}$ is a vector, and $\boldsymbol{\beta}$ is a matrix.

The offset $\boldsymbol{\alpha}$ has to be set to zero and the weights of the state node $\boldsymbol{X}^s$ are set to

$$\boldsymbol{\beta} = [\boldsymbol{\Phi}(\boldsymbol{\Delta T})\ \boldsymbol{B}_{\mathrm{BN}}] \quad , \tag{3.30}$$

so that the weights of the state nodes reflect the state transition in the presence of input $\boldsymbol{u}$. The output depends linearly on the state and is not time dependent. Thus the matrix $\boldsymbol{C}$ and $\boldsymbol{D}$ remain unchanged also in a time discrete system. Figure 3.4 shows two time-slices of a linear DBN. The nodes of the net are pictured as circles, the rectangles are used to display the weights of the Bayesian net. It is assumed that the modeled system is no jump Markov system, i.e. $\boldsymbol{D}$ = 0. As a further consequence, the dimension of the hidden state nodes is equal to the order of the differential equation describing the system.

3.1.3 Description of dynamic systems by difference equation

In the state-space description (3.8) and (3.9) the state $\boldsymbol{x}^s$ usually cannot be observed. Even if the parameters of a Bayesian network can be trained despite the occurrence of unobserved variables, it is more cumbersome. According to our experience, the number of needed iterations and examples is higher in the presence of unobserved variables. Thus it would be desirable to model with a description without hidden variables.

The starting point is again the description by differential equation (3.4). The derivatives of a function f can be approximated by the finite differences

$$\left.\frac{df}{dt}\right|_{t=k\Delta T} \approx \frac{f(k\Delta T) - f([k-1]\Delta T)}{\Delta T} \tag{3.31}$$

$$\left.\frac{d^2f}{dt^2}\right|_{t=k\Delta T} \approx \frac{f(k\Delta T) - 2f([k-1]\Delta T) + f([k-2]\Delta T)}{\Delta T^2} \tag{3.32}$$

$$\left.\frac{d^3f}{dt^3}\right|_{t=k\Delta T} \approx \frac{f(k\Delta T) - 3f([k-1]\Delta T) + 3f([k-2]\Delta T) - f([k-3]\Delta T)}{\Delta T^3} \ . \tag{3.33}$$

Derivatives of higher order are approximated in a similar manner. Thus it is possible to rewrite equation (3.4) using expressions (3.31) to (3.33) instead of the derivatives. This procedure results in a difference equation which can be solved for

$$y^m = -\sum_{i=1}^{n} a_i y^m_{t-i} + \sum_{i=0}^{n} b_i u_{t-i-d} \ , \tag{3.34}$$

with $d = \frac{T_d}{\Delta T}$ as the discrete dead-time. Please note that the coefficients in equations (3.4) and (3.34) are different, the coefficients for the time continuous system are marked by the superscript c. It is not possible to rewrite the differential equation to a difference equation without adapting the coefficients[1]. As only the structure will be used, a description of the transformation algorithm is omitted. The available algorithms are discussed e.g. in [Sch02].

In analogy to the description of dynamic systems by the Laplace transformation, time discrete systems are often described by the *Z-transformation* of a function $f(k)$

$$\mathcal{Z}\{f(k)\} = F(z) = \sum_{k=0}^{\infty} f(k) z^{-k} \ . \tag{3.35}$$

[1] For a transformation the Z-transformation might be used.

For single input single output systems the Z-transformation leads to a transfer function

$$G(z) = \frac{Y^m(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots b_n z^{-n}}{1 + a_1 z^{-1} + \cdots a_n z^{-n}} z^{-d} \quad . \tag{3.36}$$

Once again the system is described by two polynomials $A(z) = 1 + a_1 z^{-1} + \cdots a_n z^{-n}$, and $B(z) = b_0 + b_1 z^{-1} + \cdots b_n z^{-n}$, and the dead time d.

The description by Z-transformation has similar advantages as the Laplace transformation; e.g., the convolution integral, which is used to calculate the output of the serial connection of two systems, is mapped to the multiplication of the two Z-transfer functions. Similarly, the addition of two z-transfer functions is used to calculate the output of two parallel systems.

The difference equation (3.34) can also be used for estimation of the parameters a_i and b_j. Assuming that there are $n+N-1$ input- output pairs $u_{t-n+1}, u_{t-n+2} \cdots u_{t+N-1}, q_{t-n+1}, \cdots, q_{t+N-1}$, the error ϵ_{t+j} between the observed value q_{t+j} and the predicted values is, according to equation (3.34), equal to

$$\epsilon_{t+j} = q_{t+j} - q_{t+j-1}a_1 - q_{t+j-2}a_2 \cdots - q_{t+j-n}a_n + u_{t+j-1}a_1 + u_{t+j-2}a_2 \cdots + u_{t+j-n}a_n \quad . \tag{3.37}$$

For all $\epsilon_{t+1} \cdots \epsilon_{t+N}$ this results in a system of N linear equations

$$\begin{bmatrix} \epsilon_{t+1} \\ \epsilon_{t+2} \\ \vdots \\ \vdots \\ \epsilon_{t+N} \end{bmatrix} = \begin{bmatrix} q_{t+1} \\ q_{t+2} \\ \vdots \\ \vdots \\ q_{t+N} \end{bmatrix} - \begin{bmatrix} q_t & \cdots & q_{t-n+1} & u_t & \cdots & u_{t-n+1} \\ q_{t+1} & \cdots & q_{t-n+2} & u_{t+1} & \cdots & u_{t-n+2} \\ \vdots & & & \vdots & & \\ \vdots & & & \vdots & & \\ q_{t+N-1} & \cdots & q_{t-n+N} & u_{t+N-1} & & u_{t-n+N} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{3.38}$$

or shorter $\boldsymbol{\epsilon}(N) = \boldsymbol{q}(N) - \boldsymbol{M}(N)\boldsymbol{\Xi}(N)$. The matrix $\boldsymbol{M}$ denotes the matrix for the observations of the input- and output values. It is necessary to solve this equation for the parameter vector $\boldsymbol{\Xi}$. This can be done by the least square approach, the sum of the squared error is minimized by

$$\hat{\boldsymbol{\Xi}} = \left[\boldsymbol{M}(N)^{\mathrm{T}}\boldsymbol{M}(N)\right]^{-1} \boldsymbol{M}(N)\boldsymbol{q}(\boldsymbol{N}) \quad . \tag{3.39}$$

The approach just discussed is only valid for white noise ϵ. In the case of colored noise, there is no closed solution; a possible approach is e.g. gradient descent or the EM-algorithm. This case is discussed more deeply in [Unb97a].

The estimation of the control parameters may lead to a self-tuning controller whose basic principle is the identification of the system in the first step and the calculation of the controller's parameters in dependence of the plant model in a second step.

3.2 Non-Linearities in dynamic systems

For linear systems, discussed in the last section, equation (3.3) hold. All other systems are called non-linear. In this section an overview about frequently occurring non-linear units in control theory is given. Later on this list will be used to develop prototypical models for as many non-linear units as possible. The prototypical models can be combined among each other or with a linear sub-system. According to [SL91] the decomposition of a system in a linear and a non-linear subsystem is often possible and useful.

When analyzing non-linear systems the Laplace transformed is no longer the quotient of two polynomials. But it is still possible to characterize them by two differential equations

$$\dot{\boldsymbol{x}^s} = \boldsymbol{f}(\boldsymbol{x}^s, \boldsymbol{u}, t) \tag{3.40}$$

$$\boldsymbol{y}^m = \boldsymbol{g}(\boldsymbol{x}^s, \boldsymbol{u}, t) \tag{3.41}$$

with arbitrary non-linear functions $\boldsymbol{f}$ and $\boldsymbol{g}$, and the initial state

$$\boldsymbol{x}^s(t_0) = \boldsymbol{x}_0^s \ . \tag{3.42}$$

The state-space description of linear systems is a special case of differential equations (3.40) and (3.41). For modeling purposes and to analyze the behavior of dynamic systems, differential equation (3.40) may be linearized by a Taylor series. The operating point $\check{\boldsymbol{x}}^s, \check{\boldsymbol{u}}$ is usually an equilibrium point; i.e., the derivative of the state is zero. Thus $\boldsymbol{f}(\check{\boldsymbol{x}}^s, \check{\boldsymbol{u}}, t) = 0$. Using the substitutions

$$\Delta\boldsymbol{x}^s(t) = \boldsymbol{x}^s(t) - \check{\boldsymbol{x}}^s \tag{3.43}$$

$$\Delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \check{\boldsymbol{u}} \tag{3.44}$$

to denote the deviation from the operating point, differential equation (3.40) is approximated by

$$\Delta\dot{\boldsymbol{x}}^s \approx \boldsymbol{A}\Delta\boldsymbol{x}^s + \boldsymbol{B}\Delta\boldsymbol{u} \tag{3.45}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ denote the Jacobian matrices

$$\boldsymbol{A} = \left[\begin{array}{ccc} \frac{\partial f_1(\boldsymbol{x}^s,\boldsymbol{u})}{\partial x_1^s} & \cdots & \frac{\partial f_1(\boldsymbol{x}^s,\boldsymbol{u})}{\partial x_n^s} \\ \vdots & & \vdots \\ \frac{\partial f_n(\boldsymbol{x}^s,\boldsymbol{u})}{\partial x_1^s} & \cdots & \frac{\partial f_n(\boldsymbol{x}^s,\boldsymbol{u})}{\partial x_n^s} \end{array} \right] \Bigg| \begin{array}{rcl} \boldsymbol{x}^s & = & \check{\boldsymbol{x}}^s \\ \boldsymbol{u} & = & \check{\boldsymbol{u}} \end{array} \tag{3.46}$$

$$\boldsymbol{B} = \left[\begin{array}{ccc} \frac{\partial f_1(\boldsymbol{x}^s,\boldsymbol{u})}{\partial u_1} & \cdots & \frac{\partial f_1(\boldsymbol{x}^s,\boldsymbol{u})}{\partial u_r} \\ \vdots & & \vdots \\ \frac{\partial f_n(\boldsymbol{x}^s,\boldsymbol{u})}{\partial u_1} & \cdots & \frac{\partial f_n(\boldsymbol{x}^s,\boldsymbol{u})}{\partial u_r} \end{array} \right] \Bigg| \begin{array}{rcl} \boldsymbol{x}^s & = & \check{\boldsymbol{x}}^s \\ \boldsymbol{u} & = & \check{\boldsymbol{u}} \end{array} . \tag{3.47}$$

For a discussion see [Unb97a]. In this case the system can be modeled approximately as described in section 3.1. If linearization is no solution to the modeling problem, special solutions for each data set or rather engineering process have to be developed. In many cases common transfer units are met [Föl93; SL91; Unb97b]. In this section some of them are introduced. In section 5.1 Bayesian networks are introduced to model those non-linear units.

A common phenomenon in engineering is *saturation*. When a valve is closed, further diminishing the pressure has no effect. The same happens in an amplifier. For very high inputs the maximal output is reached. Further increasing the input does not change the output. Assuming a linear gain between minimal and maximal output, this type of non-linear unit is described by

$$y^m(u) = \begin{cases} -y^m_{\max} & \text{if} \quad u \leq -u_{\text{sat}} \\ \frac{y^m_{\max}}{u_{\text{sat}}} u & \text{if} \quad -u_{\text{sat}} \leq u \leq u_{\text{sat}} \\ y^m_{\max} & \text{if} \quad u \geq u_{\text{sat}} \end{cases} \tag{3.48}$$

This function consists of three different linear parts, so a first approach for modeling might be a discrete node with three states as a switch. This switch is parent of a continuous node which models the output.

Another non-linear unit, which might be applied as a simple temperature controller is the two-point element

$$y^m(u) = \begin{cases} -y^m_{\max} & \text{if} \quad u < 0 \\ y^m_{\max} & \text{if} \quad u > 0 \end{cases} . \tag{3.49}$$

Another possibility to express this relationship between in- and output is

$$y^m(u) = -y^m_{\max} \operatorname{sgn}(u) \tag{3.50}$$

When using this unit as a controller the input of the system is $u(t) = y^m_{\max}\ \text{sgn}(w(t) - q(t))$; i.e., the input is positive if the desired value is greater than the current value and vice versa. When systems with dead-time T_d has to be controlled an oscillation with the angular frequency

$$\omega_0 = \frac{\pi}{2T_d} \tag{3.51}$$

[Unb97b; Föl93] is observed. Thus, for systems with no or low dead time, a controller which reacts only when the deviation from the desired value is larger than a threshold u_Θ is from advantage. Thus a *three-point controller* with three possible outputs $-y^m_{\max}, 0, y^m_{\max}$

$$y^m(u) = \begin{cases} -y^m_{\max} & \text{if} \quad u < u_\Theta \\ 0 & \text{if} \quad -u_\Theta \leq u \leq u_\Theta \\ y^m_{\max} & \text{if} \quad u > u_\Theta \end{cases} \tag{3.52}$$

might be used for control purposes. It is possible to represent it by two two-point elements:

$$y^m_1(u) = \frac{y^m_{\max}}{2} + \frac{y^m_{\max}}{2}\text{sgn}(u - u_\Theta) \tag{3.53}$$

$$y^m_2(u) = -\frac{y^m_{\max}}{2} + \frac{y^m_{\max}}{2}\text{sgn}(u + u_\Theta) \tag{3.54}$$

$$y^m(u) = y^m_1(u) + y^m_2(u) = \frac{y^m_{\max}}{2}(\text{sgn}(u - u_\Theta) + \text{sgn}(u + u_\Theta)) \ . \tag{3.55}$$

A similar behavior is shown by the next function which shows a *dead zone* between the thresholds $-u_\Theta$ and u_Θ. For inputs smaller or greater than the thresholds the output

$$y^m(u) = \begin{cases} m_s(u + u_\Theta) & \text{if} \quad u \leq -u_\Theta \\ 0 & \text{if} \quad -u_\Theta < u < u_\Theta \\ m_s(u - u_\Theta) & \text{if} \quad u \geq u_\Theta \end{cases} \tag{3.56}$$

is also proportional to the slope m_s (confer figure 3.5).

Up to now all the introduced non-linearities consist of multiple lines. Of course this is not valid for all non-linearities. Simple counter-examples are

$$y^m(u) = u^2 \tag{3.57}$$

or arbitrary non linear functions, e.g. the dependency between volume and pressure in hydro-forming.

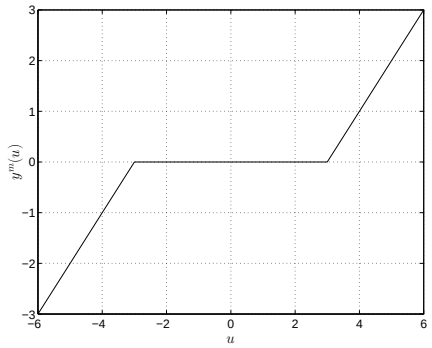


Figure 3.5: Dead zone

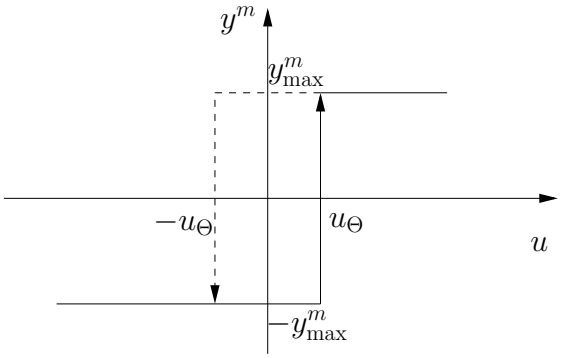


Figure 3.6: Hysteresis

Another class of non-linearities depends not only on the input, but also on the first derivative, e.g. the input/output dependency depicted in figure 3.6. There is no unique output for $u_\Theta \leq u(t) \leq u_\Theta$. The *hysteresis* was first observed in magnets, the function is equal to

$$y^m(u,\dot{u}) = \begin{cases} -y^m_{\max} & \text{if} \quad u < -u_\Theta \\ y^m_{\max}\text{sgn}(u - y^m_{\max}\text{sgn}(\dot{u})) & \text{if} \quad -u_\Theta \leq u \leq u_\Theta \\ y^m_{\max} & \text{if} \quad u > u_\Theta \end{cases} . \tag{3.58}$$

If more than one input is used also the multiplication or division of two signals is non-linear. Of course this list is not complete. But it allows to draw some conclusions. To model non-linearities with multiple lines or the dependency of the derivative of the input signal it is important to model the sgn-function.

For the general case $y^m = f(u)$ a Taylor series with multiple operating points is applied. The same approach might also be suitable for

$$y^m = u_1(t)u_2(t) \tag{3.59}$$

with operating points $\breve{u}_1$ and $\breve{u}_2$

$$y^m \approx \breve{u}_1\breve{u}_2 + \left.\frac{\partial y^m}{\partial u_1}\right|_{\substack{u_1=\breve{u}_1 \\ u_2=\breve{u}_2}} + \left.\frac{\partial y^m}{\partial u_2}\right|_{\substack{u_1=\breve{u}_1 \\ u_2=\breve{u}_2}} . \tag{3.60}$$

A direct representation with hybrid Bayesian networks is not possible, as the mean output of a continuous node depends linearly on the instantiation of the input nodes.

If the hybrid Bayesian network is inaccurate for the data to be modeled sampling mechanisms

like particle filters might be used. E.g. BUGS, a system based on Gibbs-sampling [STBG96a; STBG96b; STBG96c], allows the usage of deterministic nodes and non-Gaussian distributions. But the disadvantage is that there are no inference mechanisms with a closed solution for the general case.

Another problem in dealing with non-linearities is the detection of equilibrium points with $\dot{\boldsymbol{x}^s} = 0$. In linear systems

$$\dot{\boldsymbol{x}^s} = \boldsymbol{A}\boldsymbol{x}^s + \boldsymbol{B}_{\mathrm{BN}}\boldsymbol{u} = 0 \Rightarrow \boldsymbol{x}^s = \boldsymbol{A}^{-1}\boldsymbol{B}_{\mathrm{BN}}\boldsymbol{u} \tag{3.61}$$

there is one equilibrium point for systems with $\det(\boldsymbol{A}) \neq 0$. For systems with $\det(\boldsymbol{A}) = 0$ there might be an infinite number of equilibrium points or none. The latter case is impossible for $\boldsymbol{u} = 0$. For the non-linear case multiple equilibrium points can occur.

In the next section two different approaches for control, which calculate the input signals using the deviation of the current output from the desired value, are introduced. First PID controllers, which are widely used in industry, are discussed. The second approach are Dead-Beat controllers which guarantee a minimal settling time. For additional approaches see [Unb97a; Sch02].

3.3 Controlled systems

The aim of a controller is to change the input u of a dynamic system so that the output of a system is kept close or equal to the desired value. A simple example is the room-temperature, controlled by the heating. Here the desired value w is the favorite temperature of the inhabitants. This temperature should be kept constant despite disturbances, e.g. opened doors or windows. Usually the controller is driven by the error, the difference between the desired value and the output.

In section 1.2 approaches, usually referred to as intelligent control, are discussed. This section deals with traditional approaches, i.e. PID- and Dead-Beat controllers.

Additionally, several criteria are developed to judge the performance of the controller discussed in this section and the Bayesian controller in chapter 4.

The main elements of a control loop are depicted in figure 3.7 where all variables are assumed to be one-dimensional. The desired value w is compared with the output q of the system; e.g. the current temperature is compared with the desired one.

The difference between desired and current value e is used as input for the controller which

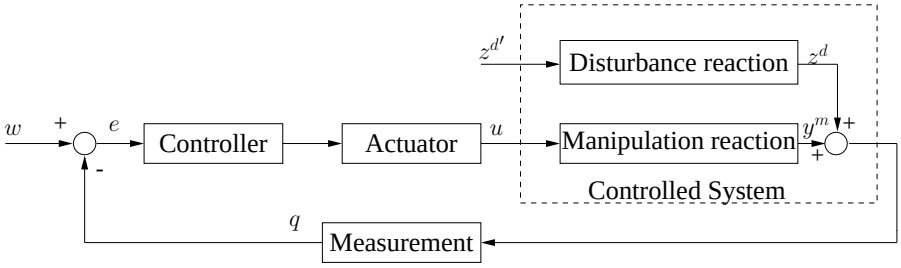


Figure 3.7: Block diagram of a controlled system

calculates the input for the actuator. The changed input value of the dynamic system results in a new output. The *manipulation reaction* describes how the system responds to new inputs. The manipulation reaction can e.g. be described by a difference equation or the state-space description as discussed in section 3.1. A crucial feature of control is the feedback loop which compares the output with the desired value.

Additionally, the controller has to deal with multiple disturbances $z_i^{d'}(t)$ from the environment. In a detailed approach the effect of each disturbance has to be modeled by its own transfer function, but for linear systems it is possible to substitute the effect of all disturbances by one disturbance variable

$$Z(s) = \sum_i G_i(s) Z_i(s) \tag{3.62}$$

which is added to the output. This model is similar to the approach used for system identification. From now on z^d is added to the output y^m so that $q(t) = y^m(t) + z^d(t)$.

In many cases controllers use the error $e(t) = w(t) - q(t)$ as input, a model of the dynamic system is not needed. To judge the performance of a controller there are several measures. The selection of a suitable measure depends on the application. The integral over the squared error

$$I = \int_0^\infty (e(t) - e_\infty)^2 dt \tag{3.63}$$

is frequently used, where e_∞ denotes the steady state error; i.e., the error which remains after convergence to a steady state. In digital control the error is only known for special points in time so that the squared error sum

$$I_\mathrm{d} = \sum_{t=0}^{t_{conv}} \Delta T (e_t - e_\infty)^2 \tag{3.64}$$

is used instead of the integral. Summation is stopped at t_{conv} when all signals are converged to

their final value. For practical reasons, convergence is assumed when all changes in the last 25 time slices are smaller then 10^{-4}.

The next section explains the function of a PID controller which is very popular in industry. As second controller type also the Dead-Beat controller is defined. The Dead-Beat controller is a digital filter which guarantees minimal settling time. As drawback a mathematical model of the dynamic system is required for parameter setting of the filter.

3.3.1 Different types of controllers

The task of a controller is to minimize the effects of an occurring disturbance. Widely used are PID controllers which have an output proportional to the error and to the integral/derivative of the error. There are different approaches for the selection of the controller-parameters depending on the user's requirements. When the mathematical description of the plant is given, a calculation which minimizes the squared error, as defined by equation (3.63) (for the calculation see [Unb97a]), or the settling time might be used. For a discussion of techniques from artificial intelligence see section 1.2.

This section starts with the discussion of the *PID controller*. Afterwards one method for the selection of the parameters which minimizes the settling time is given. It presupposes a mathematical description of the plant. If the plant parameters are unknown parameter estimation might be used, e.g. the EM algorithm. These considerations lead to the first type of self adaptive controllers, known as *self tuning controllers*(STC), which estimate the plant's parameters and calculate the controller's parameters accordingly. This approach is opposed to the *model-reference adaptive controller* (MRAC) which uses a reference model to specify the ideal response of the adaptive control system. The adaptation of the controller's parameters has the aim to approximate the reference model.

PID controller

A PID controller consists of three different parts, some of them may be missing. The (P)roportional part of a PID controller leads to an output

$$u(t) = K_{\mathrm{c}} e(t) \tag{3.65}$$

proportional to the error and to the gain of the controller K_{c}. Exceptionally, the output is denoted by $u(t)$ as the controller's output is usually used as the input of the dynamic system. The

disadvantage of a pure P controller is the remaining steady state error which is proportional to

$$e_\infty = \frac{1}{1+K_0} w\sigma(t) \tag{3.66}$$

when the input is proportional to the step function $\sigma(t)$. The closed loop gain $K_0 = K_\mathrm{c}K_\mathrm{s}$ is the product of the controller's gain K_c and the plant's gain K_s.

To avoid the steady state error, an (I)ntegral part, whose output

$$u(t) = \frac{K_\mathrm{c}}{T_I}\int_0^t e(\tau)d\tau \tag{3.67}$$

$$U(s) = \frac{K_\mathrm{c}}{T_I s}E(s) \tag{3.68}$$

is proportional to the integral of the error, is added. The output $u(t)$ of the integral part depends on the integral time T_I. Both the controller gain K_c and the integral time T_I are used to adapt the PI-controller to different dynamic systems. The PI-controller reduces the steady state error for a disturbance $z^d(t) = z_0^d\sigma(t)$ to zero.

To reduce the overshoot, a control signal

$$u(t) = K_\mathrm{c}T_D\frac{de(t)}{dt} \tag{3.69}$$

$$U(s) = K_\mathrm{c}T_D sE(s) \tag{3.70}$$

proportional to the derivative of the error is used. The (D)erivative part is governed by the derivative time constant T_D. So a PID controller, which combines all three parts, has the output

$$u(t) = K_\mathrm{c}e(t) + \frac{K_\mathrm{c}}{T_I}\int_0^t e(\tau)d\tau + K_\mathrm{c}T_\mathrm{D}\frac{de(t)}{dt} \quad . \tag{3.71}$$

As it is impossible to realize a pure D controller, a DT_1 unit with

$$U(s) = K_\mathrm{c}T_\mathrm{D}\frac{T_1 s}{1+T_1 s}E(s) \tag{3.72}$$

is used. Its behavior depends on the one hand on the derivative time constant T_D and on the other hand on the time constant T_1.

In most of the cases the controller uses the error $e(t) = w(t) - q(t)$ as input signal which results in the control loop depicted in figure 3.8 with

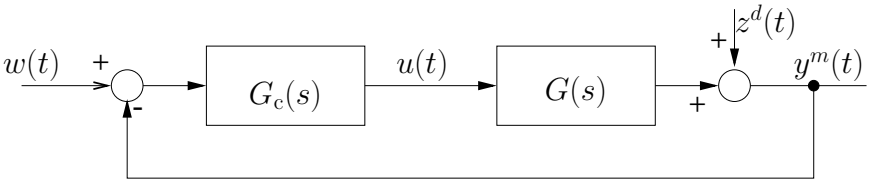


Figure 3.8: Typical control loop

$$Y^m(s) = Z^d(s) + [W(s) - Y^m(s)]G_c(s)G(s) \tag{3.73}$$

$$Y^m(s) = \frac{1}{1 + G_c(s)G(s)} Z^d(s) + \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} W(s) \ . \tag{3.74}$$

Neglecting the disturbance variable, equation (3.74) leads to

$$Y^m(s) = \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} W(s) \tag{3.75}$$

or to the control transfer function

$$G_w(s) = \frac{Y^m(s)}{W(s)} = \frac{G_c(s)G(s)}{1 + G_c(s)G(s)} \tag{3.76}$$

which reflects the reaction of the system, when the desired value is changed. This equation can be used to calculate the transfer function of the controller $G_c(s)$

$$G_c(s) = \frac{1}{G(s)} \frac{G_w(s)}{1 - G_w(s)} \ , \tag{3.77}$$

provided that the desired command response is known.

A similar equation in the discrete time domain can be used to figure out controller settings for a filter, which guarantees that the new desired value is reached within finite time. This filter, called Dead Beat controller, is discussed in the next section.

Dead Beat Controller

In digital control an equation similar to (3.77)

$$G_c(z) = \frac{1}{G(z)} \frac{G_w(z)}{1 - G_w(z)} \ , \tag{3.78}$$

is used. The only difference is that z-transformation is used instead of Laplace-transformation. To design a controller, the control transfer function $G_w(z)$ is assumed to be equal to the desired control transfer function $F_w(z)$. Thus the controller results in

$$G_c(z) = \frac{1}{G(z)} \frac{G_w(z)}{1 - G_w(z)} = \frac{A(z)}{B(z)z^{-d}} \frac{F_w(z)}{1 - F_w(z)} \tag{3.79}$$

where d is the discrete dead-time $d = \frac{T_d}{\Delta T}$. In order to get a realizable controller the dead-time of the closed loop is at least the dead-time of the system. Thus the required control transfer function is restricted to

$$F_w(z) = \frac{K(z)z^{-d}}{N(z)} \tag{3.80}$$

where $K(z)$ and $N(z)$ are arbitrary polynomials, provided that $A(z)$ and $B(z)$ have only zeros within the unit circle. That is, equation (3.80) is only applicable for stable systems.

One possible design criterion is that the output y_t^m is equal to the desired value w_t after a finite number of time steps. This requirement is expressed by the restriction of $F_w(z)$ to

$$F_w(z) = K(z)z^{-d} \ . \tag{3.81}$$

This selection guarantees only that the output is equal to the desired value at sampling time. Therefore stability of the input signals is used as next requirement. The selection

$$K(z) = \frac{B(z)}{B(1)} \tag{3.82}$$

guarantees a stable input (for an explanation see [Unb97a; Sch02]) and additionally a minimal settling time. Together with equation (3.79) this results in

$$G_c(z) = \frac{A(z)}{B(1) - B(z)z^{-d}} \ . \tag{3.83}$$

The approach of a Dead-Beat controller is not restricted to stable dynamic systems. The application to unstable systems leads to further restrictions, discussed e.g. in [Unb97a; Sch02].

Not in all cases a mathematical description of the plant is given. An empirical approach was introduced e.g. by Ziegler and Nichols [ZN42]. Even if strong restrictions apply, it is still very popular (confer page 246, [Unb97a]). The approach by Ziegler and Nichols is therefore also compared to the Bayesian controller.

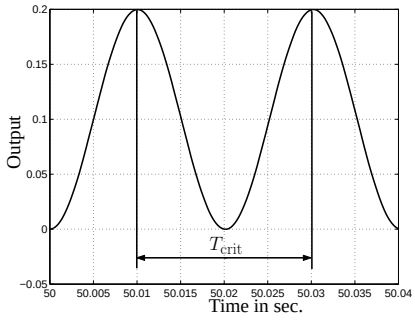


Figure 3.9: Output signal for $K_c = K_{crit}$

Type of controller	Settings using raise and delay time K_C	T_I	T_D	Controller settings using critical gain		
P	$\frac{1}{K_s}\frac{T_r}{T_u}$			$0.5K_{crit}$		
PI	$\frac{0.9}{K_s}\frac{T_r}{T_u}$	$3.33T_u$		$0.45K_{crit}$	$0.85T_{crit}$	
PID	$\frac{1.2}{K_s}\frac{T_r}{T_u}$	$2T_u$	$0.5T_u$	$0.6K_{crit}$	$0.85T_{crit}$	$0.12T_{crit}$

Table 3.1: Controller settings based on the methods of Ziegler and Nichols

Empirical approach

Another frequently used method is developed by Ziegler and Nichols. It is valid for systems approximated by first order systems with dead time T_d

$$G_s(s) = \frac{K_s}{1 + T_1 s} \exp(-T_d s) \ , \tag{3.84}$$

which are determined by a time time constant T_1, the gain K_s and the dead time T_d. The step response of such systems is characterized by the gain K_s, the raise time T_a and the delay time T_u. The parameter are figured out using the intersection of the tangent with the x-axis and the line with $y^m(t) = K_s$ as displayed in figure 3.3. In [ZN42] the settings given in table 3.1 are suggested.

A second method, also by Ziegler and Nichols, is based on the measurement of the critical controller gain K_{crit}.

Critical controller gain means that the controller gain K_c of a P controller is increased until oscillation is observed, as depicted in figure 3.9. The duration T_{crit} of the oscillation is measured. Afterwards the formulas given in table 3.1 are used for adjustment of the parameter.

Chapter 4

Modeling Linear Dynamic Systems with Bayesian Networks

As seen in chapter 3 automatic control is a field with a long tradition. Also self-adaptive control has a history of more than 50 years. Of course, self-adaptive control requires a kind of learning or training algorithm which either refines the adaptation of the controller directly, or first improves the model, and afterwards the controller's parameters depending on it. Frequently used approaches in adaptive control are neural networks and Fuzzy Logic. Bayesian networks are seldom used, although they offer a lot of attractive properties for the usage in adaptive control. The main advantage is the similarity between Kalman filter and DBNs. Thus, a-priori knowledge from control theory can easily be used to deduce the structure and parameters of a suitable model. Afterwards the Bayesian network is used without any changes to figure out suitable input for the system to be controlled.

For non-linear systems, there are no models readily available. In principle, there are also structure learning algorithms, but experience in modeling manufacturing processes shows that a lot of know-how about the modeled process is required during development of the model.

For the modeling of non-linear systems a strategy that consists of two different steps is suggested in this thesis. Firstly, well known non-linearities are modeled. Secondly, the primitive units developed in the first step can be combined - either by a well informed knowledge-engineer or by a modified structure learning algorithm - to a complex model. The suggestions for the first step are described in chapter 5.1, the second step is beyond the scope of this thesis.

In the next section the similarities between Kalman filter and DBNs are used to map the state-space model to a DBN. As DBNs are not only able to predict results from given inputs, but also to calculate suitable input signals to obtain a desired output, this model will be used to infer

suitable manipulated variables, given the desired value.

Additionally the test systems, used to evaluate the new type of controller, are introduced. In section 4.2 it is shown how normal forms (confer section 3.1.1) are applied to reduce the search space for training of Bayesian networks.

The trained models are tested with the same systems as in section 4.1, so that the performance of the trained model can be compared with the analytical one. The results obtained with this structure are sufficient in most cases. In one case however, convergence of the output signal is not achieved. To accomplish convergence in all cases a second structure, based on difference equations, is tested. When assuming that there is no disturbance during training, there are no hidden nodes left. Thus greater accuracy is obtained with less training data. The disadvantage of this model is that the Markov assumption is not met. So most of the tools for Bayesian models are not able to deal with such models directly. A work around is described in section 4.3, the expansion of the Bayesian toolbox together with the experiments for controlling higher order systems are described in section 4.4.

As the control with Bayesian networks is a completely new approach, a comparison with traditional approaches is also necessary. One method, frequently used in industrial control, are PID controllers. A traditional method for figuring out their setting is introduced by Ziegler and Nichols [ZN42]. A modern mean, used in Digital control, are Dead-Beat [Sch02] controllers. Dead Beat controllers are used to eliminate the deviation of the output from the desired value in a finite number of time steps.

The comparison with these approaches, which finishes the discussion of Bayesian control of linear systems, is presented in section 4.5. Linear systems are an important subset of dynamic systems. Sometimes the approaches introduced in this chapter are also suitable for non linear systems, provided that linearization around the operating point is possible. The other cases are discussed in the next chapter which deals with nonlinear systems.

4.1 Principle of a model based controller

The main idea of a Bayesian controller is that the dynamic system is modeled by a DBN [DDN02a; DDN03a]. The desired value is entered as evidence. A calculation of the marginal distribution of the input nodes results in the input which may be used to achieve the desired value. A linear

SISO system may be modeled by the state-space description

$$\boldsymbol{x}_{t+1}^s = \boldsymbol{A}_{\mathrm{BN}}\boldsymbol{x}_t^s + \boldsymbol{b}_{\mathrm{BN}}u_t \tag{4.1}$$

$$y_t^m = \boldsymbol{c}_{\mathrm{BN}}\boldsymbol{x}_t^s + d_{\mathrm{BN}}u_t \ , \tag{4.2}$$

where $\boldsymbol{x}_t^s$ represents the state of the system, u_t the input, and y_t^m the undisturbed output of the system. These equations were readily discussed in section 3.1, compare with equations (3.8), (3.9), and (3.23) for time discrete systems.

Provided that the state $\boldsymbol{x}_t^s$ and the input u_t are normally distributed, a Bayesian network, as displayed in figure 4.1(a), can be used to calculate the mean of the following state $\boldsymbol{x}_{t+1}^s$ and the output y_t^m. The distribution of a continuous node Y with parents $\boldsymbol{Z}$ is given by

$$p(y \mid \boldsymbol{z}) = \mathcal{N}(\alpha + \boldsymbol{\beta}\boldsymbol{z}, \gamma) \ , \tag{4.3}$$

where α denotes the mean, $\boldsymbol{\beta}$ the weight vector of the link $\boldsymbol{Z} \rightarrow Y$, and γ the variance of the distribution.

Setting the mean $\alpha = 0$ results in

$$p(\boldsymbol{x}_{t+1}^s \mid u_t, \boldsymbol{x}_t^s) = \mathcal{N}(\boldsymbol{\beta}_1 \begin{bmatrix} \boldsymbol{x}_t^s \\ u_t \end{bmatrix}, \boldsymbol{\Gamma}_1) \tag{4.4}$$

$$p(y_t^m \mid u_t, \boldsymbol{x}_t^s) = \mathcal{N}(\boldsymbol{\beta}_2 \begin{bmatrix} \boldsymbol{x}_t^s \\ u_t \end{bmatrix}, \gamma_2) \tag{4.5}$$

for the distributions of $\boldsymbol{X}_{t+1}^s$ and $\boldsymbol{Y}^m$ in figure 4.1(a). The weight $\boldsymbol{\beta}_1$ is $n \times (n+r)$-dimensional for a n-dimensional state and r-dimensional output. Weight $\boldsymbol{\beta}_2$ is $o \times (n+r)$-dimensional, where o denotes the output-dimension. When $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ are set to

$$\boldsymbol{\beta}_1 = [\boldsymbol{A}_{\mathrm{BN}} \ \boldsymbol{b}_{\mathrm{BN}}] \tag{4.6}$$

$$\boldsymbol{\beta}_2 = [\boldsymbol{c}_{\mathrm{BN}} \ d_{\mathrm{BN}}] \ , \tag{4.7}$$

the Bayesian network depicted in figure 4.1(a) models the state transition and output of a dynamic system as intended. Such a network can be used for simple control purposes, when no disturbances occur (confer [DDN00a]). For real applications disturbances have to be considered. In reality these disturbance variables may affect everywhere. For linear systems all disturbance variables, together with their transfer functions, can be added to a disturbance variable z_t^d which

acts directly on the output of the dynamic system. Thus it is necessary to distinguish the modeled output of the system y_t^m and the observed or measured output

$$q_t = y_t^m + z_t^d \tag{4.8}$$

which includes the disturbance variable z_t^d. Following this discussion figure 4.1(a) has to be extended to figure 4.1, which includes nodes for the representation of the disturbance variable. In the case discussed here, the modeled output y_t^m depends only on the input and the state, and not on the system's output (compare to the state-space description, discussed in section 3.1, equations (3.8) - (3.10)). Therefore

$$q_t = y_t^m + z_t^d = \boldsymbol{c}_{\mathrm{BN}} \boldsymbol{x}_t^s + d_{\mathrm{BN}} u_t + z_t^d \tag{4.9}$$

can be modeled directly, the weight vector $\boldsymbol{\beta}_{Q_t}$ of node Q_t is equal to

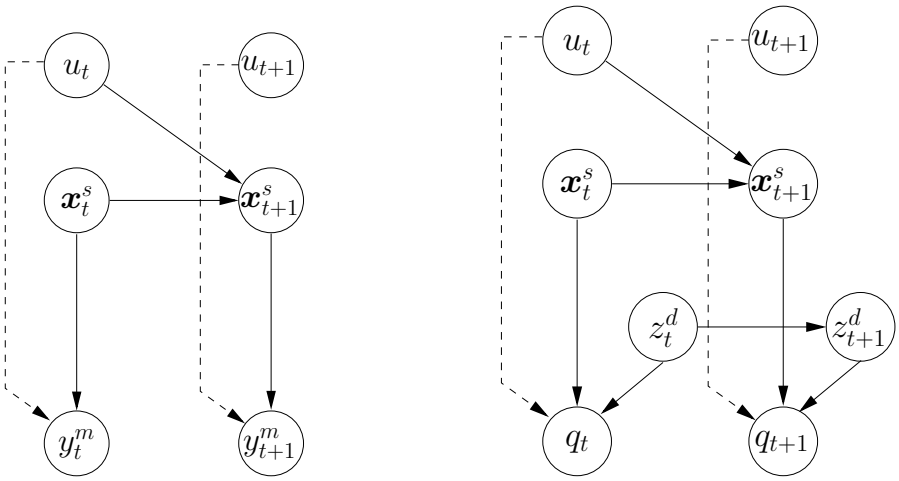
$$\boldsymbol{\beta}_{Q_t} = [\boldsymbol{c}_{\mathrm{BN}}\ d_{\mathrm{BN}}\ 1] \ . \tag{4.10}$$

The connection between the input node U_t and the output node Y_t^m or Q_t, depicted as dashed line, is only necessary for systems which react immediately on changing the input.

For systems to be controlled by a Bayesian network, the effect of a disturbance at time t has to be eliminated, by changing subsequent inputs $u_{t+j}, j \geq 1$. Thus the disturbance has to be estimated for the future. This is done by estimation of the former disturbance as difference between the modeled and the observed output. Afterwards this estimation is propagated into the future. This is done by adding a link between Z_t^d and Z_{t+1}^d as displayed in figure 4.1. When assuming that the disturbance usually stays constant from one time step to the next, this link gets a fixed weight of one.

If there are large changes of the disturbance from one time-slice to the next, the frequency of the disturbance might be larger or close to the sampling frequency. Thus the precondition of the sampling theorem does no longer hold. As a consequence the sampling rate, set manually before the training, should be increased.

Adding some more time-slices and the link between Z_t^d and Z_{t+1}^d leads to figure 4.2. The n_{past} time-slices at the left hand side are reserved for the representation of the past. In figure 4.2 three time-slices, denoted by $t-2 \cdots t$ are used for the past. Each time-slice consists of four different nodes. The layer at the top represents the input. The input u_t has an influence on the state $\boldsymbol{x}_{t+1}^s$. The state nodes are depicted as second layer in figure 4.2. The third layer is used for



(a) State-space description without disturbance variable

(b) Inclusion of disturbance variable

Figure 4.1: Model for state-space description

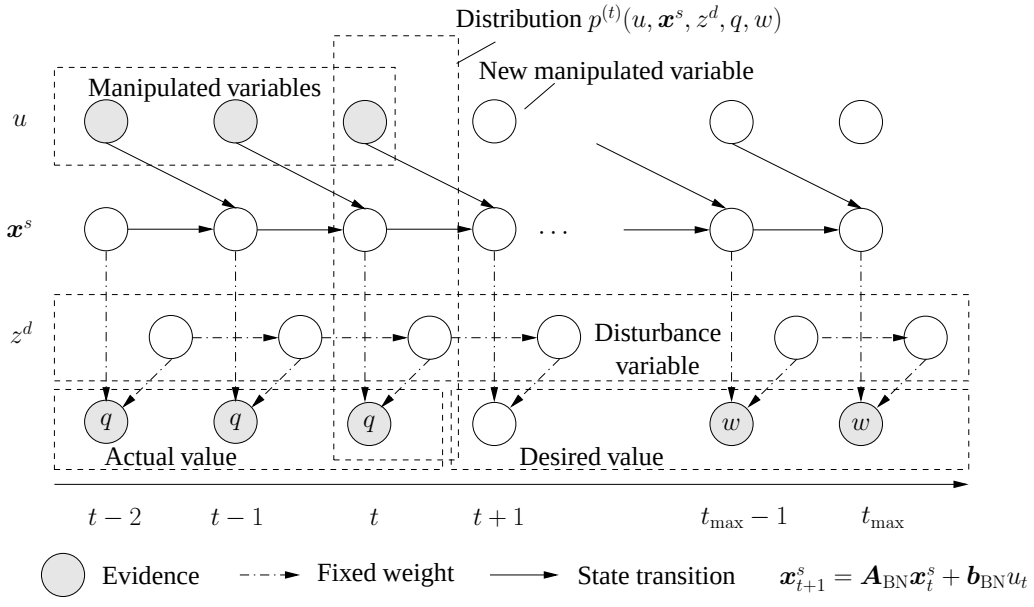


Figure 4.2: State-space model used for control

the estimation of the disturbance variable z_t^d. The observed output q_t, shown in the forth layer, is a linear function of the disturbance variable z_t^d and the state $\boldsymbol{x}_{t+1}^s$. The information available for the past is the input $u_{t-n_{\text{past}}+1} \cdots u_t$ and the observed output $q_{t-n_{\text{past}}+1} \cdots q_t$.

The output q_{t+1} is determined by z_{t+1}^d and $\boldsymbol{x}_{t+1}^s$, the input u_{t+1} has no influence on q_{t+1}. Thus no evidence is given for q_{t+1}.

For the future it is necessary to tell the controller the desired value w (compare with section 3.3). This is achieved by using w as additional evidence for q_{t+i}, that is $q_{t+2} = \cdots = q_{t_{max}} = w$. The main idea is to enter u, q and w as evidence of the shaded nodes in figure 4.2 and use the (marginal) distribution of u as input. To come to a working system, which shows a good performance, some missing parameters have to be set. First the parameters of the input nodes are yet undiscussed. The mean of the input depends on the gain of the system and on the desired value. Both of them are unknown. That means that nothing is known about the mean of the system. This is expressed by a large variance γ_U.

For the selected model all nodes in a layer have the same parameters. Exceptionally the parameters for the nodes in the first time-slice might be different from the parameters used for the remaining time-slices, as nodes in the first time-slice have no parents. Thus, in figure 4.2, the parameters for Z_{t-2}^d are different from the parameters for Z_t^d. However, usually it makes no sense to add information about the time-slice when parameters are discussed in a DBN.

A large variance γ_U for the input nodes means that the mean α_U has only a negligible influence on the marginal distribution of U. Beside the lack of knowledge about a suitable mean of the input, there is a second consideration which leads to the same result. If the desired value w is changed, the Bayesian network regards this change as observation in the future.This observation can be explained by three different causes:

- The input has changed which leads to the new desired value. This is the required reaction. Please imagine that your guest is shivering. As a reaction you will increase the desired temperature w. It is the task of the controller to increase the measured temperature q, e.g. by increasing the flow of hot water through the heating, i.e. by changing the input.

- The disturbance variable will change in the future. If both in- and output had been constant in the past, there is no reason that this will happen.

- The system changes the estimation of the state. As the system under consideration is regarded as time invariant this is the least likely cause of a changed input.

Only the first possibility results in the desired behavior. To ensure that a changed output (a new desired value) is explained by a changed input the dispersion of the input node has to be set to a

maximum. In our application the covariance matrix $\boldsymbol{\Gamma}_{\boldsymbol{X}^s} = \mathrm{diag}(\gamma_1, \cdots, \gamma_n)$ is diagonal. To get a maximal covariance for the input nodes means that $\gamma_U \gg \gamma_i \;\; 1 \leq i \leq n$ should hold.

This concludes the discussion about the parameters of U. The next node under consideration is the state node. Mean and weights of this node are calculated, the covariance is yet unknown. First it is supposed that the used model is correct, expressed by a minimal covariance. Also considerations about an occurring disturbance leads to the same result. When a disturbance occurs the observation q_{t+1} no longer fits to the state estimation. As q_{t+1} depends on $\boldsymbol{x}^s_{t+1}$ and z^d_{t+1} there are two possible explanations for a deviation between the estimated and the observed output:

- The estimation of $\boldsymbol{x}^s_{t+1}$ or q is wrong. This is synonymous with an erroneous model.
- The disturbance has changed.

As the second explanation is much more likely than the first one, the covariance of the disturbing value in the first time-slice is selected larger than the covariance of the state-nodes, i.e. $\boldsymbol{\Gamma}_{\boldsymbol{X}^s} = \mathrm{diag}(\gamma_1, \cdots, \gamma_n)\ \gamma_{Z^d_{t=1}} \gg \gamma_i \;\; 1 \leq i \leq n$. For the remaining time-slices the variances γ_1, γ_{Z^d}, and γ_Q are selected in the same order of magnitude.

The last parameter to be selected is the mean of the disturbance variable. The disturbance is regarded as white noise, the mean α_{Z^d} is set to zero.

4.1.1 Calculation of the input signal

Control is an ongoing task; i.e., in regular intervals new output signals are measured, and new input signals have to be calculated. When this task is finished all old signals are moved one time-slice to the left, the oldest-one is deleted and the current signals are entered at time-slice t. This is necessary as we work with a DBN of finite length.

This section discusses the steps of one complete circle which consists of calculation of new input, measurements, and shifting the evidences to the left.

At the beginning of each circle the signals $u_{t-n_{\mathrm{past}}+1} \cdots u_t, q_{t-n_{\mathrm{past}}+1}, \cdots q_t, q_{t+2} = w = q_{t+3} = \cdots q_{t_{max}}$ are entered as evidence. Afterwards the input signals $u_{t+1} \cdots u_{t+i}$ are calculated by marginalization. The new input signal u_{new}

$$u_{\mathrm{new}} = \frac{1}{\sum_{i=1}^{k} w_i} \sum_{i=1}^{k} w_i u_{t+i} \tag{4.11}$$

is calculated as a weighted sum of k signals. For our experiments $w_i = k + 1 - i$ is used so that

Table 4.1: Test systems for evaluation

System	K	T_1	T_2	Description
1	2	1	0.1	Damped system with gain two with no tendency to overshoot
2	2	0.1	0.1	System with damping $D < 1$. Thus the step response shows an overshoot
3	10	0.05	0.1	System with high gain and a large tendency to overshoot.

the input signals for the near future get a strong weight. If only the prediction u_{t+1} is used for the calculation of u_{new}, this might result in an oscillating signal.

After calculation, the input signal is sent to Simulink which is used for the simulation of the dynamic system. Simulink calculates the output of the system and the last in- and output is stored. Afterwards all signals are shifted one time-slice to the left, so that the total number of time-slices is kept constant.

4.1.2 Evaluation of the controller

For evaluation of the controller, the reference and the disturbance reaction is used. Reference reaction means that the system's response is evaluated, if the desired value w is changed. The disturbance reaction tests the ability of the controller to eliminate the effects of the disturbance.

For the evaluation three different systems of second order, described by the differential equation

$$Ku(t) = y^m(t) + T_1 \frac{dy^m}{dt} + T_2^2 \frac{d^2y^m}{dt^2} \quad , \tag{4.12}$$

are used. The test systems are listed in table 4.1.

As discussed in section 3.1, the behavior of a linear dynamic system is characterized by the damping $D = \frac{T_1}{2T_2}$. The step response of system number 1 has a damping $D > 1$. Thus, it shows no overshoot, as the step response of systems 2 and 3. In comparison to system number 1 and 2, test system 3 has a larger gain. Thus the three test systems show different characteristics. They can be therefore be regarded as being representative for the class of linear second order systems.

To use the Bayesian network as controller the weights has to be set so that the DBN models the behavior of the dynamic system to be controlled. Using the substitution $x_1^s = y^m$ and $x_2^s =$

$\frac{dy^m}{dt} = \dot{x_1^s}$ leads to the following equivalent system of first order differential equations

$$\begin{bmatrix} \frac{dx_1^s(t)}{dt} \\ \frac{dx_2^s(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{T_2^2} & -\frac{T_1}{T_2^2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K}{T_2^2} \end{bmatrix} u(t) \tag{4.13}$$

$$y^m(t) = [1\ 0] \begin{bmatrix} x_1^s \\ x_2^s \end{bmatrix} . \tag{4.14}$$

From this equation the state-space description

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} 0 & 1 \\ -\frac{1}{T_2^2} & -\frac{T_1}{T_2^2} \end{bmatrix} \ ; \ \boldsymbol{b} = \begin{bmatrix} 0 \\ \frac{K}{T_2^2} \end{bmatrix} \\ \boldsymbol{c} &= [1\ 0] \qquad\qquad d = 0 \end{aligned} \tag{4.15}$$

is deduced. Using equations (3.26) and (3.28) the state-space description is transformed to a discrete time state-space description. With the constants for system 1 and $\Delta T = 0.05\ s$ this leads to

$$\begin{aligned} \boldsymbol{A}_{\mathrm{BN}} &= \begin{bmatrix} 0.9605 & 0.0096 \\ -0.9631 & -0.0026 \end{bmatrix} \ ; \ \boldsymbol{b}_{\mathrm{BN}} = \begin{bmatrix} 0.0791 \\ 1.9262 \end{bmatrix} \\ \boldsymbol{c}_{\mathrm{BN}} &= [1\ 0] \qquad\qquad d_{\mathrm{BN}} = 0\ . \end{aligned} \tag{4.16}$$

For the judgment of the Bayesian controller, first the desired value is changed from 0 to 10 to test the reference reaction. The in- and output signals q respectively u are recorded to calculate the evaluation criteria given in table 4.2. For example the settling time t_s until the difference between desired value and output q remains under a threshold is measured. A second evaluation criterion is the squared error sum.

After convergence of in- and output signals the disturbance z^d is changed from 0 to 1, so that the disturbance reaction is tested.

The results, shown in table 4.3, are obtained with a dynamic Bayesian network with a sampling period of $\Delta T = 0.05\,s$. It should be stressed that there is nearly no steady state error. Of course also PI and PID controllers are able to reduce the steady state error to zero.

The time $t_s(z^d = 0, c\%)$ until the desired value is obtained is acceptable. It can be compared to a Dead-Beat controller, explained in section 3.3.1, whose transfer function is calculated using the plant's transfer function. This type of controller is able to achieve the desired value in a finite number of time steps which is equal to the order of the dynamic system and a term, depending

Table 4.2: Used evaluation criteria for controllers

Criterion	Description
Squared error sum $I_{\mathrm{d}}(z^d)$	Calculated according to equation (3.64). This sum is similar to the squared integral criterion, but is adapted to the time discrete case.
Overshoot	Difference between the maximal output and the desired value
e_∞	Steady state error, that is the remaining error after convergence of the output signal.
$t_s(z^d = 0, c\%)$	Time from changing the desired value w until the difference between the desired value and the current observed output remains smaller than $c\%$ of the desired value.
$t_s(z^d = 1, c\%)$	Same as $t_s(z^d = 0, c\%)$, but the disturbance is changed from 0 to 1.

on the dead time of the system. For our examples a Dead-Beat controller would react in 0.1 s as the test-systems are of second order and have no dead time.

The price for the short reaction time of the Dead-Beat controller are large input signals which might exceed the capability of the actuators. The reaction time for the Bayesian controller can be shortened when $u_{\mathrm{new}} = u_{t+1}$. Please confer with section 4.5 for a comparison between Bayesian controller and traditional controller.

As a result of this section it is concluded that Bayesian controllers work as intended, if they are supplied with a correct model. The next step towards a self adaptive controller is to use training algorithms to get the weights of the Bayesian network. This will be the subject of the next section.

4.2 Trained Bayesian controller

Before starting the discussion about the training of the Bayesian network it is necessary to clarify which parameters are trained. In the preceeding section it was explained that a Bayesian controller needs a special relationship between the node‘s covariances, in order to guarantee an optimal performance of the controller. Thus, the following parameters are excluded from the training:

- Covariance of the input nodes. The result is that the controller reacts to a changed disturbance or desired value by adapting the input.

- The covariance of the nodes representing the disturbance. For our test models this value is

Table 4.3: Results for Bayesian controller with calculated weights

System	1	2	3
$I_{\mathrm{d}}(z^d = 0)$	8.4	9.6	10.3
$e_\infty(z^d = 0$	0.02	0.01	0
Overshoot	0.02	0.56	0.92
$I_{\mathrm{d}}(z^d = 1)$	0.15	0.19	0.23
$e_\infty(z^d = 1)$	0	0	0.01
$t_s(z^d = 0, 1\%)$ [s]	0.45	0.45	0.7
$t_s(z^d = 0, 3\%)$ [s]	0.35	0.4	0.45
$t_s(z^d = 1, 1\%)$ [s]	0.45	0.55	0.6
$t_s(z^d = 1, 3\%)$ [s]	0.3	0.35	0.45

fixed to 5 for the first time-slice, for the remaining time-slices $\gamma_{Z^d} = 0.01$.

- The links between Z_t^d and Q_t get a fixed weight of one.

Excluding the parameters from training has the positive side effect that the search space is restricted. This is relevant as the EM-algorithm, used for training, converges towards a local extremum.

Beside the reduction of the search space, obtained by fixing the parameter listed above, the search space is further restricted by the usage of normal forms, discussed in section 3.1.1. The consideration starts with the observation that the differential equation

$$\sum_{i=0}^{n} a_i^c \frac{d^i y^m(t)}{dt^i} = \sum_{j=0}^{m} b_j^c \frac{d^j u(t)}{dt^j} \tag{4.17}$$

depends on less parameters than the state-space description (confer equations (3.8) and (3.9), discussed in section 3.1). Thus there is no unique mapping from a differential equation to the state-space description. The usage of the observable canonical form removes this ambiguity. When using the observable canonical form $\boldsymbol{c}_{\mathrm{BN}}$ is set to $[0 \cdots 0\ 1]$, thus only one link from the last state node to the output is needed. The net simplifies as depicted in figure 4.3 which uses a dynamic system of second order as example.

The dashed line from $x_{1,t}^s$ to $x_{1,t+1}^s$ at the right hand side should indicate that this connection is superfluous as long as only the normal form is regarded, The experiments in this section are made with a net where $x_{1,t}^s$ and $x_{1,t+1}^s$ are connected, as experiments have shown that the results

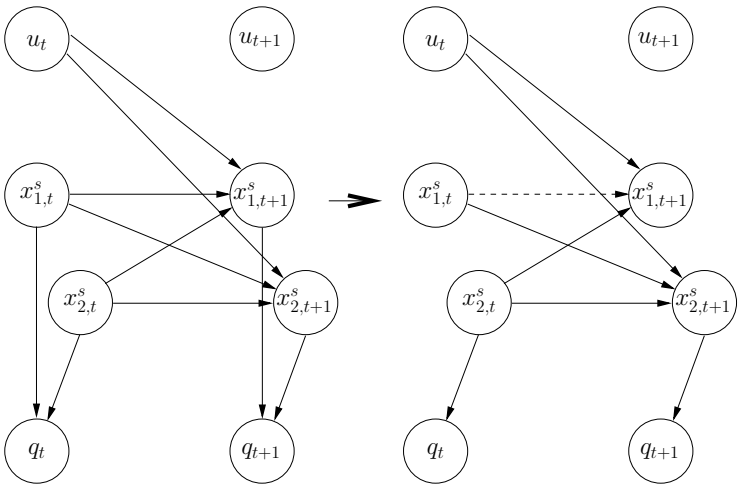


Figure 4.3: Simplification of a dynamic system of second order

are slightly better if a net including this edge is used.

The link $Z_t^d \rightarrow Q_t$ is part of the representation of equation (3.10); i.e., the observed output is obtained by adding the modeled output to the disturbance variable. Thus all links to Q_t are fixed, therefore this node is excluded from training.

The next question concerns the used training signals. We used $u(t) = u_{\text{step}}\sigma(t)$, which is similar to the step response $\sigma(t)$, but with different step heights u_{step}. As second training signal the impulse response was used. This means that

$$u_t = \begin{cases} u_{\text{impulse}} & t = 0 \\ 0 & t > 0 \end{cases} \tag{4.18}$$

with u_{impulse} as random number. Both signals are frequently used in control theory.

For the training 40 time-series were generated using Simulink. Then the Bayesian network is trained for 5 iterations to adapt the weights of the state nodes. Afterwards, new simulations are started to gain fresh training signals. The procedure is repeated 4 times so that all in all 20 iterations are made. According to our experience 20 iterations are sufficient for training. As training algorithm the EM-algorithm is used, as this algorithm is also able to deal with hidden variables, e.g. the state nodes X_1^s and X_2^s. The experiments carried out are the same as in the last section, first the desired value is changed from 0 to 10 to evaluate the reference reaction. After convergence the disturbance is increased. Thus, in the first moment, the output signal increases from 10 to 11. Each experiment is repeated 10 times so that also the stability of the introduced approach can be judged.

Table 4.4: Results of experiments with state-space model, system 1

Experiment number	1	2	3	4	5	6	7	8	9	10	Mean
$I_{\mathrm{d}}(z^d = 0)$	7.46	7.43	7.45	7.61	7.37	7.69	7.67	7.63	7.51	noConv.	7.53
$e_\infty(z^d = 0)$	0.01	-0.01	0.01	0.00	-0.01	0.01	0.02	0.01	0.00	noConv.	0.01
Overshoot	0.53	0.31	0.52	0.45	0.38	0.55	0.49	0.50	0.53	noConv.	0.48
$I_{\mathrm{d}}(z^d = 1)$	0.15	0.15	0.16	0.16	0.15	0.18	0.17	0.17	0.16	noConv.	0.16
$e_\infty(z^d = 1)$	0.03	0.02	0.03	0.04	0.02	0.04	0.05	0.04	0.03	noConv.	0.03
$t_s(z^d = 0,1\%)$ [s]	0.70	0.60	0.70	0.70	0.50	0.75	0.75	0.75	0.70	noConv.	0.68
$t_s(z^d = 0,3\%)$ [s]	0.50	0.35	0.45	0.50	0.40	0.55	0.55	0.55	0.50	noConv.	0.48
$t_s(z^d = 1,1\%)$ [s]	0.55	0.45	0.50	0.55	0.45	0.55	0.60	0.55	0.55	noConv.	0.53
$t_s(z^d = 1,3\%)$ [s]	0.30	0.30	0.30	0.35	0.30	0.35	0.35	0.35	0.35	noConv.	0.33

Table 4.5: Results of experiments with state-space model, system 2

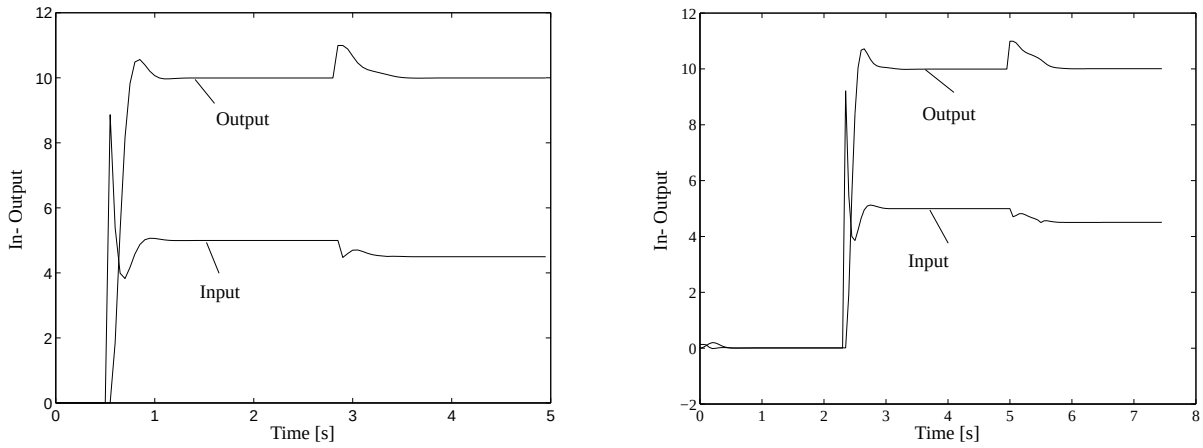
Experiment number	1	2	3	4	5	6	7	8	9	10	Mean
$I_{\mathrm{d}}(z^d = 0)$	9.45	9.46	9.47	9.50	9.50	9.40	9.43	9.51	9.52	9.44	9.47
$e_\infty(z^d = 0$	-0.01	-0.00	-0.01	-0.01	-0.00	-0.00	-0.01	-0.01	-0.01	0.00	0.00
Overshoot	0.72	1.03	0.79	0.68	1.11	1.29	0.78	0.76	0.67	0.93	0.88
$I_{\mathrm{d}}(z^d = 1)$	0.26	0.27	0.27	0.29	0.27	0.26	0.27	0.28	0.29	0.28	0.27
$e_\infty(z^d = 1)$	0.01	0.02	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
$t_s(z^d = 0,1\%)$ [s]	0.55	0.85	0.65	0.55	0.85	0.70	0.60	0.80	0.75	0.85	0.72
$t_s(z^d = 0,3\%)$ [s]	0.45	0.70	0.45	0.45	0.70	0.55	0.45	0.50	0.45	0.55	0.53
$t_s(z^d = 1,1\%)$ [s]	0.70	0.75	0.75	0.85	0.70	0.75	0.70	0.80	0.90	0.85	0.78
$t_s(z^d = 1,3\%)$ [s]	0.60	0.60	0.60	0.65	0.60	0.60	0.60	0.60	0.65	0.65	0.62

In general, the results for the trained Bayesian controller, listed in tables 4.4 to 4.6, are similar to the results obtained for the analytical case. The most crucial point is the missing convergence in the tenth case of table 4.4. Thus the mean given in the last column is taken from the first nine cases.

The squared error sum I_{d} of the trained Bayesian controller is better than in the analytical case. The reason for this fact gets obvious, when the input signal for the trained case is compared with the analytical case. It becomes evident that the trained controller shows a stronger reaction as the controller whose weights are set analytically.

The larger input signals lead to a larger overshoot of 0.48 in comparison to 0.02 in the analytical case. The mean amount of the steady state error for the reference reaction is 0.01. In the worst case – not including case 10 which shows no convergence – the steady state error e_∞ is 0.02, which corresponds to 2 ‰ of the desired value. For the disturbance reaction the steady state error amounts to 0.05 in the worst case which is worse than the steady state error in the reference reaction.

The results for system 2, shown in table 4.5, are slightly better than the results for system 1.



(a) Calculated weights

(b) Trained weights

Figure 4.4: Signals for system 2

Particularly convergence was achieved in all cases. The established tendency, observed in the evaluation of the first test-system, is also valid for system 2. The squared error sum is slightly smaller as in the analytical case. This is due to a larger input (Compare figures 4.4(a) and 4.4(b)) which also leads to a larger overshoot.

The mean of the steady state error is less or equal to 1‰ of the desired value for both the reference and disturbance reaction.

The settling time is better for the inferred model, e.g. the inferred model needs 0.45 s until the new desired value is reached, the average time for the trained model is 0.72 s.

The model for system three show the limits of the state-space approach. The average steady-state error for the third test system is 0.26, but the median of 0.02 shows that in most cases an acceptable result is obtained.

Also the steady state error for the disturbance reaction shows that the training process is at its limits. The average steady state error is 0.35, which is beyond 3% of the desired value. But also for the disturbance reaction the median of approximately 0.05 is acceptable.

The settling time $t_s(z^d = 0, c\%)$ is much larger as in the analytical case. That is due to two cases with a remaining error larger than 1 or 3% (Confer table 4.6, experiment number 3 and 4).

When these two cases are left out of the calculation the mean $t_s(z^d = 0, 1\%)$ is equal to 0.84s.

Table 4.6: Results of experiments with state-space model, system 3

Experiment number	1	2	3	4	5	6	7	8	9	10	Mean
$I_{\mathrm{d}}(z^d = 0)$	10.03	9.82	10.01	25.37	10.06	9.84	9.87	10.00	9.86	11.39	11.63
$e_\infty(z^d = 0$	0.03	0.01	-0.51	1.95	0.02	0.00	0.01	0.02	0.00	0.08	0.26
Overshoot	1.15	0.86	1.75	7.27	1.24	0.91	1.04	1.26	1.04	0.88	1.74
$I_{\mathrm{d}}(z^d = 1)$	0.52	0.45	0.02	0.15	0.52	0.39	0.49	0.43	0.36	0.92	0.43
$e_\infty(z^d = 1)$	0.09	0.06	0.41	2.54	0.04	0.03	0.04	0.05	0.02	0.19	0.35
$t_s(z^d = 0, 1\%)$ [s]	1.00	0.60	4.70	18.00	1.00	0.45	0.50	0.90	0.50	1.80	2.95
$t_s(z^d = 0, 3\%)$ [s]	0.50	0.40	4.70	18.00	0.55	0.45	0.45	0.55	0.45	0.65	2.67
$t_s(z^d = 1, 1\%)$ [s]	1.85	1.40	3.70	14.05	1.30	0.95	1.25	1.00	0.85	5.50	3.19
$t_s(z^d = 1, 3\%)$ [s]	1.00	0.95	3.70	14.05	0.95	0.75	0.90	0.80	0.70	1.80	2.56

As summary of the experiments described in section 4.1 and 4.2 it is concluded that Bayesian networks are a suitable mean for self-adaptive control, provided a successful training process is given. In section 4.3 a second model based on the difference equations, introduced in section 3.1.3, is examined. This model has the great advantage of less hidden nodes. The Experiments described in section 4.3 have shown that a higher accuracy in control is achieved, even if less examples are needed for training.

4.3 Higher order Markov-model

In section 3.1.3 a second possibility for describing linear dynamic systems was introduced. In the description by difference equation (confer equation (3.34))

$$y^m = -\sum_{i=1}^{n} a_i y^m_{t-i} + \sum_{i=1}^{n} b_i u_{t-i} \tag{4.19}$$

no hidden nodes are left. Instead of using state nodes, former in- and outputs u_{t-i} and y^m_{t-i} respectively are taken to calculate a prediction of the next output. As in the state-space description also the disturbance variable has to be taken into account. That is, the estimation of z^d_t has to be added to y^m_t to figure out the observed output. This consideration leads to the model depicted in figure 4.5 which shows a dynamic system of second order.

In comparison to the state-space model it is not possible to save the layer for the modeled output y^m_t. Adding the disturbance variable z^d_t to y^m_t directly (confer figure 4.6) leads to wrong predictions in presence of a disturbance.

When looking at figure 4.5, particularly at node Y^m_{t+2}, one notices that the Markov assumption, explained in section 2.4, is not met as Y^m_{t+2} depends also on Y^m_t and U_t. For models of higher order a recourse to Y^m_{t-i} and $U_{t-i}, i > 1$ is necessary for the prediction of Y^m_{t+2}. This causes prob-

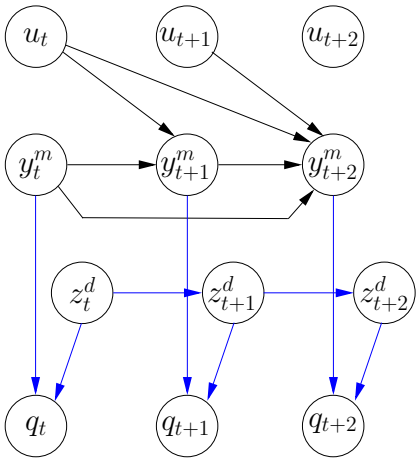


Figure 4.5: Modeling of equation (3.34) by a dynamic Bayesian network

lems, as some of the toolboxes for Bayesian networks, including the one implemented by K. P. Murphy which is used for our experiments, are not able to deal with Markov models of higher order. There are two obvious solutions.

- Adding additional nodes to the model with the task to transfer the values of random variables, belonging to former time-slices, to the current one [DDN02b].

- Reimplementation of a small part of the Bayesian network toolbox. Due to the evaluation mechanism, the unrolling of the dynamic Bayesian network, the necessary expansion of the toolbox is restricted to the generation of the Bayesian network. The evaluation algorithm does not need to be changed. Also the training algorithm is left untouched as it is also based on the unchanged unrolled Bayesian network. The results obtained with this model to control systems of second and third order are discussed in section 4.4.

The first approach, used tor the experiments in this section, has the advantage that it can be used for rapid prototyping so that the supposition that the difference equation model is advantageous for a Bayesian controller can be checked easily. When additional nodes are used, the same value is represented by two different nodes in different time-slices. This leads to the structure in figure 4.7.

Of course it has to be guaranteed that the same value is assigned to nodes representing the same value. For example the values u_t and y_t^m are represented in the first and second time-slice in figure 4.7. It is guaranteed by two different mechanisms that all nodes representing u_t or y_t^m get the same value. For input nodes the value is entered twice as evidence. For systems that show no

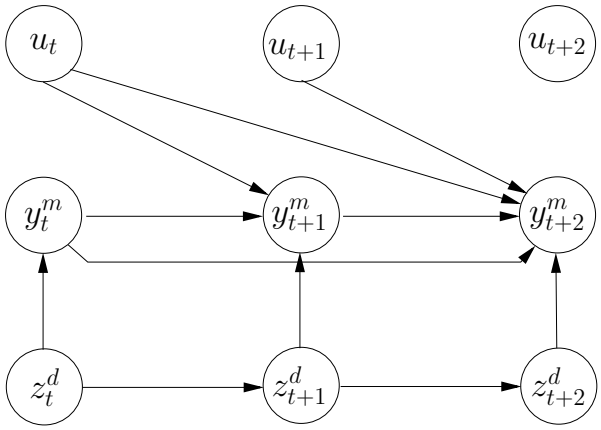


Figure 4.6: Faulty model which leads to incorrect prediction

immediate reaction on changes of the input nodes, the links between u_t and y_t^m are superfluous. Thus, one of the input nodes is unnecessary.

For the modeled output of the system this is not possible as there is no evidence given for y_t^m. The solution in this case is a link between two nodes representing the same value with a fixed weight of one. An example is the link between y_t^m in the time-slice at the left and in the middle of figure 4.7.

The next step concerns the training of the Bayesian network. Our aim using the difference equation is to use as less hidden nodes as possible. When assuming that there is no disturbance during training, it is possible to set $z^d = 0$ and $y^m = q$. Thus there are no hidden nodes left.

Similar to the experiments with the state-space descriptions, some of the parameters are clamped in order to guarantee that the controller acts as intended. The following parameters are excluded from training:

- Variance of the input nodes γ_U
- Variance of the modeled output γ_{Y^m}
- Both edges to Q has a fixed weight of one, the variance γ_Q is set to a small value, for the described experiments γ_Q = 0.01
- The connection between Z_t^d and Z_{t+1}^d has a fixed weight of one, the variance of the first time-slice is set to three (at the first time-slice it is not possible to make proper predictions as the information from former signals is missing), in the remaining time-slices γ_{Z^d} = 0.01.

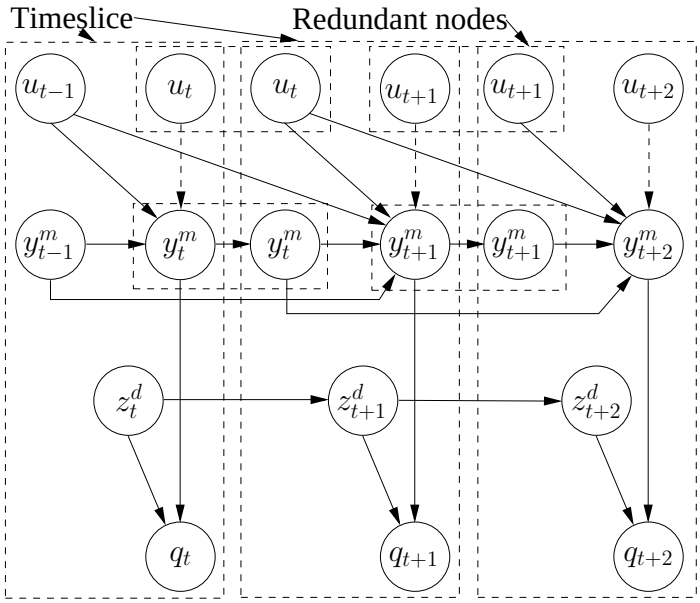


Figure 4.7: Model for difference equation with redundant nodes

As there are no hidden nodes left, the number of iterations is reduced to five, but of course the convergence of the EM-algorithm is observed immediately. The number of data sets used for training remains 40.

The results obtained for system 1 with the difference equation model are depicted in table 4.7. The most important result is that the control signal converged in all cases; i.e., the difference equation model shows the stability necessary for real world applications. This argument is supported by the low distortion of the training results. When table 4.4 and 4.7 are compared the most impressive result is that the dynamic system controlled with the difference equation model shows nearly no overshoot in comparison to an overshoot of 0.48 of the state-space model.

Table 4.7: Results of experiments with higher order Markov model, system 1

Experiment number	1	2	3	4	5	6	7	8	9	10	mean
$I_{\mathrm{d}}(z^d = 0)$	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51
$e_\infty(z^d = 0)$	-0.09	-0.09	-0.05	-0.07	-0.05	-0.10	-0.08	-0.07	-0.06	-0.06	0.07
Overshoot	-0.09	-0.09	-0.05	-0.07	-0.05	-0.10	-0.08	-0.07	-0.06	-0.06	-0.07
$I_{\mathrm{d}}(z^d = 1)$	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
$e_\infty(z^d = 1)$	0.02	0.02	0.05	0.04	0.05	0.00	0.03	0.04	0.04	0.05	0.03
$t_s(z^d = 0, 1\%)$ [s]	0.75	0.75	0.60	0.60	0.60	2.40	0.65	0.65	0.60	0.60	0.82
$t_s(z^d = 0, 3\%)$ [s]	0.45	0.45	0.40	0.40	0.40	0.45	0.40	0.40	0.40	0.40	0.42
$t_s(z^d = 1, 1\%)$ [s]	0.30	0.60	0.30	0.30	0.30	0.60	0.30	0.30	0.30	0.30	0.36
$t_s(z^d = 1, 3\%)$ [s]	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

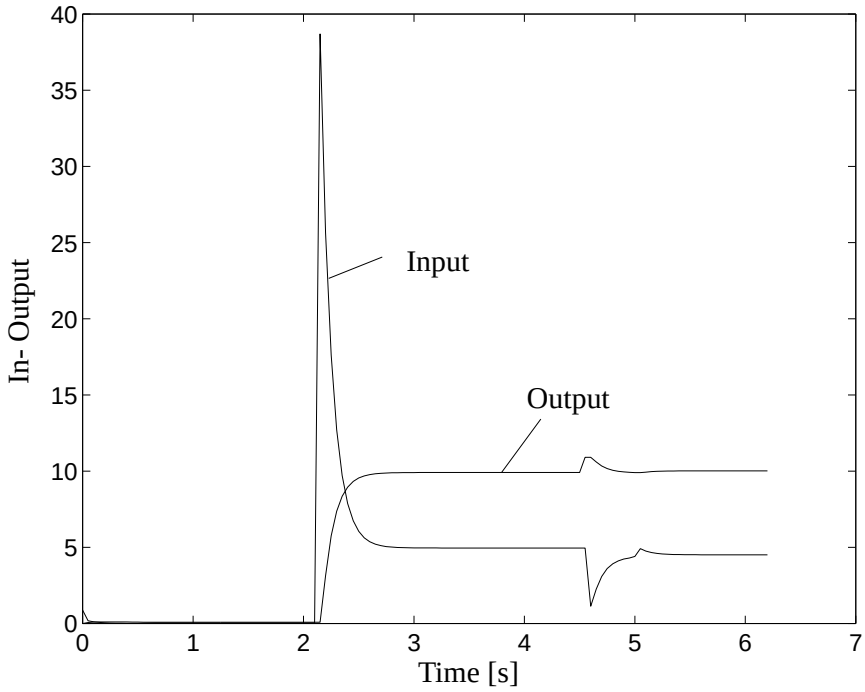


Figure 4.8: Signals of difference equation controller, system 1

The squared error sum for the reference reaction is better for the state-space model. The reason for this is the same as in the comparison between the analytical and trained state-space model. The larger overshoot results in a faster reduction of the difference between observed output and desired value and therefore to a smaller squared error sum (compare figure 4.8).

For the disturbance reaction the difference equation model shows better results, a hint that the difference equation model is trained more accurately. Comparing the settling time shows similar results. The state-space model is slightly better as long as the reference reaction is compared (0.68 s in comparison to 0.82 s of the difference equation model). If the disturbance reaction is compared the difference equation model shows better results (0.33 s vs. 0.25 s).

Using system 2 for comparison shows similar results as system 1. Less overshoot (0.45 vs. 0.88) of the difference equation model which leads to a slightly worse squared error sum I_{d} (9.82 vs. 9.47). Despite the worse squared error sum the settling time, that is the time until the desired value is reached, is shorter for the difference equation model.

As mentioned in section 4.2, system 3 is suited to show the limit of the state-space approach. Comparing the results of the state-space model and the difference equation model, depicted in tables 4.6 and 4.9, shows that the new structure clearly overcomes this limit.

Table 4.8: Results of experiments with higher order Markov model, system 2

Experiment number	1	2	3	4	5	6	7	8	9	10	mean
$I_{\mathrm{d}}(z^d = 0)$	9.82	9.83	9.82	9.82	9.82	9.82	9.82	9.83	9.83	9.82	9.82
$e_\infty(z^d = 0)$	0.03	-0.03	-0.05	-0.04	-0.03	-0.05	-0.04	-0.03	-0.04	-0.04	0.04
Overshoot	0.45	0.45	0.43	0.44	0.45	0.43	0.44	0.46	0.44	0.44	0.45
$I_{\mathrm{d}}(z^d = 1)$	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
$e_\infty(z^d = 1)$	-0.04	-0.04	-0.06	-0.05	-0.04	-0.05	-0.05	-0.04	-0.05	-0.05	-0.05
$t_s(z^d = 0, 1\%)$ [s]	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
$t_s(z^d = 0, 3\%)$ [s]	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
$t_s(z^d = 1, 1\%)$ [s]	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
$t_s(z^d = 1, 3\%)$ [s]	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25

Table 4.9: Results of experiments with higher order Markov model, system 3

Experiment number	1	2	3	4	5	6	7	8	9	10	mean
$I_{\mathrm{d}}(z^d = 0)$	9.90	9.89	9.89	10.10	9.89	9.89	9.90	9.89	9.91	9.89	9.92
$e_\infty(z^d = 0)$	0.01	0.01	0.01	0.03	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Overshoot	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15
$I_{\mathrm{d}}(z^d = 1)$	0.19	0.19	0.19	0.18	0.19	0.19	0.19	0.19	0.19	0.19	0.19
$e_\infty(z^d = 1)$	-0.04	-0.04	-0.04	0.00	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.03
$t_s(z^d = 0, 1\%)$ [s]	0.70	0.65	0.65	1.70	0.65	0.65	0.65	0.65	0.75	0.65	0.77
$t_s(z^d = 0, 3\%)$ [s]	0.45	0.45	0.45	1.05	0.45	0.45	0.45	0.45	0.45	0.45	0.51
$t_s(z^d = 1, 1\%)$ [s]	0.60	0.60	0.60	0.55	0.60	0.60	0.60	0.60	0.60	0.60	0.60
$t_s(z^d = 1, 3\%)$ [s]	0.25	0.25	0.25	0.30	0.25	0.25	0.25	0.25	0.30	0.25	0.26

The results obtained with the difference equation model are in all respects better than these obtained with the state-space model. As usual the state-space model shows the larger overshoot. In this case this does not lead to a smaller squared error sum. The fact that the trained difference equation model sometimes even surpasses the state space model whose weights are set, is particularly from interest. For instance the squared error sum is smaller for both, the reference and the disturbance reaction. Also the settling times are nearly the same. The disadvantage of the approach with redundant nodes, as depicted in figure 4.7, is the overhead for the computation of the redundant nodes which increases linearly with the order of the system.

4.4 Modeling of higher order systems

In sections 4.1 to 4.3 all used test systems are of second order. Even if the approach with redundant nodes is still possible for higher order systems, a lot of overhead is caused by this solution. In principle there are different possibilities to model systems with order $n > 2$ which will be discussed in this section more or less from the theoretical point of view.

- When the state-space approach is used, it is possible to increase the number of state nodes or the dimension of the state node to model a system of higher order.
- Splitting the system in subsystems of second order.
- Splitting of the system in parallel subsystems.
- Usage of higher order Markov models, being able to predict the output by resource to former in- and output signals.
- Approximation of a higher order system by a system of second order.

When the state-space approach is used, it is theoretically also possible to model MIMO systems of higher order. But as seen in section 4.2 the training of the state-space model might cause problems. One additional problem for systems with $n > 2$ is the high number of the state nodes itself. When the junction tree of the higher order model is constructed, the state nodes and all of their parents are all content of one clique. For a SISO system of forth order this leads to nine-dimensional cliques, as $\boldsymbol{X}_{t-1}^{s}$ and U_{t-1} are parents of the state nodes $\boldsymbol{X}_{t}^{s}$. This leads to numerical problems during inversion of the covariance matrix, included in the inference algorithm introduced in [Lau92].

One solution of the problem is to split the system in a serial connection of second order systems. The idea is to split the numerator and denominator of the transfer function

$$G(s) = \frac{N(s)}{D(s)} = \frac{N_1(s)}{D_1(s)} \frac{N_2(s)}{D_2(s)} \cdots \frac{N_{n/2}(s)}{D_{n/2}(s)} \tag{4.20}$$

into polynomials $N_i(s)$ and $D_i(s)$ of second order. This reduces the numerical problems, but there are two restrictions. This approach is only suitable for SISO systems, as the transfer function of MIMO systems are polynomial matrices. The other restrictions is the training which might cause problems.

The second approach is the division in parallel subsystem. Assuming that all poles in $D(s)$ are real

$$G(s) = \sum_{k=1}^{p} \left(\frac{c_{k,1}}{(s - s_k)} + \frac{c_{k,2}}{(s - s_k)^2} + \cdots + \frac{c_{k,r_k}}{(s - s_k)^{r_k}} \right) \tag{4.21}$$

where p is the number of different poles, s_k is a pole of $D(s)$ and r_k denotes how often $(s - s_k)$ occurs the decomposition of $G(s)$. Thus $G(s)$ is decomposed in parallel subsystems, as parallel subsystems are mapped to the sum of their transfer functions. As the number of poles is not always known before modeling starts, this approach cannot be used in all cases.

Table 4.10: Test results for system number 4

	BN4
$I_{\mathrm{d}}(z^d = 0)$	66.45
$e_\infty(z^d = 0)$	0.05
Overshoot	1.80
$I_{\mathrm{d}}(z^d = 1)$	1.44
$e_\infty(z^d = 1)$	0.09
$t_s(z^d = 0, 1\%)\,[s]$	7.12
$t_s(z^d = 0, 3\%)\,[s]$	4.40
$t_s(z^d = 1, 1\%)\,[s]$	10.92
$t_s(z^d = 1, 3\%)\,[s]$	3.68

The most promising approach is the application of higher order Markov models. For the evaluation of this approach the Bayesian toolbox is expanded so that Markov models of higher order can be implemented directly without the usage of redundant nodes. For a third order system

$$G(s) = \frac{0.4s + 2}{0.01s^3 + 0.5s^2 + 0.2s + 1} \tag{4.22}$$

whose step response shows an overshoot, the results are depicted in table 4.10. The results are obtained with a sampling period of $\Delta T = 0.4\,s$ and 40 training examples. Trials with $\Delta T = 0.05\,s$ fail. The reason for this might be the small number of time-slices, so that a long term prediction is not given. For the reference reaction and the disturbance reaction the steady state error is below the 1% level. For a comparison with traditional controller see section 4.5.

The last possibility for modeling higher order systems is to select a model of lower order, for example

$$G(s) = \frac{K}{(1 + T_1 s)(1 + T_2 s)} \tag{4.23}$$

might be used . Other models are introduced in [Unb97a].

4.5 Comparison to PI and Dead-Beat controller

In sections 4.1 to 4.4 the capabilities of Bayesian controllers are examined based on the criteria listed in table 4.2. This section deals with controllers used in industrial practice in order to compare their performance with the Bayesian controller (see also [DDN03b]). Due to the numerous methods to figure out controller settings it is impossible to try out all current approaches.

We have selected the approach by Ziegler and Nichols [ZN42] which, despite its age, is

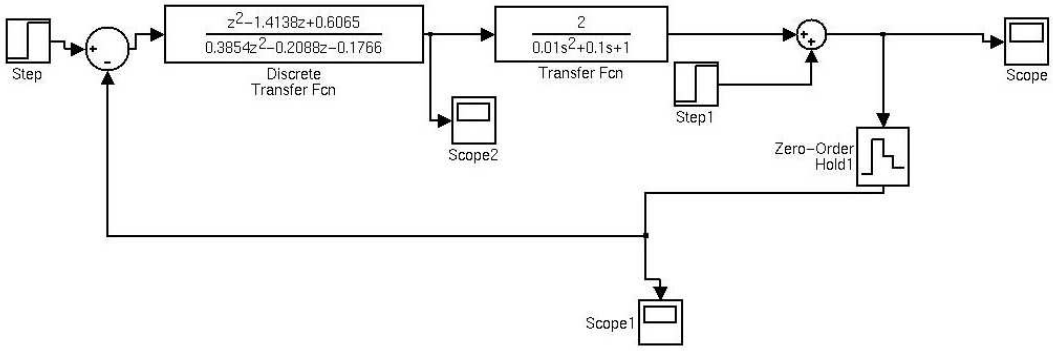


Figure 4.9: Setup for experiments with PI and Dead-Beat controller

Table 4.11: Settings for PI controller

Number	K_{crit}	t_{crit}	Controller $D(z)$
1	32.58	0.1	$\frac{14.661z-6.039}{z-1}$
2	2.25	0.29	$\frac{1.0125z-0.8083}{z-1}$
3	0.213	0.37	$\frac{0.0959z-0.0807}{z-1}$
4	64	0.17	$\frac{28.8z-19.0549}{z-1}$

one of the most commonly used approaches. When this approach is applied to digital control a high sampling period has to be selected so that the controller operates nearly in a continuous time space. Moreover this mechanism is originally developed for overdamped systems, a precondition only given for system 1.

Dead-Beat controllers are particularly designed for digital control. They guarantee that the desired value is reached within a finite number of time steps, depending on the order of the system and the dead time. Therefore this type of controller is suited to give a lower bound for the settling time t_s. But is has to be kept in mind, that this method is based on the exact knowledge of the transfer function and of the desired reference reaction. Particularly, the first requisite is usually not given. For our experiments the control loop depicted in figure 4.9 is used.

To figure out the settings according to Ziegler and Nichols the controller is first used as pure P-controller, i.e. $G(z) = K_{\text{c}}$. The controller gain K_{c} is increased until the closed loop starts oscillation. The controller gain K_{crit} and the critical time t_{crit} are measured. For our test systems these values are given in table 4.11. Afterwards the controller is calculated according to table 3.1. The resulting controller is dedicated for continuous time. To adapt it to discrete time

Table 4.12: Settings for Dead-Beat controller

Number	Transfer function $G(z)$	Controller $D(z)$
1	$\frac{0.0791z^{-1}+0.0188z^{-2}}{1-0.9578z^{-1}+0.0067z^{-2}}$	$\frac{1-0.9578z^{-1}+0.0067z^{-2}}{0.0978-0.0791z^{-1}-0.0188z^{-2}}$
2	$\frac{0.2088z^{-1}+0.1766z^{-2}}{1-1.4138z^{-1}+0.6065z^{-2}}$	$\frac{1-1.4138z^{-1}+0.6065z^{-2}}{0.3854-0.2088z^{-1}-0.1766z^{-2}}$
3	$\frac{1.1286+1.0377z^{-1}}{1-1.5622z^{-1}+0.7788z^{-2}}z^{-1}$	$\frac{1-1.5622z^{-1}+0.7788z^{-2}}{2.1663-1.1286z^{-1}-1.0377z^{-2}}$
4	$\frac{0.544z^{-1}+0.0529z^{-2}-0.0127z^{-3}}{1-1.5730z^{-1}+0.8651z^{-2}-2.0612e-09z^{-3}}$	$\frac{1-1.573z^{-1}+0.8651z^{-2}-2.0612e-09z^{-3}}{0.5842-0.544z^{-1}-0.0529z^{-2}+0.0127z^{-3}}$

the controller setting was mapped to discrete time domain using the according matlab function. This mapping was done using the same sampling period ΔT as for the Bayesian controller to ensure compatibility of results. But this selection is questionable as it does not guarantee the requirement that the controller operates in nearly continuous time. For the tests, discussed in this section, only PI-controllers are used. Tests with PID controller yield no satisfying results. Possible reasons may be the fact that the Ziegler-Nichols approach is usually restricted to overdamped systems in continuous time domain.

For the Dead-Beat controller the dynamic system is mapped to discrete time domain and afterwards the controller was calculated. The results are depicted in table 4.12.

The controllers listed above are compared to a Bayesian controller, based on the difference equation model. In difference to the results, discussed in section 4.3, the models are implemented directly using an expansion of the BN-toolbox for higher order Markov models. Two different settings for the Bayesian network are tested. The first version, called BNT1, uses only u_{t+1} for the calculation of u_{new}. That is the number of nodes k in equation (4.11) is set to 1. Hence

$$u_{\text{new}} = u_{t+1} \quad . \tag{4.24}$$

The second version BNT4 uses four nodes for the calculation of u_{new}, i.e

$$u_{\text{new}} = \frac{1}{10}\left(4u_{t+1} + 3u_{t+2} + 2u_{t+3} + u_{t+4}\right) \quad . \tag{4.25}$$

As figures 4.10(a) and 4.10(b) show, one of the main effects is that the input signal is damped down. This makes sense, because the actuator might be too slow to follow the oscillation. The

Table 4.13: Test results for system number 1 and 2

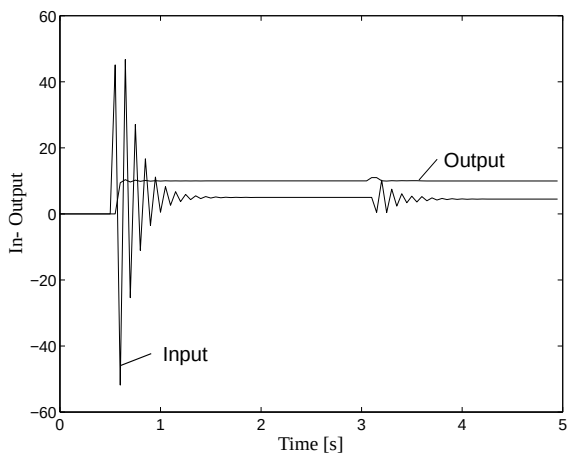
	Test System 1 $\Delta T = 0.05\ s$				Test System 2 $\Delta T = 0.05\ s$			
	ZN	DB	BN1	BN4	ZN	DB	BN1	BN4
$I_{\mathrm{d}}(z^d = 0)$	12.43	5.2	4.98	8.09	15.39	6.04	5.02	9.66
$e_\infty(z^d = 0)$	0	0.01	0.03	0.06	0	0	0.01	0.02
Overshoot	8.82	0.01	0.06	0.06	4.03	0	0.36	0.54
$I_{\mathrm{d}}(z^d = 1)$	0.12	0.05	0.09	0.11	0.15	0.06	0.11	0.16
$e_\infty(z^d = 1)$	0	0.01	0.02	0.03	0	0	0.03	0.04
$t_s(z^d = 0, 1\%)$ [s]	1.1	0.1	0.10	0.50	3.35	0.1	0.35	0.45
$t_s(z^d = 0, 3\%)$ [s]	0.85	0.1	0.10	0.35	2.5	0.1	0.20	0.40
$t_s(z^d = 1, 1\%)$ [s]	0.55	0.1	0.15	0.55	1.5	0.1	0.15	0.30
$t_s(z^d = 1, 3\%)$ [s]	0.3	0.05	0.15	0.20	0.5	0.1	0.15	0.25

Table 4.14: Test results for system number 3 and 4

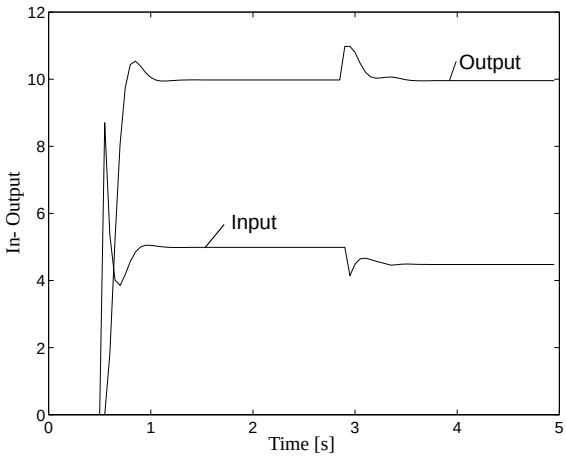
	Test System 3 $\Delta T = 0.05\ s$				Test System 4			
	ZN	DB	BN1	BN4	ZN	DB	BN1	BN4
$I_{\mathrm{d}}(z^d = 0)$	21.51	6.15	4.97	9.89	13.87	40.20	39.80	66.45
$e_\infty(z^d = 0)$	0.01	0	0.00	0.00	0.00	0	0.03	0.05
Overshoot	1.70	0	0.00	1.15	6.34	0.22	0.23	1.80
$I_{\mathrm{d}}(z^d = 1)$	0.21	0.06	0.12	0.19	0.31	0.40	0.94	1.44
$e_\infty(z^d = 1)$	0	0	0.03	0.04	0	0	0.05	0.09
$t_s(z^d = 0, 1\%)$ [s]	6.05	0.1	0.05	0.65	2.65	1.2	3.88	7.12
$t_s(z^d = 0, 3\%)$ [s]	4.4	0.1	0.05	0.45	2.0	0.8	2.32	4.40
$t_s(z^d = 1, 1\%)$ [s]	2.5	0.1	0.55	0.60	1.5	0.4	5.48	10.92
$t_s(z^d = 1, 3\%)$ [s]	1.05	0.1	0.15	0.25	0.85	0.4	1.20	3.68

results for Ziegler Nichols are worst. Particular for system 4, the sampling period ΔT has to be decreased to 0.05 s. The other systems were tested with $\Delta T = 0.4\ s$. That means that the squared error sum in table 4.14 should not be compared to BN1, BN4, and the Dead-Beat controller.

The comparison for system 1 shows that the Bayesian network BN1 shows similar results as the Dead-Beat controller. For an overdamped system it might be sensible to select only one node for the calculation of u_{new}. In this case similar settling times t_s are obtained. Systems two and three both have a damping $D < 1$, so that oscillation is possible. The results for BN1 and DB are similar, but as figure 4.10(a) shows, too large input signals lead to oscillation. Also the Dead-Beat controller shows a sudden change of the input signal at the beginning (see figure 4.11 for the in- and output signals of the Dead-Beat controller) For the Dead-Beat controller an appropriate mean to limit the signals of the actuator is to increase the sampling period ΔT, for Bayesian



(a) BN1



(b) BN4

Figure 4.10: Signals of Bayesian controller, system 2, based on difference equation

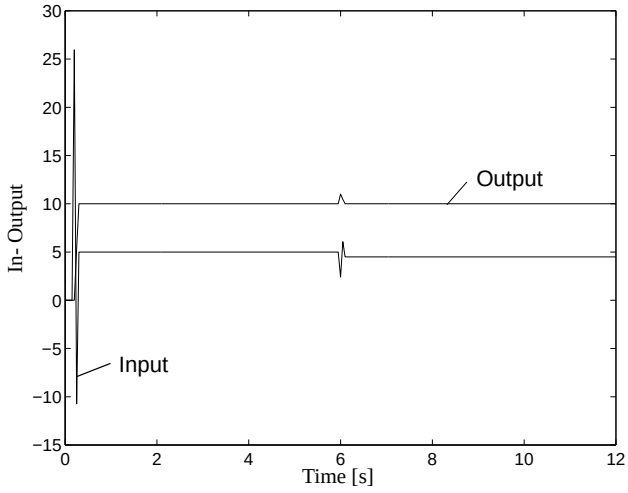


Figure 4.11: Signals of system 2 for the Dead-Beat controller

networks the number of nodes used for calculation of u_{new} can be increased. This leads of course to an increase of the squared error sum, as the first input signal is decreased. Additionally, the settling time is increased.

The right hand side of table 4.14 shows that the Bayesian network approach is also suitable for systems of third order. The steady state error is below 1% of the desired value. But the settling time is worse than for the Dead-Beat controller.

In chapter 4 the application of Bayesian controllers to linear systems has been discussed. The next chapter discusses the modeling of non-linearities with Bayesian networks. In some books about non-linear control, some frequently occurring functions are listed. The idea is to design models for these elements which can be combined to more complex models and have the advantage, that a lot of a-priori knowledge can be used for the design. Thus some of the parameters can be clamped, so that the training effort is reduced.

It is hard to compare the different methods. The Bayesian controller has a high time-complexity. According to [Lau92] the "most complex operation is the weak marginalization over a given clique. If the clique contains discrete variables $X \in \boldsymbol{\Delta}_{\mathcal{G}}$ with state space of cardinality n_X and q continuous variables, then the computational complexity is of the order of magnitude $q^3 \prod_{X \in \boldsymbol{\Delta}_{\mathcal{G}}} n_X$".

In this chapter only continuous nodes are used, thus an upper limit of the time complexity is q^3. The largest clique of the state-space description contains $2n$ state nodes plus the input nodes (remember that at least one clique must contain the state nodes and its parents). The time complexity of q^3 results from matrix inversion, hence it makes no difference that only strong marginalization is applied during inference.

The training time of a state-space model with 25 nodes is approximately 1600 s, the evaluation time 0.43 s. Thus the run-time is the weak point of the Bayesian network. Its advantage is the ability to adapt itself to different systems.

In contrary to the Bayesian controller the Dead Beat controller is based on a mathematical description. If this description is available or can be estimated, the Dead-Beat controller is a good choice, because it guarantees a minimal settling time and the control signals are easily calculated.

The approach of Ziegler and Nichols is restricted to overdamped systems. Additionally, it is originally developed for continuous time systems. Thus it is a very valuable empirical formula, but far from optimal.

So the Bayesian controller is recommended if the system is unknown, In the other case the Dead-Beat controller offers better features. But the reader should keep in mind that the Bayesian controller is in a prototypical stage (compare chapter 9). Moreover the comparison of this chapter

is incomplete as other self-adaptive methods are not regarded.

Chapter 5

Bayesian networks for modeling nonlinear dynamic systems

This section discusses the approximation of a nonlinear function by piecewise linear approximation. In a first approach, introduced in section “Linear Approximation”, it is shown how a hybrid Bayesian network simulates several Taylor series at the same time.

All input variables that have a nonlinear influence on the output are represented both by a discrete and a continuous variable. The discrete variable selects a point u_a closest to the current input u. In first models an extra node E_Q is used that calculates the quantization error $e_Q = u - u_a$. This model is used without modifications for the modeling of calibration in hydroforming.

In section “Simplification of linear approximation” it is shown how this node is saved. This simplified version is used for the model of saturation.

In Section 5.2 the approximation of a nonlinear function by several Taylor series is combined with the modeling of dynamic systems. Additionally the results of section 8 are applied to restrict the run-time.

5.1 Prototypical modeling of nonlinear units

Linear Approximation

A differentiable function f is approximated by a Taylor series

$$\grave{f}(u) = f(u_a) + f'(u_a)(u - u_a), \tag{5.1}$$

at a point u_a. The function $\grave{f}$ denotes the approximation of f, f' the first derivative and u_a an arbitrary point.

To model the approximation of a function f with several Taylor series the model depicted in figure 5.1 is used. The mean of $f(U)$'s distribution, representing $f(u)$,

$$\alpha_{f(U)} + \beta_{f(U)} e_Q = \grave{f}(u) \tag{5.2}$$

is equal to the approximation of the function f, provided that the mean

$$\alpha_{f(U)} = f(u_a) \ , \tag{5.3}$$

and the weight

$$\beta_{f(U)} = -f'(u_a) \tag{5.4}$$

is equal to the first derivative of f at point u_a. The node E_Q calculates the difference between the input u and its closest point u_a. This is easily obtained by setting the mean of E_Q

$$\alpha_{E_Q} = u_a \tag{5.5}$$

and the weight β_{E_Q} between node U and E_Q

$$\beta_{E_Q} = -1 \ . \tag{5.6}$$

It remains to show how to select the correct points u_a. Usually a link points from the cause to the effect of an event, so that a link $U \rightarrow U_\mathrm{d}$ would be naturally. The introduction of such a link contradicts the assumption that there are no continuous parents of discrete nodes. For exceptions see [Mur99; LSK01].

Our experiments have shown that triggering of U_d also works with a link $U_\mathrm{d} \rightarrow U$. To understand how the continuous node triggers the discrete node assume that u is closest to u_a. Thus the amount $|u_a - u| < |u_i - u|, a \neq i$. Therefore the probability distribution of node U is maximal if $U_\mathrm{d} = a$ which leads to an increased probability of $U_\mathrm{d} = a$. The increased probability for $U_\mathrm{d} = a$ is used to select the corresponding parameters for the node representing $f(u)$.

In order to obtain acceptable training results the following two conditions have to be met.

- The selected points $u_1, u_2, \cdots, u_k$ (k equal to the number of states of U_d) should be near the centers of the straight lines used to approximate the function f. This is a matter of a good initialization.

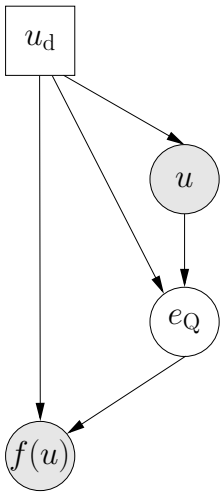


Figure 5.1: Linear approximation by multiple Taylor series

- The standard deviations associated with the different means u_i of U should be in the same order of magnitude as the difference between two neighbored points u_i and u_j. If the scattering of the data is different for different regions of the function f, the (co)variances can be clamped or tied. At least the BN-toolbox, used for the experiments, offers these features. For a discussion how the EM-algorithm is changed to enable clamped or tied covariances see [Mur98a]. If the size of the regions differs severely it is necessary to divide some larger regions, even if they might be modeled by one line.

A critical point in training are test-plans, if one state represents only one setting, and the second state several settings. As an example imagine a model of the threshold process, shown in figure 5.2.

The first state represents an input $u = 1.5$, the second state all inputs between 2.5 and 5.5. Assuming that each experiment is repeated 6 times, the variance for the second state is estimated at

$$\gamma_2 = \frac{1}{23}\left(6(2.5-4)^2 + 6(3.5-4)^2 + 6(4.5-4)^2 + 6(5.5-4)^2\right) = 1.304 \ . \tag{5.7}$$

For the first state variance is zero. In our example $\gamma_1 = 0.0025 \ll \gamma_2$ is used to demonstrate the effect of two heavily different covariances. A comparison of the two Gaussian distributions $p(u|\alpha_1 = 1.5, \gamma_1 = 0.0025)$ and $p(u|\alpha_2 = 4, \gamma_2 = 1.304)$ is shown in figure 5.3.

It can be seen that close to $u = 1.5$ the probability of state 1 is much larger than of state 2. Figure 5.4 (which shows the same two probability density functions) however, exhibits the problem that the probability of state 2 is larger than for state 1 if the input is smaller than 1.25. Thus the model will fail to make predictions for values smaller than 1.25.

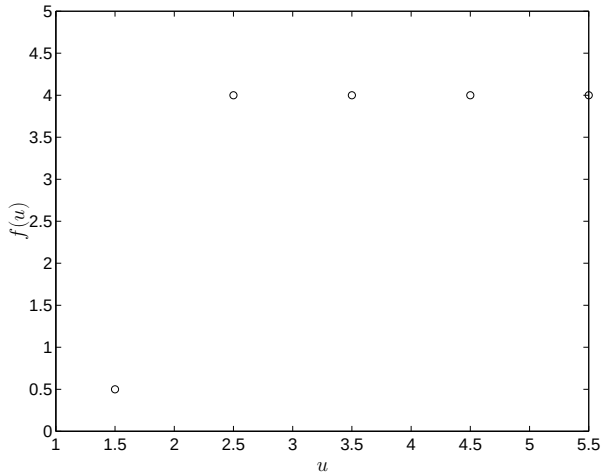


Figure 5.2: Example for a threshold function

There are two possible solutions for the problems just discussed. The first is to use an additional state with a similar variance, so that state one is responsible for small inputs, state 2 models inputs close to the threshold and state three is used for inputs beyond the threshold. An example is the model for the calibration process, treated in section 7.1.1.

The second solution is to tie the covariances for different states, that is to change the maximization step of the EM algorithm, so that $\gamma_i = \gamma_j$ for different states $i \neq j$.

Simplification of linear approximation

In the last section the approximation of an arbitrary function by several Taylor series is discussed. This pattern is successfully applied to model preforming and calibration, discussed in section 7.1.1. Figure 5.1 shows that an extra node E_Q for the quantification of the error is used. To simplify this model the approximation of a function f at a point u_a

$$\begin{aligned} \grave{f}(u) &= f(u_a) + f'(u_a)(u - u_a) && (5.8) \\ &= f(u_a) - f'(u_a)u_a + f'(u_a)u && (5.9) \end{aligned}$$

is split in a constant term $f(u_a) - f'(u_a)u_a$ and a second term $f'(u_a)u$ depending on u. Setting the offset of the output node $f(U)$ to

$$\alpha_{f(U)} = f(u_a) - f'(u_a)u_a \tag{5.10}$$

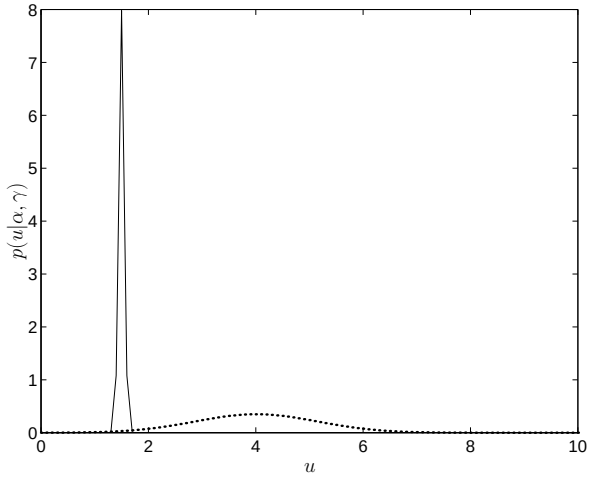


Figure 5.3: Two Gaussians

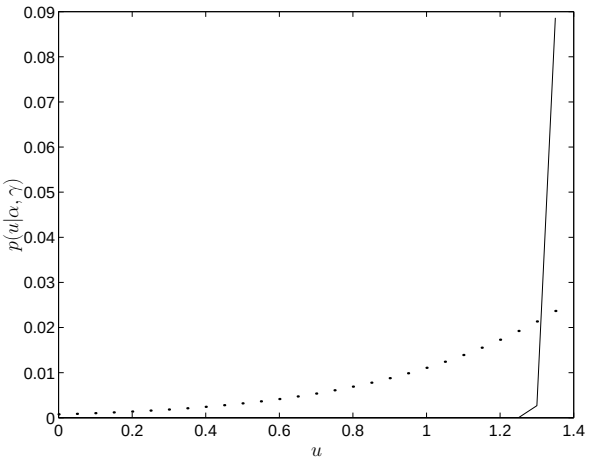


Figure 5.4: Critical point

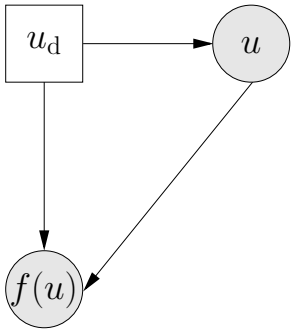


Figure 5.5: Simplified linear approximation

and the weight to

$$\beta_{f(U)} = f'(u_a) \tag{5.11}$$

shows that a continuous node with a discrete parent is able to approximate a function f. The used model is depicted in figure 5.5.

A precondition is that the states u_d are triggered so that the probability of one state is usually much higher than the probability of the remaining states. This is again achieved by setting

$$\alpha_U(u_\mathrm{d} = a) = u_a \tag{5.12}$$

the mean for node U close to the selected operating points u_a. The number of states of U_d is of course equal to the number of operating points. The standard deviation $\sqrt{\gamma_U}$ should be selected similar to the distance between two neighbored operating points. Usually it is sufficient to select

a suitable value as initialization to start the training process. During the training process the value for γ_U is improved by the EM algorithm.

Saturation

One of the nonlinearities discussed in section 3.2 is the saturation f_{sat} that is described by equation (3.48). This function can be modeled by three straight lines. For the first one, a point $u_1 < u_{\mathrm{sat}}$ has to be selected which models the region $u \leq u_{\mathrm{sat}}$. Therefore $f_{\mathrm{sat}}(u_1) = -y^m_{\max}$. The derivative in that region vanishes ($f'_{\mathrm{sat}}(u_1) = 0$) as the output does not depend on the input.

To model the second region a point u_2 obeying $-u_{\mathrm{sat}} \leq u_2 \leq u_{\mathrm{sat}}$ is selected. The slope $f'_{\mathrm{sat}}(u_2) = \frac{y^m_{\max}}{u_{\mathrm{sat}}}$ is equal to the quotient of the maximal output $y^m_{\max}$ by the input u_{sat}. Provided that u_2 is in the center of the region $u_2 = 0$, the function value $f_{\mathrm{sat}}(u_2) = 0$.

Similar to u_1, a point u_3 with $u_3 > u_{\mathrm{sat}}$ is selected with $f_{\mathrm{sat}}(u_3) = y^m_{\max}$ and $f'_{\mathrm{sat}}(u_3) = 0$. The consideration above results in the following means

$$\alpha_U(U_{\mathrm{d}} = 1) = u_1 \tag{5.13}$$

$$\alpha_U(U_{\mathrm{d}} = 2) = u_2 \tag{5.14}$$

$$\alpha_U(U_{\mathrm{d}} = 3) = u_3 \tag{5.15}$$

for the input node U. These settings, abbreviated by $\alpha_U = \{u_1 \;\; u_2 \;\; u_3\}$, represent the centers of the piecewise approximation. The parameters of the output node are according to equation (5.10)

$$\alpha_{f(U)} = \{-y^m_{\max} \;\; 0 \;\; y^m_{\max}\} \;\; . \tag{5.16}$$

The weights of the output node $f(U)$

$$\beta_{f(U)} = \{0 \;\; \frac{y^m_{\max}}{u_{\mathrm{sat}}} \;\; 0\} \tag{5.17}$$

contain the derivatives at different points $u_i, 1 \leq i \leq 3$.

There are only two different parameters which determine the saturation curve, the minimal and maximal output $y^m_{\max}$, and u_{sat}. For the training of the model it is assumed that all states of the input node U_{d} have the same probability. Additionally the saturation model implemented in Simulink has as single parameter the lower and upper limit, the maximal output $y^m_{\max}$ is therefore equal to u_{sat}. As a consequence the weight of the output node is clamped to

$$\beta_{f(U)} = \{0 \;\; 1 \;\; 0\} \;\; . \tag{5.18}$$

As training input 600 examples are taken that are uniformly distributed between -12 and 12. After the training the saturation is modeled with a maximal error of 0.10 and relative error of 2.27%. A comparison of the saturation curve with the prediction is given in figure 5.6.

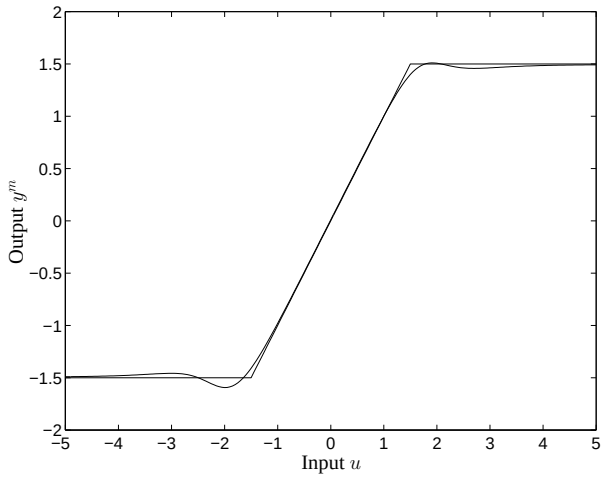


Figure 5.6: Comparison of model and real signals for saturation

Hysteresis

The hysteresis curve depends not only on the input u, but also on the sign of the first derivative. The sign of the first derivative is calculated by

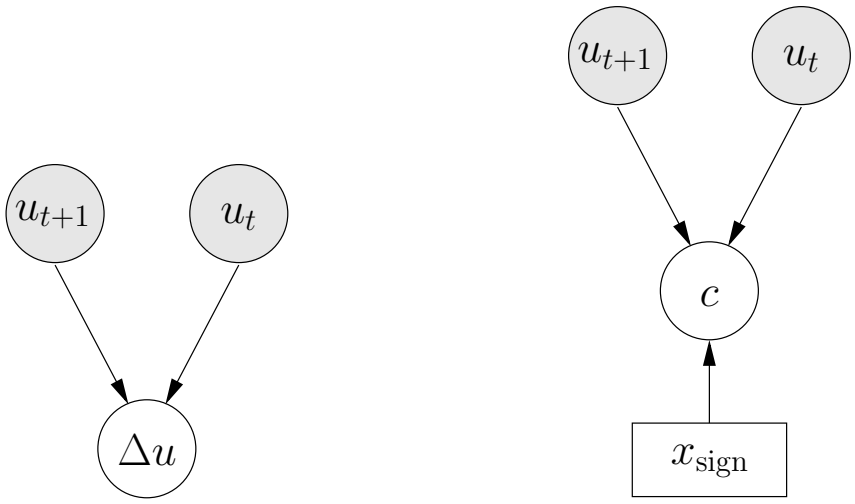
$$\text{sign}(\dot{u}) = \text{sign}(u_{t+1} - u_t) \ . \tag{5.19}$$

The difference between $u_{t+1} - u_t$ is calculated by the Bayesian network depicted in figure 5.7(a).

The problem is to detect the sign of the difference $u_{t+1} - u_t = \Delta u$. Adding a discrete parent to node ΔU does not work. The usage of different offsets $\alpha_{\Delta u}$ is not suitable to distinguish between different signs, because the mean depends on the weights and the evidence of the nodes U_{t+1} and U_t.

The trick is to use a node C_{H} with a constant evidence $c > 0$ and a discrete parent X_{sign} of node C_{H} which switches between different weights of C_{H} (confer figure 5.7(b)).

Assume that $\boldsymbol{\beta}_{C_{\text{H}}}(\text{sign}(\dot{u}) = -1) = w_\beta[-1 \ \ 1]$; i.e., the node C_{H} calculates $u_t - u_{t+1}$ given that $\text{sign}(\dot{u}) = -1$ and the weight w_β is set to one. The appropriate setting of w_β is discussed



(a) Calculation of Δu (b) Sign of Δu

Figure 5.7: Parts of the hysteresis model

later. For $\text{sign}(\dot{u}) = 1$ the weights

$$\boldsymbol{\beta}_{C_{\text{H}}}(\text{sign}(\dot{u}) = 1) = w_\beta[1 \quad -1] \tag{5.20}$$

are set the other way round. Given that the evidence for node C_{H} is always a positive constant c the Bayesian network works as follows. If $u_{t+1} > u_t$ the difference $u_{t+1} - u_t$ is positive, the difference $-u_{t+1} + u_t$ is negative. As c is positive, it is more likely that $x_{\text{sign}} = 1$.

When $u_t > u_{t+1}$, the difference $-u_{t+1} + u_t$ is positive. Thus the likelihood of c being positive is increased by $x_{\text{sign}} = -1$. Therefore the Bayesian network depicted in figure 5.7(b) is able to approximate the sign function. To be sure that the sign-function switches fast when the relationship between u_{t+1} and u_t changes, the weights w_β of the links $U_t \rightarrow C_{\text{H}}$ and $U_{t+1} \rightarrow C_{\text{H}}$, the variance of node C_{H} and the constant c have to be selected carefully. If $\Delta u = u_{t+1} - u_t$ the two distributions for $\text{sign}(\dot{u}) = \pm 1$ are

$$p(\Delta u|\text{sign}(\dot{u}) = 1) = \frac{1}{2\pi\sqrt{\gamma}} \exp -\frac{(c - \Delta u w_\beta)^2}{2\gamma} \tag{5.21}$$

$$p(\Delta u|\text{sign}(\dot{u}) = -1) = \frac{1}{2\pi\sqrt{\gamma}} \exp -\frac{(c + \Delta u w_\beta)^2}{2\gamma} \tag{5.22}$$

To be sure that the probability $P(x_{sign} = 1)$ switches fast from 0 to 1 if the difference $u_{t+1} - u_t$

changes its sign, the quotient $p(\Delta u|\text{sign}(\dot{u}) = 1)/p(\Delta u|\text{sign}(\dot{u}) = -1)$ needs to be maximal. This quotient is

$$\frac{p(\Delta u|\text{sign}(\dot{u}) = 1)}{p(\Delta u|\text{sign}(\dot{u}) = -1)} = \frac{\exp -\frac{(c-\Delta u w_\beta)^2}{2\gamma}}{\exp -\frac{(c+\Delta u w_\beta)^2}{2\gamma}} \tag{5.23}$$

$$= \exp \frac{2\Delta u w_\beta c}{\gamma} \ . \tag{5.24}$$

In order to maximize this quotient, w_β or c have to take on large values, the variance has to be very small. For our experiments $\gamma = 0.2$, $w_\beta = 4$, and $c = 10$.

After the sign is detected, it is easy to expand figure 5.7(b) so that the hysteresis curve is modeled. Only one output node has to be added. Additionally, a link between U_{t+1} and X_{sign} is added to decide whether the current input is below or beyond the threshold u_Θ (confer equation (3.58)). Thus X_{sign} has four states, two states are used to encode whether the current input exceeds the threshold u_Θ, two of them are used to represent the sign $\text{sign}(\dot{u})$.

Figure 5.9 shows that the Bayesian network depicted in figure 5.8 is able to model the hysteresis curve. Note that figure 5.9 is based on a Bayesian network with calculated weights.

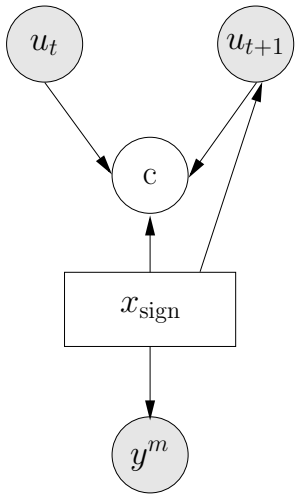


Figure 5.8: Bayesian model for hysteresis

The training of the Bayesian network is not successful, an example of a failed trial is shown in figure 5.10.

It can be seen that the threshold u_Θ is estimated incorrectly . The problem is that there is no single parameter responsible for the encoding of u_Θ. The threshold is encoded by the selection

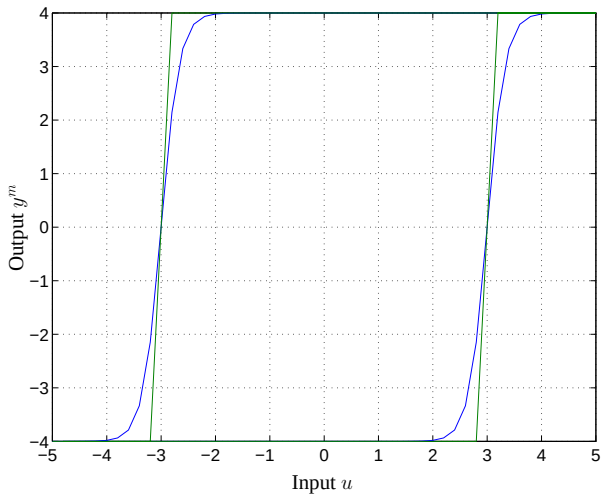


Figure 5.9: Comparison of model and real signals for hysteresis

of the four different offsets

$$\alpha_{U_{t+1}} = \{u_1 \;\; u_2 \;\; u_3 \;\; u_4\} \;\; . \tag{5.25}$$

Assuming that $u_3 < -u_\Theta < u_1 \approx u_4 < u_\Theta < u_2$ the threshold is determined by

$$-\hat{u}_\Theta = \frac{u_3 + u_1}{2} \tag{5.26}$$

and

$$\hat{u}_\Theta = \frac{u_2 + u_4}{2} \;\; . \tag{5.27}$$

As u_2 and u_3 are only determined by the mean of the training data below (beyond)the lower(upper) threshold u_Θ,and not on the hysteresis curve, the result of the training depends heavily on the selected training data.

5.2 Control of non-linear systems

In the last section the approach of piecewise linear approximation of nonlinear functions is discussed. Good results are obtained if the curve to be modeled can be divided in several straight lines which are selected according to the input u.

If the straight lines, approximating the curve, depend not solely on the input, but for instance on a hidden state or on the sign of the first derivative of the input u, this approach is no longer applicable. Therefore, a nonlinearity applied directly to the input is regarded in the next section.

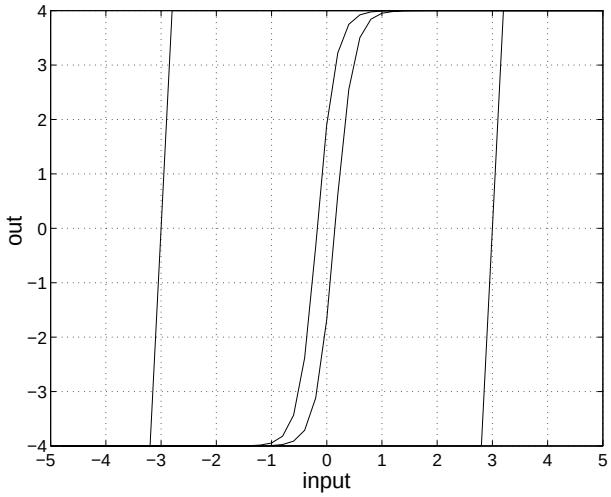


Figure 5.10: Failed trial to train the network of figure 5.8

5.2.1 Expansion of difference equation model

A frequently occurring nonlinearity is saturation. Two examples are an amplifier and a valve. If the output of an amplifier has reached the supply voltage, further increasing the input has no effect. When a valve is completely closed, a further reduction of the flow rate is not possible. Therefore a serial connection of saturation and a dynamic system of second order is discussed in this section as an example to analyze the usage of hybrid dynamic Bayesian networks.

As a starting point the difference equation model (confer equation (4.19)) is combined with the saturation model (confer figure 5.5 for the used saturation model). The resulting dynamic Bayesian network is displayed in figure 5.11, the Simulink model in figure 5.12. The difference equation model is used for first tests, because it provides more stable results in the linear case.

In comparison to the saturation model two changes are made, indicated in figure 5.13.

1. The nodes in the layer, denoted with u_{d}, have two states instead of three. In first trials it is observed that a mean input is achieved when all three states have equal probabilities, instead of one state having probability 1. Thus one of the three states is superfluous.

   To save one state it is assumed that the nodes representing the observed input, denoted by U_{c}, and the hidden output of the saturation unit U_{h} are only connected by the discrete node U_{d}. That is the link between U_{c} and U_{h} is also saved, and that the hidden node U_{h} is regarded as conditionally independent from U_{c} given U_{d}. To couple U_{c} and U_{h}, the minimal and maximal output of the saturation element is assigned as means

$$\alpha_{U_{\mathrm{c}}} = \alpha_{U_{\mathrm{h}}} = \{0 \;\; y^m_{\max}\} \tag{5.28}$$

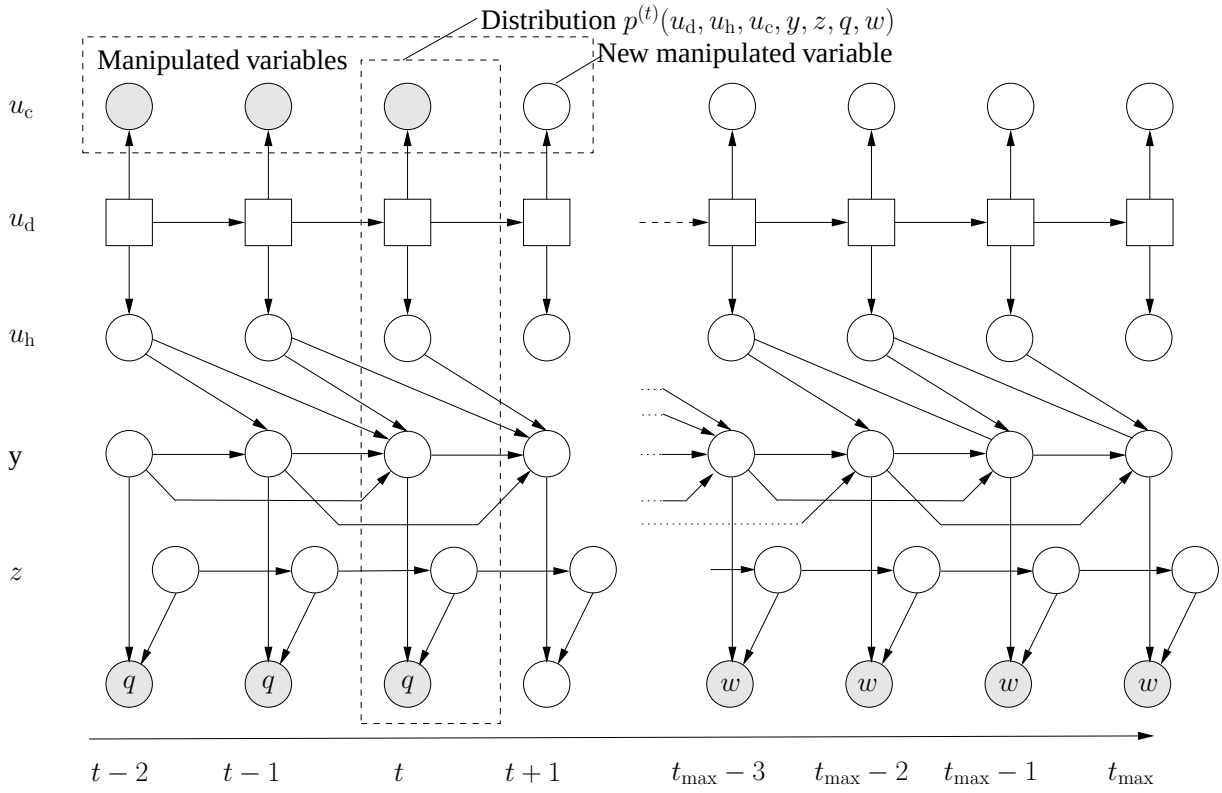


Figure 5.11: Modeling a saturation at the input

to both nodes. To obtain a tight link, the variance of U_h is set to a low value $\gamma_{U_\mathrm{h}} = 0.01$.

Intermediate outputs between 0 and y^m_max are achieved by different probabilities for U_d's states. For example, $P(u_\mathrm{d} = 2) = 0.8$ would result in an output close to $0.2 \cdot 0 + 0.8 \cdot y^m_\mathrm{max}$ if all other influences are neglected. The reduction of the number of states has also a positive influence on the runtime which is proportional to 2^{t_max} instead of 3^{t_max}.

2. A link $U_{\mathrm{d},t} \rightarrow U_{\mathrm{d},t+1}$ is added. In the control of linear systems any a-priori knowledge for the input nodes is avoided, by setting the variance γ_U to a maximal value. If there were no links $U_{\mathrm{d},t} \rightarrow U_{\mathrm{d},t+1}$, the probability distribution of $U_{\mathrm{d},t}$ would have a severe influence on the calculation of the manipulated value. By adding a link $U_{\mathrm{d},t} \rightarrow U_{\mathrm{d},t+1}$ this influence is reduced, as the conditional probability of the states of $U_{\mathrm{d},t+1}$ depends on the probability of the previous time-slice. Therefore there are more time-slices used to estimate the desired distribution.

In a first step a controller is used whose weights are calculated analytically. The result of a controller with 6 nodes used for the representation of the past and 3 nodes for the future is

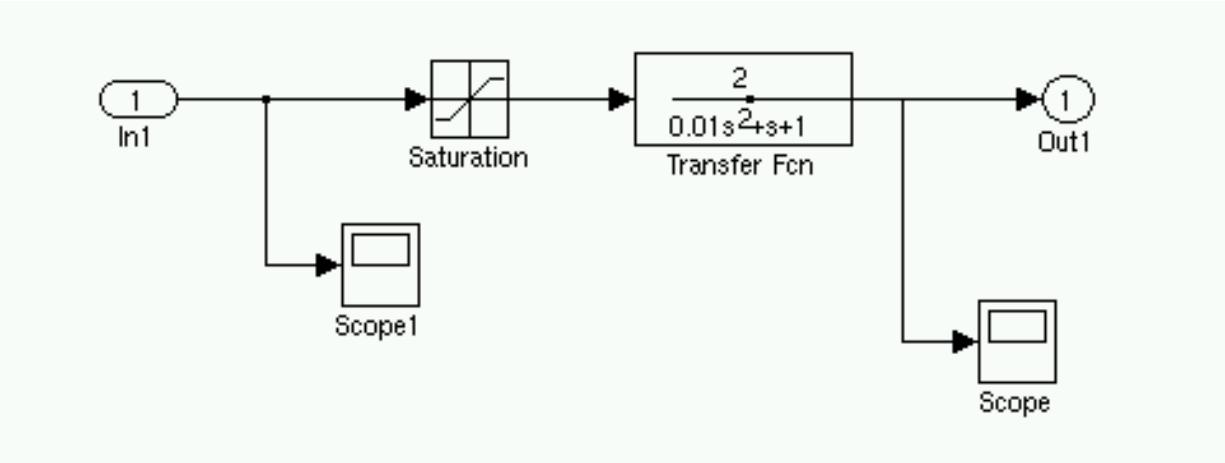


Figure 5.12: Simulink model of test system

displayed in figure 5.14.

Table 5.1: Results of experiments with expansion of difference equation model

Experiment number	1
$I_{\mathrm{d}}(z^d = 0)$	20.11
$e_\infty(z^d = 0)$	-0.02
Overshoot	0.26
$I_{\mathrm{d}}(z^d = 1)$	0.18
$e_\infty(z^d = 1)$	-0.05
$t_s(z^d = 0, 1\%)$	2.65
$t_s(z^d = 0, 3\%)$	0.85
$t_s(z^d = 1, 1\%)$	1.20
$t_s(z^d = 1, 3\%)$	0.3
Evaltime	32.11

At the beginning of the test shown in figure 5.14 ($t < 2.9\,\mathrm{s}$), the desired value w is equal to zero. The observed output q after convergence q is 1.26, that is the steady state error e_∞ for a desired value of $w = 0$ is $e_\infty = 1.26$. The reason is that the distribution of $U_{\mathrm{d},t}$ does not only depend on U_{c}, which represents the required control signal, but also on previous probabilities of $U_{\mathrm{d},t-1}$. During design, some constants, e.g. the means of U_{c} and U_{h}, and the probability distribution of $U_{\mathrm{d},1}$, are selected so that the controller shows the best performance for $w = 10$.

At $t = 2.95\,\mathrm{s}$, the desired value is changed to $w = 10$. Figure 5.14 shows that the controller operates as intended in this case. After a short time the desired value is reached nearly without deviation (compare table 5.1). The maximal input of the manipulated variable is approximately 8.45 which is lower than the maximal input used for the linear dynamic system with the same dynamic. But it is also lower than the maximal output of the saturation; i.e., the settling time t_s

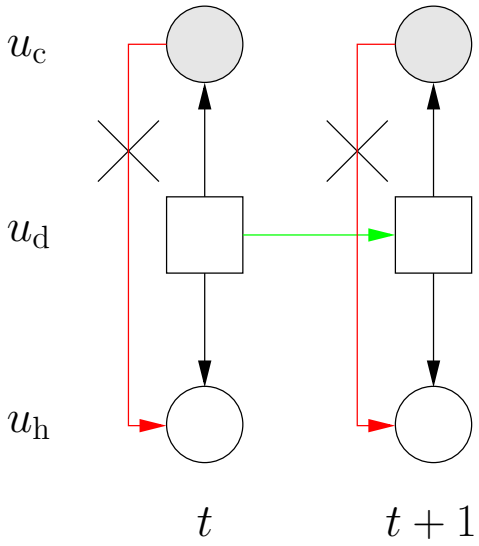


Figure 5.13: Changes of links at the input

is longer than necessary. It could be shortened by a larger input.

The overshoot of the system is larger than for the linear system (compare table 4.3, system 1). The reason are the links $U_{\mathrm{d},t-1} \rightarrow U_{\mathrm{d},t}$ which increase the sluggishness, but are necessary to decrease the steady state error.

At $t = 13.75\,\mathrm{s}$ the disturbance z^d is changed to 1, once again the controller reacts as intended; i.e., after $0.3\,\mathrm{s}$ the largest part of the disturbance is eliminated and the steady state error is nearly zero.

The training of the model is more complicated than in the linear case. In the linear case it is assumed that $z^d = 0$ during training, thus no hidden nodes are left. In the nonlinear case the evidence of the nodes $U_{\mathrm{h},t}$ is not given. The missing evidence for nodes $U_{\mathrm{h},t}$ leads to problems to adapt the weight of the nodes representing the model output Y^m. Training with a fixed variance $\gamma_{U_\mathrm{h}} = \{0.01 \;\; 0.01\}$ fails, the controller shows no reaction when the desired value is changed.

Figures 5.15(a) and 5.15(b) shows the result of two trials to train the controller with the variance of the hidden input node $\gamma_{U_\mathrm{h}} = \{16 \;\; 16\}$ set to 16. For both trials the same set of parameters are used.

At the beginning the desired value is set to zero. In both figures a large steady state error is observed (compare figure 5.15(a) for $t < 6\,\mathrm{s}$ and figure 5.15(b) for $t < 9\,\mathrm{s}$). Afterwards the desired value is set to 10. In figure 5.15(a) the desired value is reached with low deviation, figure 5.15(b) shows a complete failure of training. A comparison of these two figures shows that the training of the difference equation model is unstable. The most probable reason is that

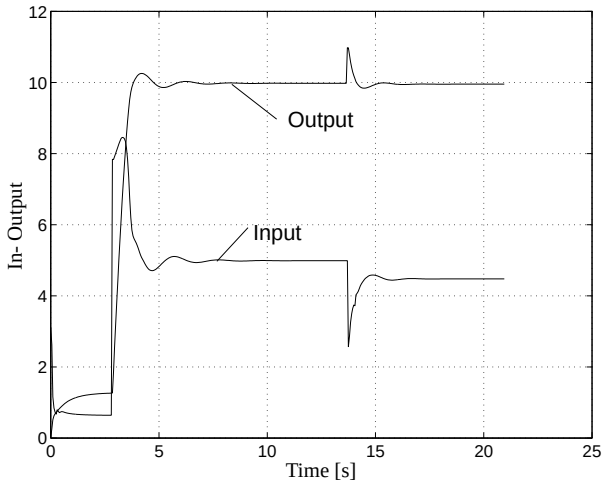


Figure 5.14: Reference and disturbance reaction of saturation model

the intended relation between Uc and U_{h} is not learnt correctly despite a-priori knowledge about saturation.

The next section shows that the problems discussed in this section are overcome with an extension of the state-space model.

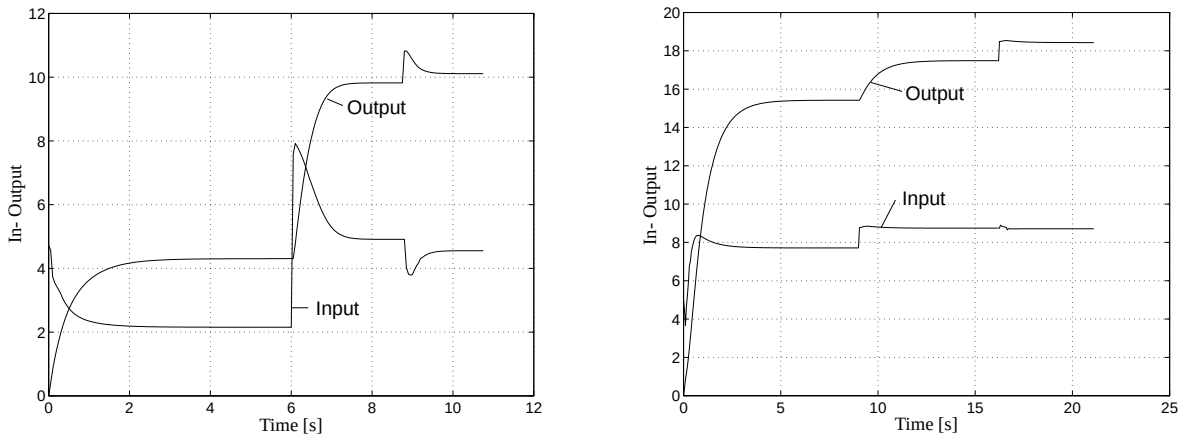
5.2.2 State-space model

The model introduced in the last section shows unsatisfactory training results. The problem consists of learning the relationship between the input u_{c} and u_h. The latter is the estimated output after saturation has taken place. Additionally, the performance depends on the selected desired value w.

In this section it is tested how this problem is overcome with a state-space model with three different operating points. The operating points are triggered by the input u. As in the piecewise approximation, discussed at the beginning of this chapter, different inputs u result in different probabilities for the states u_{d}. The states of U_{d} switch between different means and weights of the state nodes. Switching between different weights means that the input matrix is changed depending on the input, whereas the transfer matrix is not changed.

The first operating point represents the lower saturation. The weight between U_{d} and $\boldsymbol{X}^s$ in that case is 0, the mean is equal to the product of the input vector $\boldsymbol{b}_{\mathrm{BN}}$ and the lower saturation level. In our experiments the lower saturation was set to zero, not to $-y^m_{\max}$ as indicated by equation (3.48), so that the mean is also zero in that case.

The second point represents the usual operation mode, the mean of $\boldsymbol{X}^s$ is zero as in the linear



(a) Better result

(b) Complete failure

Figure 5.15: Signals of trained difference equation model

case, the weight is equal to $\boldsymbol{b}_{\mathrm{BN}}$.

The third state of U_{d} represents the upper saturation level. As in the lower saturation the input has no effect on the state; i.e. the weight between U_{d} and $\boldsymbol{X}^s$ is zero. The mean results from the product $\boldsymbol{b}_{\mathrm{BN}} y^m_{\max}$ of the input vector $\boldsymbol{b}_{\mathrm{BN}}$ and the upper output $y^m_{\max}$. This discussion results in the means of the input node U

$$\alpha_U = \{u_1 \;\; u_2 \;\; u_3\} \tag{5.29}$$

where the operating points are sorted according to unequation $u_1 < 0 < u_2 < u_{\mathrm{sat}} < u_3$. The means and weights of the state nodes are

$$\alpha_{\boldsymbol{X}^s} = \{0 \;\; 0 \;\; \boldsymbol{b}_{\mathrm{BN}} y^m_{\max}\} \tag{5.30}$$

$$\beta_{\boldsymbol{X}^s} = \{0 \;\; \boldsymbol{b}_{\mathrm{BN}} \;\; 0\} \;. \tag{5.31}$$

The remaining parameters of the disturbance layer, designated with z^d, and the output layer, with the nodes for the observed output q and the desired value w, remain unchanged.

For the training α_U is clamped; i.e., the intended operating points are preselected, so that only the dynamic of the system has to be learnt. After training with 40 data sets and 3*5 iterations the results listed in table 5.2 are obtained.

In contrast to the experiments discussed in chapter 4, the number of nodes for the representation of the past is reduced to 5, the nodes used for the representation of the future are restricted

Table 5.2: Results of experiments with state space

Experiment number	1	2	3	mean
$I_{\mathrm{d}}(z^d = 0)$	21.6964	20.7516	21.3030	21.2504
$e_\infty(z^d = 0)$	0.0691	-0.0426	0.0449	0.0522
Overshoot	0.0894	0.0013	0.1510	0.0806
$I_{\mathrm{d}}(z^d = 1)$	0.1128	0.1105	0.1105	0.1113
$e_\infty(z^d = 1)$	0.0973	-0.0127	0.0843	0.0647
$t_s(z^d = 0, 1\%)$	1.3000	1.0500	1.2500	1.2000
$t_s(z^d = 0, 3\%)$	0.7000	0.7000	0.7000	0.7000
$t_s(z^d = 1, 1\%)$	3.2500	0.3000	0.7000	1.4167
$t_s(z^d = 1, 3\%)$	0.2000	0.2000	0.2000	0.2000
Traintime	32827	29251	29345	30474
Evaltime	37	38	45	40

to 2. According to the discussion in chapter 8, this has only a slight impact on the result. But the reader should keep in mind that the experiments in chapter 8 are executed with a difference equation model.

Table 5.2 shows that good results are obtained in all cases. Convergence is achieved in all experiments, and the steady state error e_∞ is below 1% of the desired value.

The squared error is larger than the squared error of system 1 that has the same dynamic as the test-system in this section, but no saturation at the input (For a description of system 1 see table 4.1, for the results obtained with a linear state-space system see table 4.4). The larger squared error is due to the the connection of the discrete input nodes $U_{\mathrm{d},t}$ which is again necessary to reduce the influence of the a-priori knowledge, but cuts the input peak (confer figure 5.17). The settling time has increased, particularly for the reference reaction, which is partially due to saturation. The input exceeds the upper threshold for more than half a second.

For the disturbance reaction, neither the lower nor the upper threshold are reached. Thus, the longer settling time indicated by table 5.2 in comparison to table 4.4 is due to the Bayesian controller. The main problem is the run-time. The evaluation time of 40 s exceeds the threshold for real-time operation. One reason for the large run-time is that the experiments in this section are done with the inference algorithm introduced by [LJ99] which is slower than the older one [Lau92]. For a comparison of runtime see chapter 8. But the usage is essential as the former inference algorithms lacks numerical stability.

The second reason is that the run-time increases exponentially with the number of time-slices if exact inference is required.

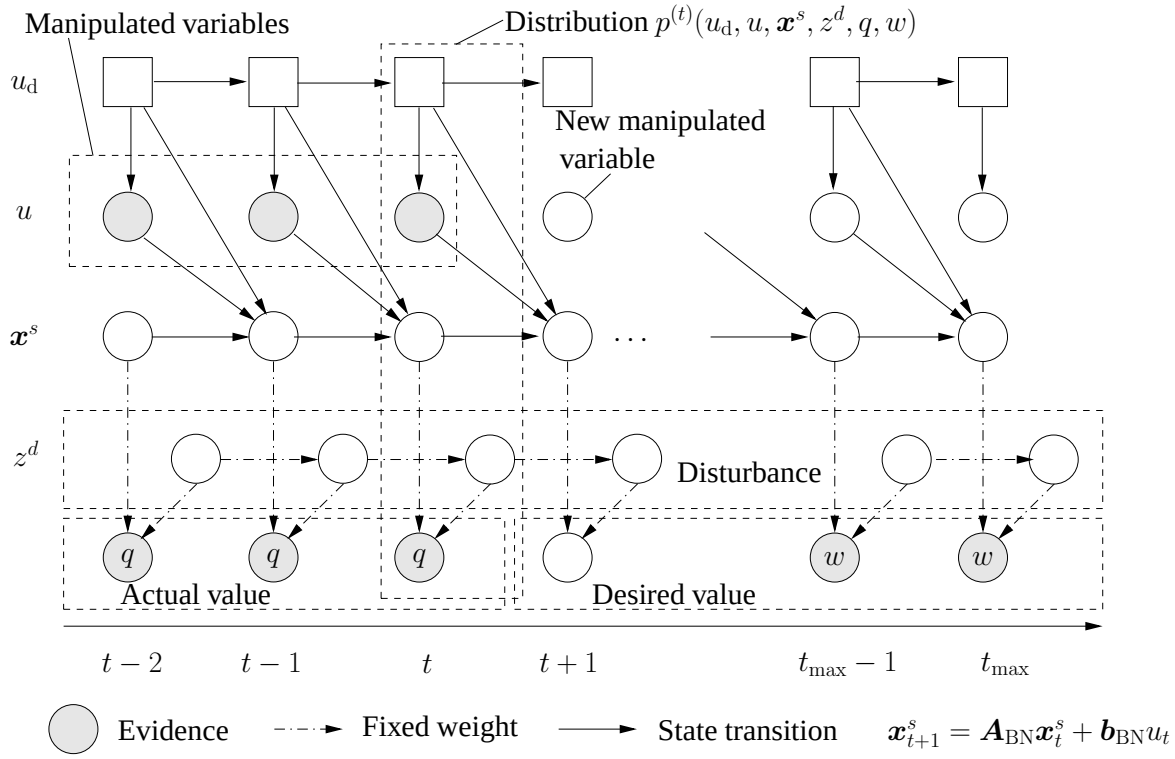


Figure 5.16: State space approach

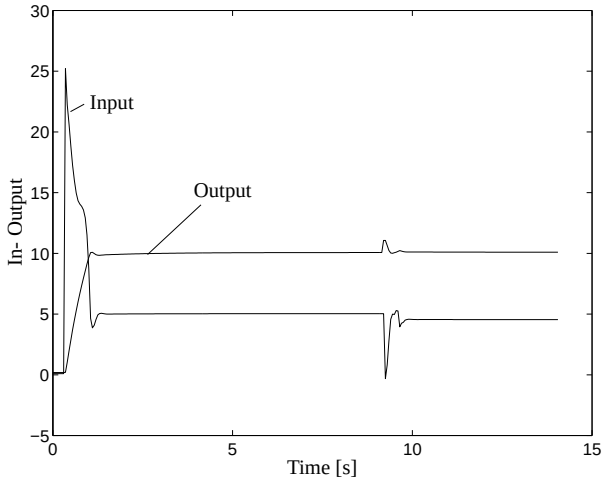


Figure 5.17: Signals of state space approach

Chapter 6

Modeled manufacturing processes

The experiments in chapters 4 and 5 are done with data obtained by simulations with Simulink. The advantages of simulations are that a large amount of data can be collected in a controlled environment. This is of importance particularly to test the approach under circumstances which are rarely observed in reality.

In this chapter modeling with Bayesian networks is applied to process data, collected by cooperating institutes within the frame of SFB 396 (Collaborative Research Center, number 396) “Robust shortened process sequences for lightweight sheet parts”.

The first process, hydroforming, is divided into the steps preforming, hydrocalibration, welding and trimming. The second modeled process is injection moulding which consists of the steps preheating, handling, and injection of the plastic.

6.1 Hydroforming

6.1.1 Preforming and calibration

Preforming

During hydroforming tubes or blanks are formed by high internal pressure. At the chair of manufacturing technology two blanks are formed at the same time. In a first step, both blanks are pressed on top of each other at the flange by different clamping forces applied by a hydroform press (see table 6.1 for a list of the parameters, measured during preforming). During the hydroforming process, fluid is pressed between the two blanks. As a result, the pressure between the blanks increases and the blanks are formed into a tool. Figures 6.1 and 6.2 show the dependency of the pressure on the volume of the hydroforming fluid between the blanks. Both figures show

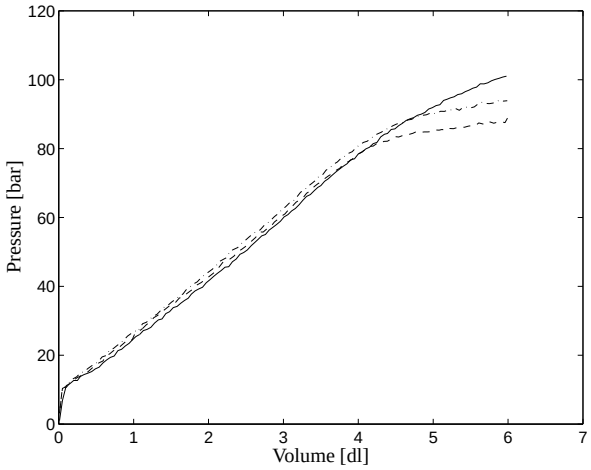


Figure 6.1: Preforming with a force of 200 kN

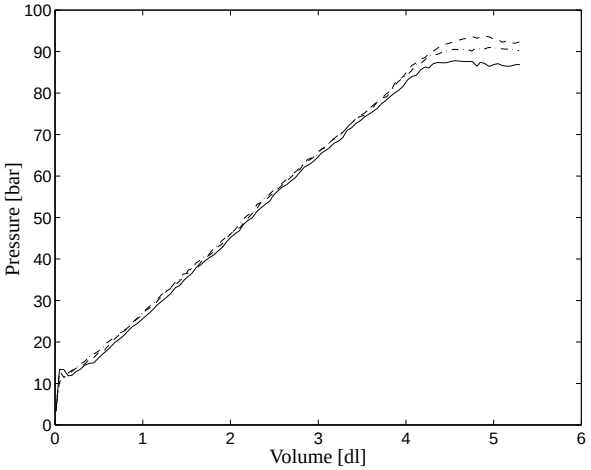


Figure 6.2: Preforming with a force of 300 kN

Table 6.1: Parameters for preforming

Variable	Min	Max	Unit	Remarks
Force	200	500	kN	Increased in steps of 100 kN.
Volume	0.1	7.01	dl	Volume is increased, until first leaks occur.
Pressure	1.04	137.35	bar	

the results of three experiments with equal clamping forces. With increasing pressure more and more leaks occur, resulting in a smaller slope of the pressure. The curve progression depends on the clamping force. Small clamping forces lead to a better flow of material into the die, but the process is stopped at a lower pressure due to occurring leaks. When less material is drawn into the form, this might lead to failures during calibration, caused by lower tension. There are two possible steps after preforming. The first possibility is that preforming is directly followed by hydrocalibration. That means that clamping forces are increased to a maximal value, so that the leaks are sealed. Afterwards, more hydroforming fluid is pressed between the blanks to form the edges. The second possibility is that a welding process takes place to seal the leaks observed at the end of preforming. In the next paragraph calibration is discussed. The welding process is discussed later in section 6.1.3.

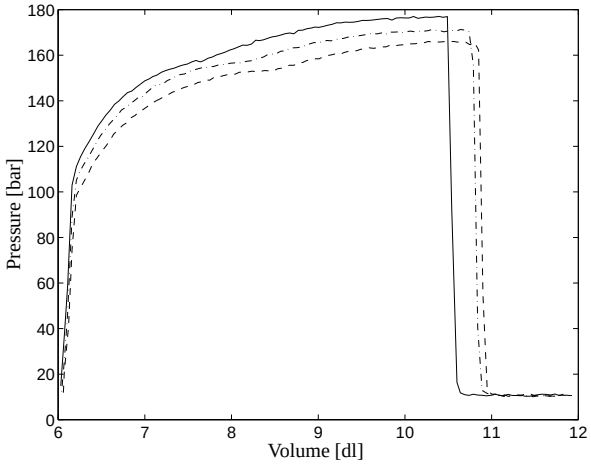


Figure 6.3: Calibration, preforming done with a force of 200 kN

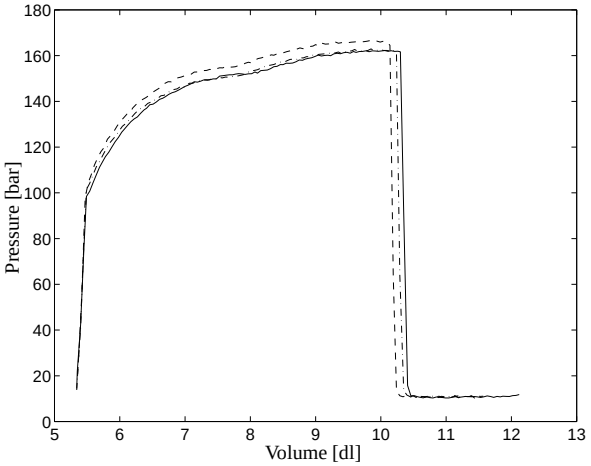


Figure 6.4: Calibration, preforming done with a force of 300 kN

Table 6.2: Parameters for calibration

Variable	Min	Max	Unit	Remarks
Force	200	500	kN	During calibration the force is increased to a maximum. But the force used during preforming has an influence on the calibration process, so the preforming forces are used as an additional parameter for calibration.
Volume	5.33	12.12	dl	Volume is increased, until the blanks burst. After bursting the pressure drops nearly to zero.
Pressure	1.212	203.38	bar	

Calibration

The data to model hydrocalibration (confer table 6.2) are collected at an early phase of the “Collaborative Research Center”. That means calibration follows directly after preforming. In the current version calibration is followed by laser beam welding.

No more fluid is pressed between the blanks after preforming is finished. When the press is prepared to finish hydrocalibration in one step, the pressure drops nearly to zero, but the hydroforming fluid remains between the blanks. Thus the initial volume for hydrocalibration is not zero, but equal to the volume at the end of preforming (confer figures 6.3 and 6.4). A steep increase of pressure is observed first, until the pressure at the end of preforming is reached. Then the slope changes, increasing the volume has a smaller effect on the pressure. The most important

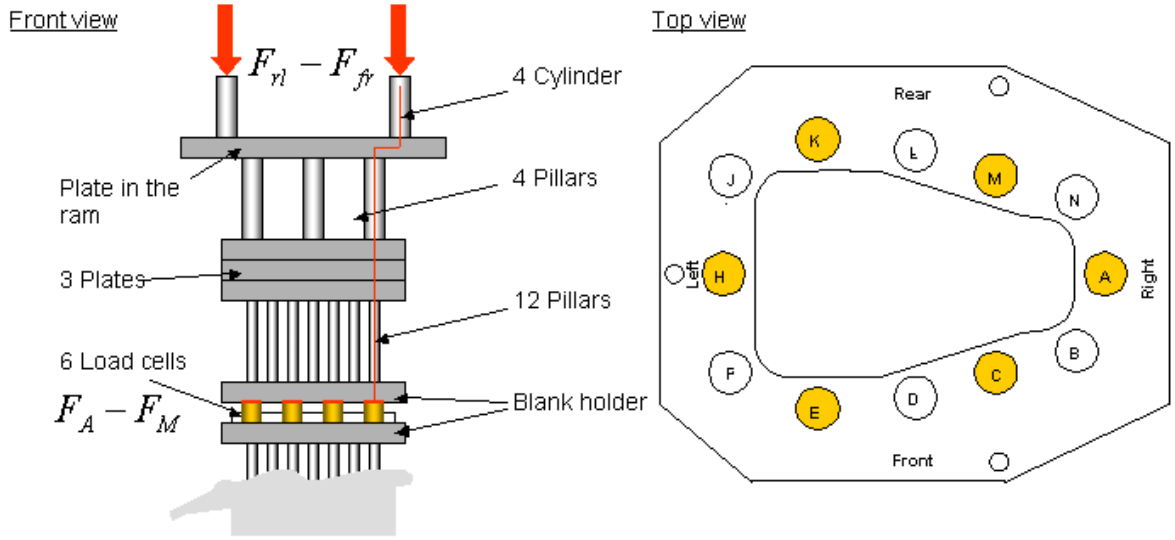


Figure 6.5: Picture of the hydroforming tool, source LFT

point, which is of course avoided in production, is the bursting of the blanks. The bursting point in figures 6.3 and 6.4 is situated between $V = 10$ dl and $V = 11$ dl. Our data are collected during material tests, so that this effect is included in our data. That means that the function to be modeled is not only non-linear, but has even a non-continuous point. The exact prediction of the bursting-point would be helpful to increase the volume during calibration as much as possible without risking to loose the component. A comparison between figure 6.3 and 6.4 shows that the clamping force has an influence on the curve. Thus different models are trained for different clamping forces used during preforming.

Modeling only one global clamping force is a simplification as four different clamping forces effect at different points. It is the aim that the clamping forces at the flange are equal, so that the flow of material is independent of the position. This point is discussed in the next section.

6.1.2 Modeling the forces of the press

Figure 6.5 shows the construction of the hydroforming tool. At the top of the tool the forces of four different cylinders have an effect on the plate in the ram. The force of each of these cylinders is controlled individually. The cylinders are situated at the rear, left and right hand side, and at the front at the left and right hand side. The forces at the cylinders are therefore denoted by $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}$, and F_{fr}. From the plate on the top the forces act on four different pillars mounted on three plates. Via twelve pillars the original forces affect the blank holder. The positions of

Table 6.3: Parameters to model the forces

Variable	Min	Max	Unit	Remarks
Forces at cylinder $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}, F_{\mathrm{fr}}$	50	400	kN	There are four different cylinders which provide the necessary forces to press the blanks on each other. Two of them are placed at the rear, left and right. The other two are placed at the front.
Forces at the blank holder, $F_{\mathrm{A}}, F_{\mathrm{C}}, \cdots, F_{\mathrm{M}}$	16.9	292.30	kN	It is usually not possible to calculate the standard deviation, as the experiments are not repeated in most of the cases. The only exception is given in table 6.4. The six places of measurement are given in figure 6.5.

the twelve pillars A - N is described at the right hand side of figure 6.5. It is the aim to get equal forces $F_{\mathrm{A}}, F_{\mathrm{C}}, \cdots, F_{\mathrm{M}}$ at positions A - M to guarantee an optimal flow of material. A Bayesian controller requires therefore knowledge about the dependency between the forces adjusted at the top of the hydroforming press and the forces measured at the positions A, C, E, H, K, and M.

For the exploration of the dependency, 69 tests with 37 different settings are executed at the chair of manufacturing technology. One experiment ($F_{\mathrm{rl}} = F_{\mathrm{rr}} = F_{\mathrm{fl}} = F_{\mathrm{fr}} = 225\ \mathrm{kN}$) is repeated six times and is used to examine the dispersion of the experiments. Table 6.3 gives a coarse overview about the test data used for examination, table 6.4 shows the dispersion of one test.

For most of the data the results are reproducible with high accuracy. A numerical calculation makes no sense in most of the cases, as usually the experiments are only repeated twice.

An analytical model, a neural network and a Bayesian network are trained using the 69 data sets . Now predictions are figured out for 14, yet un-presented, settings and compared to reality. The results are discussed in section 7.1.2 (compare also [WBS$^+$01]).

6.1.3 Laser beam welding

After preforming and cutting, welding takes place. Two types of joints are examined, the lap seam joint and the lap edge joint. Several adjustments of the parameters in tables 6.5 and 6.6 are tested to get an impression of optimal process parameters.

The laser beam is driven with a velocity v around the component. At the end a small part is

Table 6.4: Mean and Dispersion for the experiment with $F_{\mathrm{rl}} = F_{\mathrm{rr}} = F_{\mathrm{fl}} = F_{\mathrm{fr}} = 225\ \mathrm{kN}$

Variable/Position	Standard deviation	Mean
F_{A}	2.7 kN	155.9 kN
F_{C}	11.4 kN	141.1 kN
F_{E}	0.4 kN	124.0 kN
F_{H}	5.7 kN	171.3 kN
F_{K}	3.6 kN	101.9 kN
F_{M}	0.7 kN	150.3 kN

Table 6.5: Continuous Parameters for laser-beam welding

Variable	Min	Max	Unit	Standard deviation	Remarks
Defocussing	-6	6	mm	—	Tested in steps of 2mm, when the laser is out of focus for less then -3 mm, the quality of the joint is decreasing.
Offset	0	6	mm	—	
Velocity	2	7	$\mathrm{m/min}$	—	
Tensile force	0	5370	N	1460.1	Output variable, representing the quality of the weld.

welded a second time to be sure that the component is tight after welding.

During the welding process the laser might be out of focus, in both the x or z-axis. This is described by the offset or defocussing of the laser beam (see figure 6.6). At the edges of the component it is difficult to guarantee that the angle between the x-axis and the laser beam (lap edge joint) is 0° (For the lap seam joint the angle between the z-axis and the laser beam is regarded). The tensile force is the measured output of the process and should be maximized.

When looking at the data, three parts can be distinguished. In the main part the number of joints is kept constant, defocussing and the offset are set to zero. In this part of the data the setting angle, the joint type and the velocity are changed. The results of these experiments are shown in table 6.7 and figure 6.7. The result is that a velocity of approximately 4 $\mathrm{m/min}$ is nearly optimal for the lap edge joint.

In the second part the influence of the velocity and the number of joints on the quality of the joint is tested. The rest of the parameters is not changed. Table 6.8 shows the mean of 6 experiments per line. It indicates that a second joint results in a large difference of the tensile

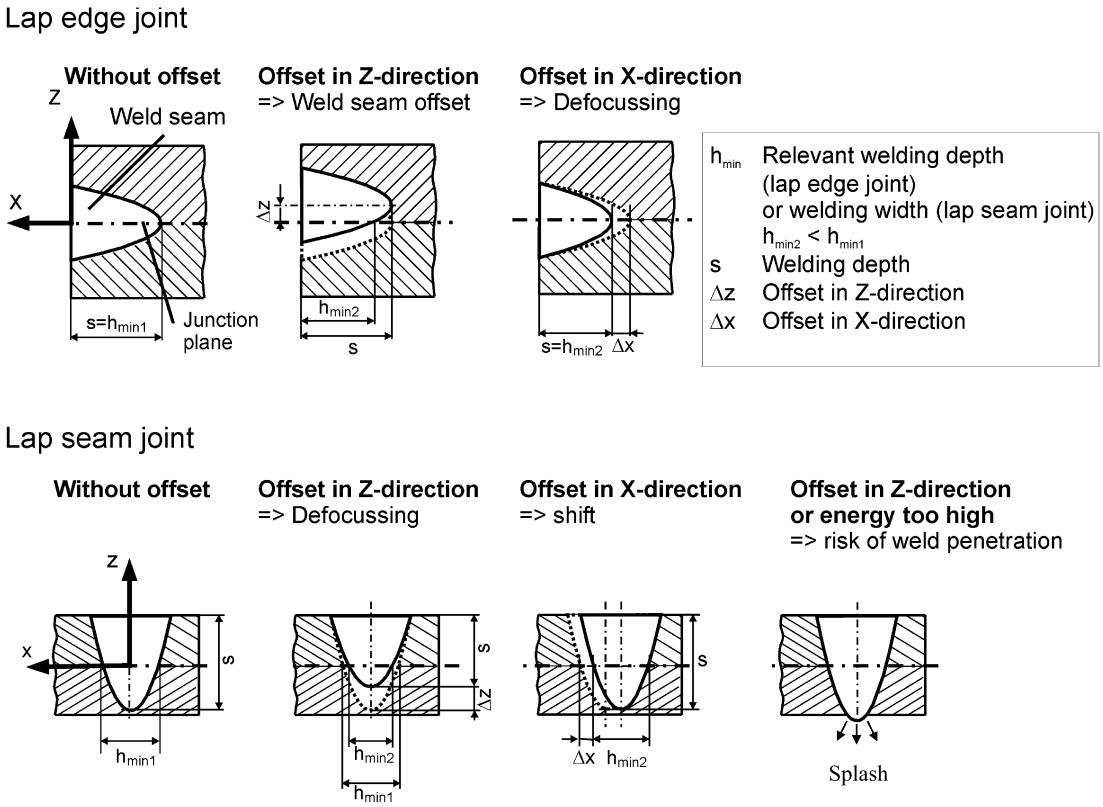


Figure 6.6: Difference between lap edge and lap seam joint, picture taken from [Kre02]

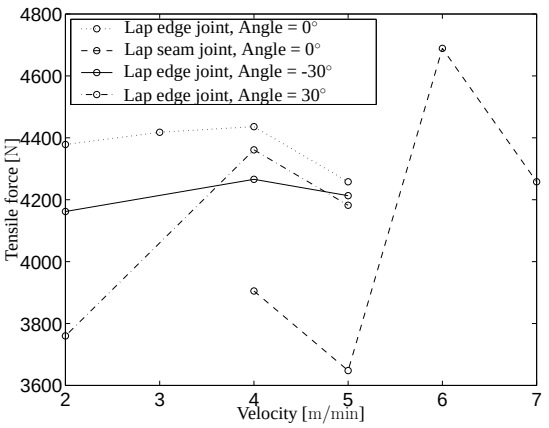


Figure 6.7: Dependency of tensile force F on velocity v

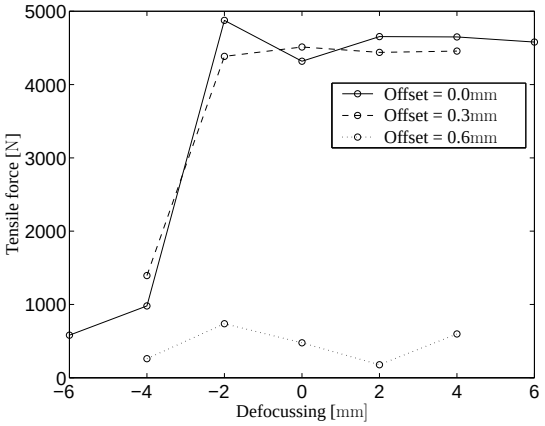


Figure 6.8: Tensile force depending on offset and defocussing

Figure 6.9: Component produced by injection moulding

force for a velocity of 4 $\mathrm{m/min}$.

The third part explores the influence of the offset and defocussing. Figure 6.8 indicates that there are two thresholds for the offset and defocussing. If these thresholds are exceeded, the quality of the weld decreases. According to figure 6.8 the threshold for the offset is somewhere between 0.3 and 0.6 mm, the threshold for the admitted defocussing between -4 and -2 mm.

6.2 Injection moulding

At the experiments, carried out at the Institute of Polymer Technology, composite components, consisting of a blank and plastic are insert moulded. The resulting component that is depicted in figure 6.9 has the form of a T.

The plastic is injection-moulded around a blank, having a geometry of 249.3mm $\times$ 73.7mm $\times$ 1.0 mm [EZ98]. At the beginning of the production process the cleaned blank is preheated. Preheating is done first by an infrared heater, to shorten the process cycle, and afterwards by a convection oven. After preheating the blank is carried automatically into the cavity. There is no

Table 6.6: Discrete Parameters for laser-beam welding

Variable	Possible Values	Remarks
Type of joint	{Lap edge joint, lap seam joint}	
Number of weldings	{1 2}	The effect on the force is only tested for different velocities.
Setting angle	{-30° 0° +30°}	

Table 6.7: Force depending on type of weld, angle and velocity

Type of weld	Angle	Velocity $[\mathrm{m/min}]$	Tensile force F [N]	σ_F
Lap edge joint	$0°$	2	4378	180
Lap edge joint	$0°$	3	4418	185
Lap edge joint	$0°$	4	4436	111
Lap edge joint	$0°$	5	4258	156
Lap edge joint	$-30°$	2	4162	249
Lap edge joint	$-30°$	4	4266	280
Lap edge joint	$-30°$	5	4213	192
Lap edge joint	$+30°$	2	3760	466
Lap edge joint	$+30°$	4	4361	168
Lap edge joint	$+30°$	5	4182	95
Lap seam joint	$0°$	4	3905	180
Lap seam joint	$0°$	5	3648	185
Lap seam joint	$0°$	6	4689	111
Lap seam joint	$0°$	7	4258	156

Table 6.8: Influence of the number of joints

Velocity	Tensile force for one joint [N]	σ_F	Tensile force for two joints [N]	σ_F
3	4380	104	4451	116
4	4438	120	4657	243
5	4362	135	4379	234

control whether the blank has reached the desired preheating temperature τ_p. During transport the blank cools off. This cooling process is not modeled. It is assumed that this factor is constant due to the automatic transport.

Also the cavity is preheated, its temperature is denoted by τ_c. The melted plastic, whose temperature is denoted by τ_m, is injected by high pressure and a constant velocity $v = 10\,\mathrm{mm/s}$ into the cavity. The curve of the pressure is depicted in figure 6.10. After injection the pressure is kept constant for a short time to reduce the warpage. This pressure is called holding pressure P_h. Note that an index “h” is used to distinguish the holding pressure P_h from the probability P. A list of the input parameters is given in table 6.9.

It is the aim of the production process that the resulting composite has a minimal warpage and that the take-off tension F, to separate the plastic from the blank, is maximal. To test the impact of the input parameters on the quality parameters, listed in table 6.10, each input pa-

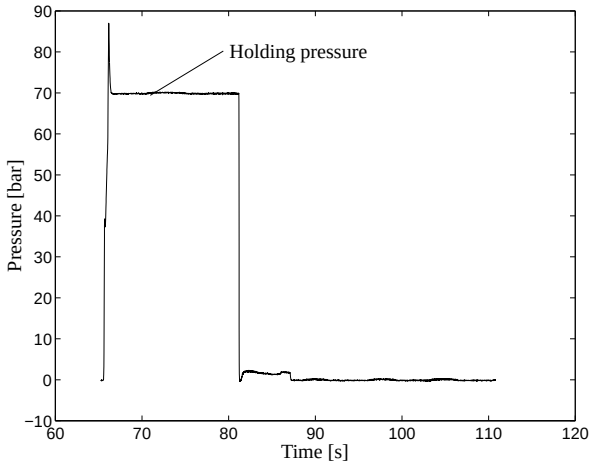


Figure 6.10: Pressure used for injection

rameter is tested at five different levels. As this would result in 5^4 experiments, a statistical test plan, discussed in [EZ98], is employed. In 2^4 settings (outer experiments) all combinations of $\tau_{\mathrm{p}} \in \{80, 260\}^{\circ}C, \tau_{\mathrm{m}} \in \{260, 280\}^{\circ}C, \tau_{\mathrm{c}} \in \{80, 120\}^{\circ}C$, and $P_{\mathrm{h}} \in \{30, 70\}\mathrm{bar}$ are tested. The next 16 experiment (inner experiments) are carried out with all combinations of $\tau_{\mathrm{p}} \in \{125, 215\}^{\circ}C, \tau_{\mathrm{m}} \in \{265, 275\}^{\circ}C, \tau_{\mathrm{c}} \in \{90, 110\}^{\circ}C$, and $P_{\mathrm{h}} \in \{40, 60\}\mathrm{bar}$. As last experiment the result for the central point $\tau_{\mathrm{p}} = 175^{\circ}C, \tau_{\mathrm{m}} = 270^{\circ}C, \tau_{\mathrm{c}} = 100^{\circ}C$, and $P_{\mathrm{h}} = 50$ bar is measured. According to the results in [EZ98] the preheating temperature and the holding pressure have a large impact on the warpage. The take-off tension F is mostly influenced by the preheating temperature τ_{p}, the melt temperature τ_{m} and the holding pressure P_{h}. These results coincide with the first Bayesian models, discussed in the report of project part C1 [KN98] for the years 96 - 98. Beside the input values, additional parameters are measured. Of course these values are strongly correlated to the input values listed in table 6.9. So all of them have an influence on the quality parameters. A selection, according to the correlation between the selected parameter and the take-off tension, is made to reduce the number of models to be analyzed. The selected values, together with a short explanation, are listed in table 6.11.

In the first period of the Collaborative Research Center only the input values are used in the model. In section 7.2 it is discussed whether the additional measurements listed in table 6.11 are suited to improve the prediction of the take-off tension.

Table 6.9: Input parameters for injection moulding

Variable	Min	Max	Unit	Remarks
Preheating temperature τ_{p}	80	260	$^{\circ}\mathrm{C}$	Temperature of the metal. During handling the metal is moved from preheating to the tool, and thus cooling takes place. Five different settings $\tau_{\mathrm{p}} \in \{80, 125, 170, 215, 260\}$ are tested.
Melt temperature τ_{m}	260	280	$^{\circ}\mathrm{C}$	Five different settings are tested for the temperature of plastic $\tau_{\mathrm{m}} \in \{260, 265, 270, 275, 280\}$.
Temperature of cavity τ_{c}	80	120	$^{\circ}\mathrm{C}$	Five different settings $\tau_{\mathrm{c}} \in \{80, 90, 100, 110, 120\}$.
Holding pressure P_{h}	30	70	bar	Pressure $P_{\mathrm{h}} \in \{30, 40, 50, 60, 70\}$ which is used for the injection of the plastic.
Velocity v	10	10	$\mathrm{mm/s}$	The velocity is kept constant for all experiments.

Table 6.10: Quality parameters for injection moulding

Variable	Min	Max	Unit	Remarks
Take-off tension F	3130	5920	N	Force needed to divide the metal from the plastic.
Warpage	-0.04	0.68	mm	Warpage of the product.

Table 6.11: Additional measurements for injection moulding

Variable	Min	Max	Unit	Remarks
Cushion C	0.4	19.1	mm	Length proportional to the amount of plastic, which is not pressed in the tool.
Plasticizing stroke s_{pl}	67.4	71.0	mm	
Maximal cavity temperature $\tau_{\mathrm{c,max}}$	88	132	$^{\circ}\mathrm{C}$	Maximal temperature of the cavity.
Work for injection W_{inj}	1960	3380	Nm	Energy needed for filling the form.

Chapter 7

Process models

This chapter discusses the developed models for the manufacturing processes, discussed in chapter 6. Most of the models use both discrete and continuous nodes to model nonlinearities. The applied technique is discussed in section 5.1. Only the distribution of forces is modeled by a pure linear model.

To judge the quality of the developed model the relative error

$$e_{\mathrm{r}} = \frac{\mathrm{abs}(y_{\mathrm{p}} - y)}{\mathrm{abs}(y)} \quad , \tag{7.1}$$

is used, i.e. the deviation of the predicted value y_{p} from the actual value y is divided by the actual value y. Equation (7.1) is only applied for continuous random variables y.

For discrete random variables, either the percentage of misclassifications is given or continuous values are assigned to each discrete value. In the latter case equation (7.1) is used again. The selected possibility is listed together with the model. If possible, the obtained relative error e_{r} is compared to the dispersion of the data to estimate whether the error is due to scattering in the data or caused by the model.

To assess a model, it is also necessary to test its ability to make predictions for yet unpresented examples. Thus, the model is usually trained with 90% of the data, the relative error e_{r} is calculated based on the predictions for the remaining 10% of the data. This procedure is employed to judge the models for preforming, calibration, and for the distribution of forces.

The data of the welding process can be divided into 48 blocks with equal input data within the block. Here, the relative error is measured by training with 47 blocks and figuring out predictions for the remaining block. Using this mechanism, it is guaranteed that the Bayesian network has never seen the example before.

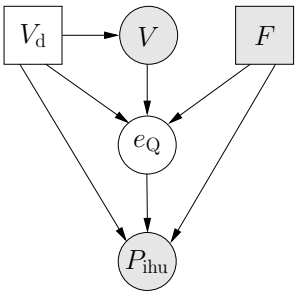


Figure 7.1: Model for preforming

Models are developed for two manufacturing processes. First, hydroforming is modeled, containing the subprocesses preforming, calibration and welding. The second process is injection moulding.

7.1 Hydroforming

7.1.1 Preforming and calibration

Preforming

In preforming there are two different input variables, the volume V of the hydroforming medium, pressed between the blanks and the clamping forces F, used to press together the two blanks. The output variable is the inner pressure P_ihu.

Having a look at figure 6.1 and 6.2 indicates that the curve can be modeled by two straight lines for each clamping force. Hence, linear approximation, as discussed in section 5.1, is applicable. Thus the pressure is approximated by

$$\grave{P}_\mathrm{ihu}(V) \approx P_\mathrm{ihu}(V_k) + P'_\mathrm{ihu}(V_k)(V - V_k) \ , \tag{7.2}$$

where k is the configuration used for the approximation. The values $P_\mathrm{ihu}(V_k)$ and $P'_\mathrm{ihu}(V_k)$ correspond to parameters of the output node. Thus the pressure has the discrete node V_d and F as parents. The force F has four different states. The node representing the volume V_d has two states, used to distinguish the two lines before and after the occurrence of leaks. A comparison of model 7.1 with figure 5.1 shows that the principle of linear approximation is used nearly without any changes. To improve the training results, the weight for the link $V \rightarrow e_\mathrm{Q}$ is fixed to -1. Additionally, some examples are removed from the training set, to ensure that both lines have approximately the same number of examples in the data set. The following initializations are

Table 7.1: Accuracy obtained with preforming model

Variable	F	V	P_{ihu}
Relative Error	6.9%	4.5%	2.3%

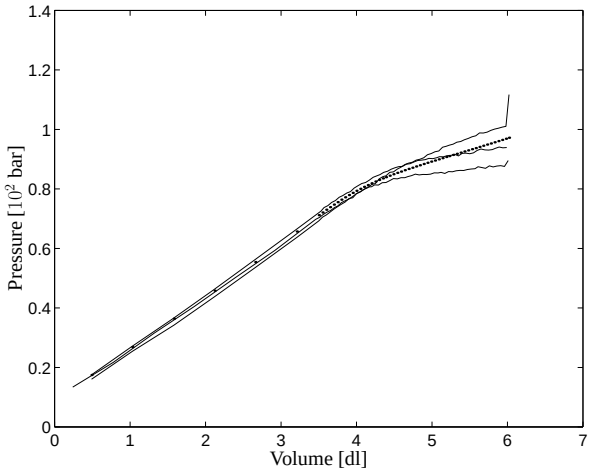


Figure 7.2: Preforming with $F = 200$ kN

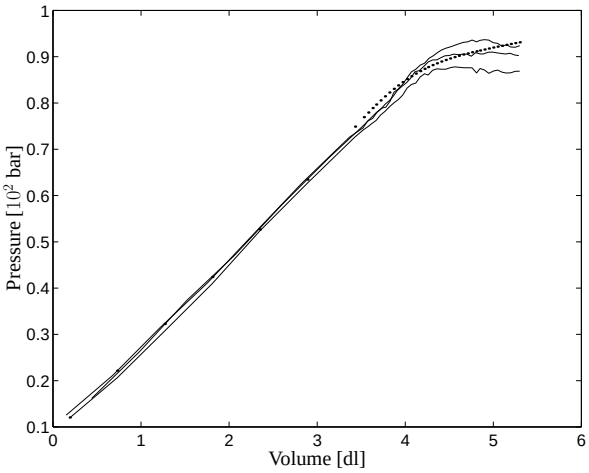


Figure 7.3: Preforming with $F = 300$ kN

used:

$$\alpha_{P_{\text{ihu}}} = \{0.4\ 0.45\ 0.5\ 0.5\ 0.92\ 0.86\ 1\ 1.14\} \tag{7.3}$$

$$\gamma_{P_{\text{ihu}}} = \{0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\} \tag{7.4}$$

$$\alpha_{e_{\text{Q}}} = \{2\ 2\ 2\ 2\ 5\ 5\ 5\ 5\}\ . \tag{7.5}$$

The means $\alpha_{P_{\text{ihu}}}$ of the node P_{ihu} are chosen approximately equal to the pressure $P_{\text{ihu}}(V_k)$ at the points V_k. The points V_k are defined by $\alpha_{e_{\text{Q}}}$ and $\alpha_V = [2\ 5]$. The selections are obtained directly from figures 6.1 and 6.2.

After training with the EM-algorithm (approximately 20 iterations), cross validation is executed with the results specified in table 7.1.

For evaluation the clamping forces $F = 200\ \text{kN} \cdots 500\ \text{kN}$ are assigned to the different states of F.

Assuming that there is an equal number of examples for all clamping forces, the mean of the forces is 350kN. A relative error of 6.9% corresponds to an error of 24.15 kN. Moreover the volume and the pressure are modeled with high accuracy. Figures 7.2 to 7.5 depict the prediction of the pressure for different clamping forces $F = 200\ \text{kN} \cdots 500\ \text{kN}$. There is almost no difference between the predictions, indicated by dotted lines and the three actual data sets (solid

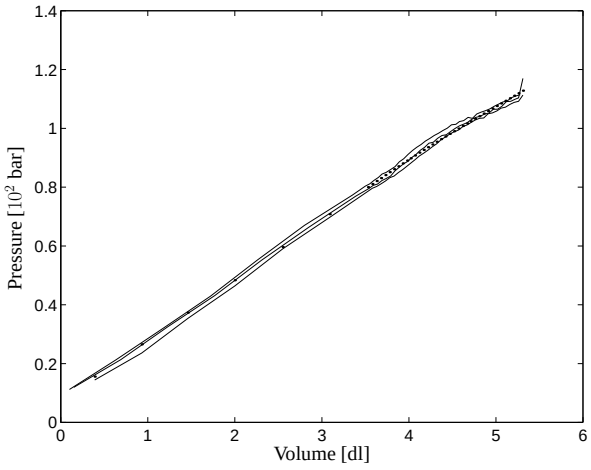


Figure 7.4: Preforming with $F = 400$ kN

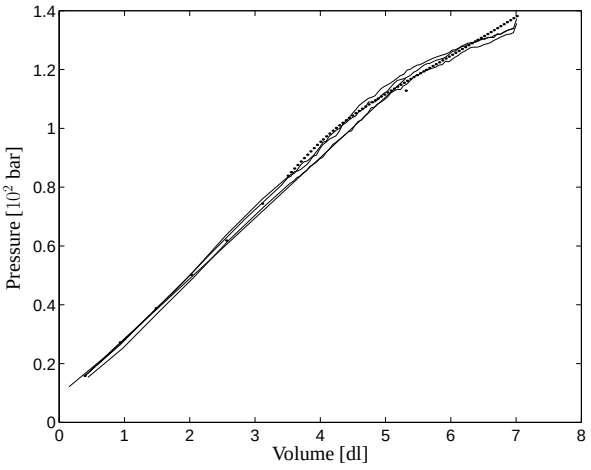


Figure 7.5: Preforming with $F = 500$ kN

lines). The points are closer for high volumes. This is caused by the fact that not all points are used for training to guarantee a similar number of data points before and after the occurrence of first leaks. Similar results are obtained with variational approximation, a method which uses continuous nodes as parents of a binary, discrete node. This binary node is used to distinguish between the two lines, before and after occurring of leaks, for each clamping force. This model is discussed in [DDN00b], but has the drawback of slower training.

Calibration

Directly after preforming the calibration takes place[1]. Figures 6.3 and 6.4, which show the relationship between the volume and the pressure, illustrate that a nonlinear curve has to be modeled. Additionally, a non-continuous point, caused by bursting, has to be modeled (Compare figure 6.3 at $V = 10.5$ dl to figure 6.4, $V = 10.2$ dl).

As in preforming, the technique of linear approximation is used. In comparison to preforming the clamping forces are fixed, so that F is no longer used as input node. But different models are used for different clamping forces, as the result of calibration depends on the forces used during preforming.

As a result of this consideration, the principle model of figure 5.1 can be used without any changes. Care should be taken when selecting the number of states for node V_{d} (see figure 7.6). The non-continuous point requires a small covariance. To ensure that the pressure for high volumes is also predicted correctly, the region after bursting is modeled by two different states.

[1]The data to be modeled are gathered in the second phase of the special research center 396. In the meantime the preforming process is followed by the welding process. Calibration is executed after welding.

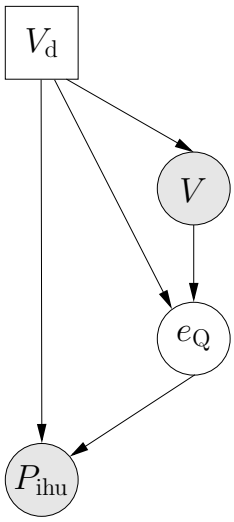


Figure 7.6: Model of the calibration process

One is responsible for the abrupt change, when bursting takes place, and one is used to keep the predictions low for high volumes.

A very important point, when using a linear approximation, is the correct initialization of the model. For example, the experiments discussed later on are carried out with 6 different states for node V_{d} and the following initializations for $F = 200\ \mathrm{kN}$ (for each force a model with different initializations is trained).

$$\alpha_V = \{6.2 \pm 0.1 \;\; 6.5 \pm 0.1 \;\; 9.0 \pm 0.1 \;\; 10.0 \pm 0.1 \;\; 11.0 \pm 0.1 \;\; 11.5 \pm 0.1\} \quad (7.6)$$

$$\gamma_V = \{0{+}10 \;\; 0{+}10 \;\; 0{+}10 \;\; 0{+}10 \;\; 0{+}10 \;\; 0{+}10\} \quad (7.7)$$

$$\alpha_{P_{\mathrm{ihu}}} = \{200 \pm 2 \;\; 100 \pm 5 \;\; 150 \pm 5 \;\; 170 \pm 5 \;\; 10 \pm 1 \;\; 10 \pm 1\}\ . \quad (7.8)$$

The notation 6.2 ± 0.1 means that the parameter is initialized with 6.2 plus a normally distributed (mean set to zero, standard deviation to one) random variable multiplied by 0.1.

To test the model, the data, discussed in subsection 6.1.1, is divided in a training set, containing 90% of the data, and a validation set with 10% of the data. Afterwards predictions are calculated for the validation set. The results plus the standard deviation are itemized in table 7.2. The first impression is that the prediction of the pressure is inaccurate, which is misleading in most of the cases. As figures 7.7 till 7.10 indicate; the dotted line, representing the prediction of the Bayesian network, is close to the original data, depicted by a solid line.

The high error is caused by predictions close to the bursting point. Here, an accurate prediction is nearly impossible due to scattering in the data.

Table 7.2: Relative error plus standard deviation for prediction of the volume and the pressure

Force F	200 kN	300 kN	400 kN	500 kN
V	5.43 $\pm$ 6.80%	4.65 $\pm$ 9.43 %	4.83 $\pm$ 8.41%	3.02 $\pm$ 5.15 %
P_{ihu}	53.38 $\pm$ 136.66%	38.97 $\pm$ 96.47 %	31.45 $\pm$ 86.22%	36.62 $\pm$ 95.16 %

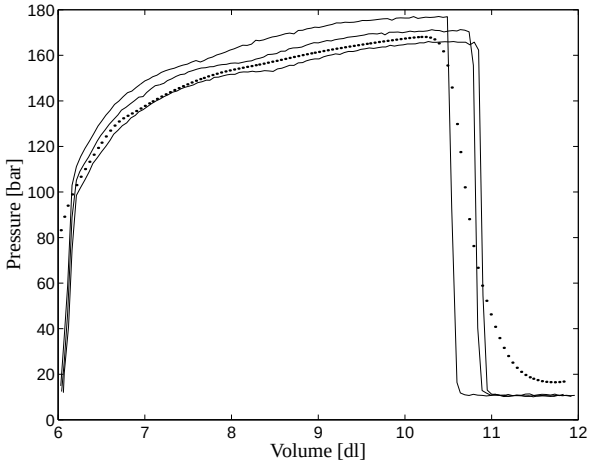


Figure 7.7: Prediction of the calibration pressure, $F = 200$ kN

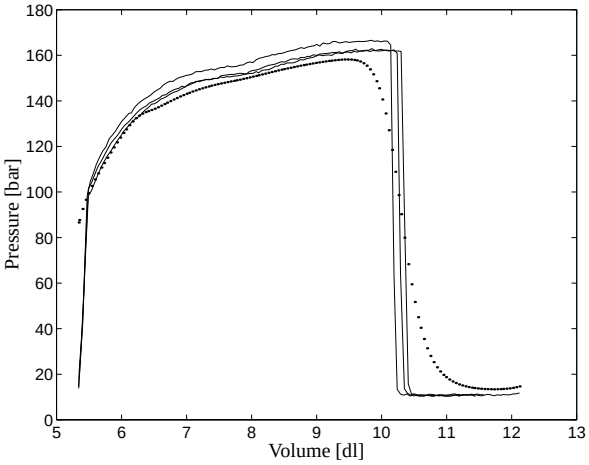


Figure 7.8: Prediction of the calibration pressure, $F = 300$ kN

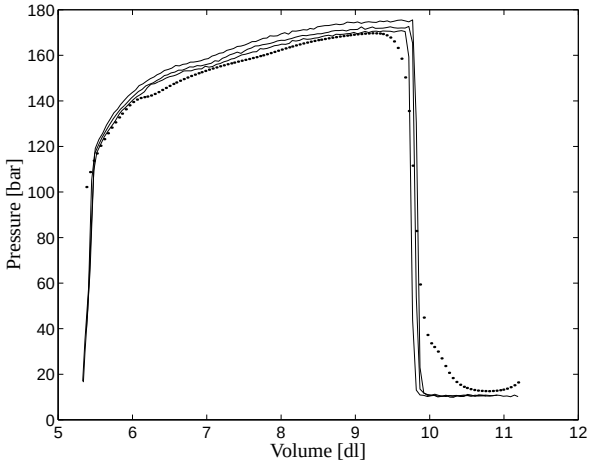


Figure 7.9: Prediction of the calibration pressure, $F = 400$ kN

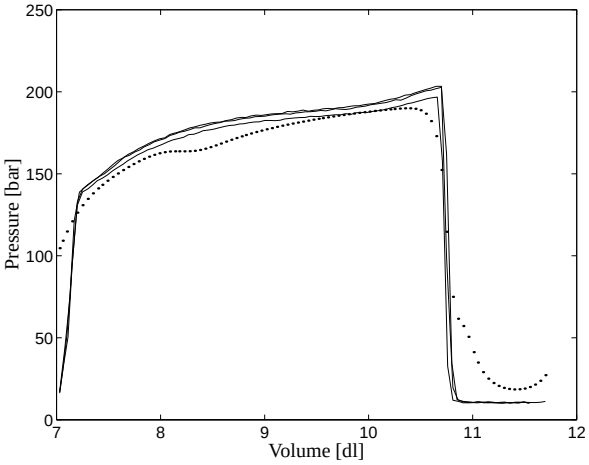


Figure 7.10: Prediction of the calibration pressure, $F = 500$ kN

Table 7.3: Control points, forces given in kN

Number	F_{fl}	F_{rl}	F_{rr}	F_{fr}	F_{A}	F_{C}	F_{E}	F_{H}	F_{K}	F_{M}
1	50	50	50	50	41	43	34	49	21	34
2	94	113	94	87	63	67	52	76	41	61
3	100	100	100	100	66	65	52	72	36	59
4	150	150	150	150	108	102	83	109	64	96
5	73	241	200	50	87	75	61	103	69	99
6	200	200	200	200	139	132	108	143	87	126
7	260	262	139	193	129	150	126	173	106	133
8	250	250	250	250	179	169	135	173	112	161
9	300	300	300	300	216	206	165	211	139	195
10	335	300	300	150	171	180	150	213	144	182
11	350	350	350	350	249	237	191	242	163	224
12	365	319	227	358	209	226	184	236	154	195
13	392	396	305	396	237	250	201	258	173	223
14	400	400	400	400	280	264	210	266	183	253

High differences between the actual values and the prediction take place for very small volumes. In all cases, the predicted values are too high (compare the solid and the dotted lines in figures 7.7 to 7.10). The reason might be a low number of data for that section of the curve. A changed initialization is not expected to lead to an improvement, since the selected values are at the lower end.

In the next section the distribution of the forces occurring in the press are modeled. This process is of great importance for control.

7.1.2 Modeling the forces of the press

An important point in hydroforming, having a major influence on the result, are the forces found at the load cells. It is the aim to have similar or equal forces at all points. This results in an equal movement of the blank in the form. Then the thickness of the blanks remains more or less equal which helps to avoid bursting during calibration. The relationship between the forces at the cylinder $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}, F_{\mathrm{fr}}$ and at the load cells $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$ (compare figure 6.5) is modeled to support the control. In a first step 69 experiments are executed to gather training data for the Bayesian network. Afterwards, the model is used to make predictions for 14 yet un-presented points listed in table 7.3. The results are compared with reality.

Given the forces at the load cells $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$, predictions for the inputs $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}, F_{\mathrm{fr}}$ are also calculated. For further judgment cross-validation is applied.

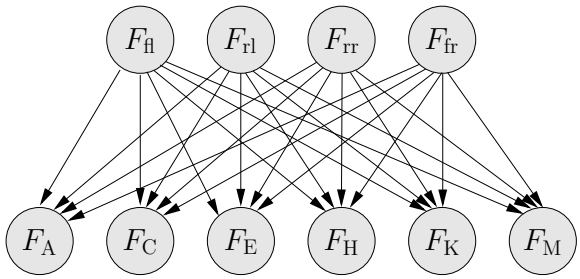


Figure 7.11: Bayesian network for modeling the forces of the hydroforming press

The used model is depicted in figure 7.11. Each cylinder force $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}, F_{\mathrm{fr}}$ is connected with each output force $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$. All nodes are continuous ones, that is the model is purely linear with no hidden nodes. Thus the initialization is unimportant, the EM-algorithm used to train the 44 parameters (4 means and dispersions for the input nodes, 6×4 weights, 6 means and dispersions for the output nodes) converges immediately. That is the EM algorithm calculates the correct parameters in the first iteration. the second iteration is used to calculate the log-likelihood of the model, the third iteration detects convergence.

To test the model, predictions for 14 control points are made. The relative error of the predictions is listed in table 7.4 in the columns $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$. The result for the best and the worst position is depicted in figures 7.12 and 7.13.

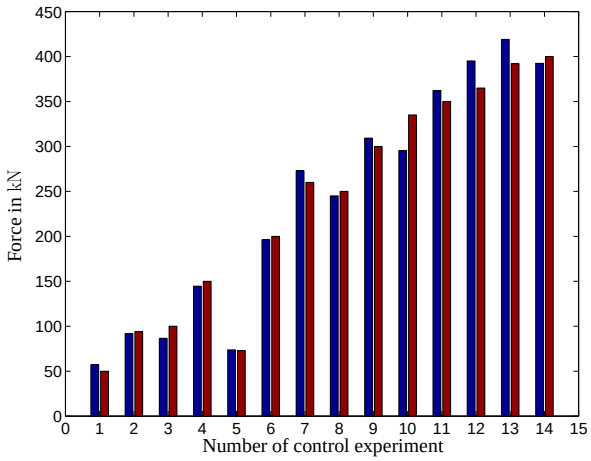


Figure 7.12: Prediction of the used force at the cylinder at the front, left hand side (best result)

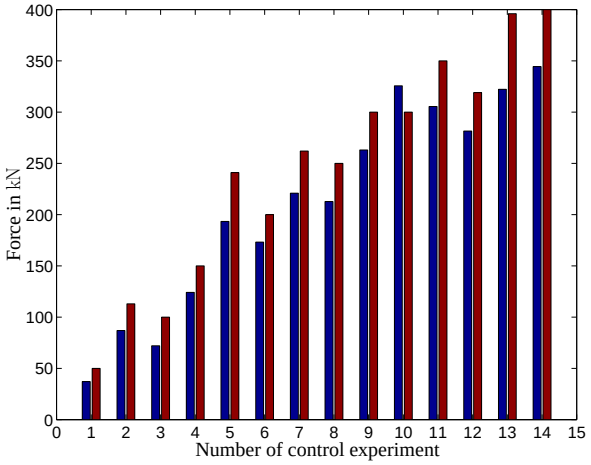


Figure 7.13: Prediction of the used force at the cylinder at the rear, left hand side (worst result)

The bar at the left hand side represents the prediction of the Bayesian network, the bar at the right hand side the reality. The same experiments are also executed with neural networks

Table 7.4: Relative error for control experiment

F_{fl}	F_{rl}	F_{rr}	F_{fr}	F_{A}	F_{C}	F_{E}	F_{H}	F_{K}	F_{M}
5.6%	16.8%	9.7%	10.9%	4.7%	5.8%	4.9%	9.6%	5.0%	5.9%

and regression polynomials. As discussed in [WBS+01] all three techniques are well suited for prediction of the forces $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$.

In contrast to regression polynomials and neural networks [2], Bayesian networks are also able to predict a required input to get a desired output. This is possible as Bayesian networks model a distribution of all variables. The model itself does not distinguish between in and output variables, even if in most of the cases the links are directed from the input to the output. To examine, whether the trained model could also act as a controller, the six output forces measured for the points listed in table 7.3 are entered as evidence and predictions for the forces at the cylinders $F_{\mathrm{rl}}, F_{\mathrm{rr}}, F_{\mathrm{fl}}, F_{\mathrm{fr}}$ are calculated. The relative error of the prediction is listed in table 7.4. In comparison to the prediction of the forces at the load cells $F_{\mathrm{A}} \cdots F_{\mathrm{M}}$, the results are less accurate. A possible reason is that there might be no unique input which leads to the desired output. It is unlikely that the model is inadequate, as the output forces are predicted with high accuracy. To increase the quality of control additional nodes for an occurring error might be used, as done in the difference equation model in section 4.3. The results for F_{A} and F_{H} are depicted in figures 7.14 and 7.15. The prediction of the Bayesian network is at the left hand side, the observations at the right.

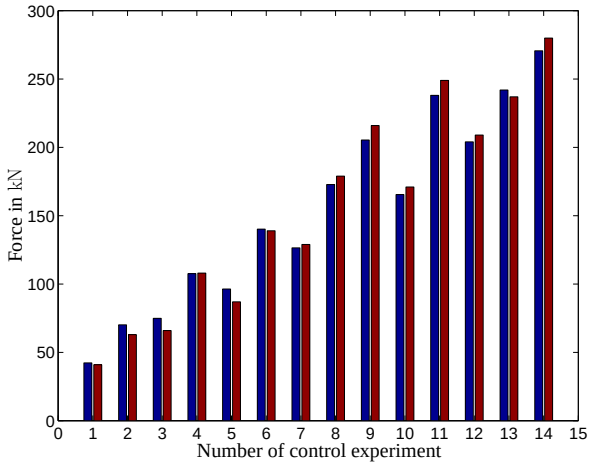


Figure 7.14: Prediction for force at load cell A

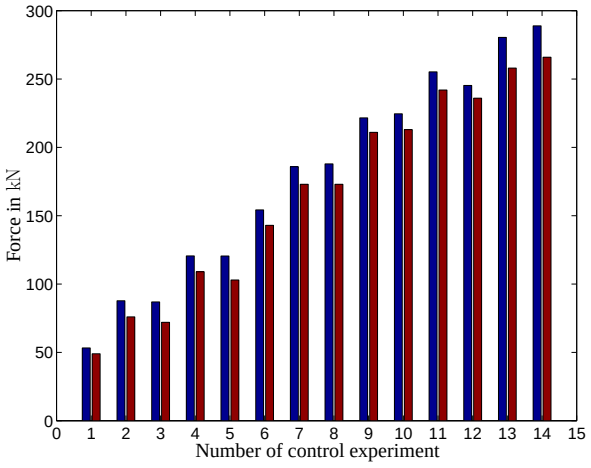


Figure 7.15: Prediction for force at load cell H

[2] The experiments concerning neural networks are done with a feedforward net, trained with backpropagation. Other topologies, e.g. Boltzmann machines or Hopfield net are also able to act as associative memory.

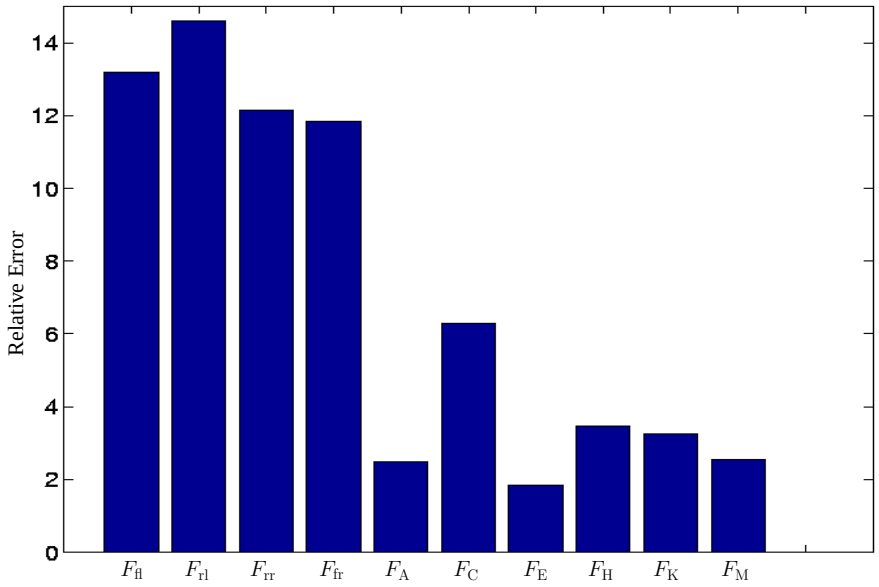


Figure 7.16: Results of cross validation

Table 7.5: Standard deviation of relative error

$\sigma(F_{\mathrm{fl}})$	$\sigma(F_{\mathrm{rl}})$	$\sigma(F_{\mathrm{rr}})$	$\sigma(F_{\mathrm{fr}})$	$\sigma(F_{\mathrm{A}})$	$\sigma(F_{\mathrm{C}})$	$\sigma(F_{\mathrm{E}})$	$\sigma(F_{\mathrm{H}})$	$\sigma(F_{\mathrm{K}})$	$\sigma(F_{\mathrm{M}})$
11.3%	14.5 %	11.0%	12.3%	3.0%	5.1%	2.9%	2.7%	3.2%	3.1%

For further validation of the model, cross-validation is used. That is the union of the former training and control-data is split arbitrarily in a training set, containing 90% of the data, and a validation set. This procedure is repeated 10 times with arbitrarily generated training and validation sets. The relative error of the results is displayed in figure 7.16, the standard deviation of the relative error is given in table 7.5.

A comparison with table 6.4, indicating the standard deviation of six measurements for the forces $F_{\mathrm{rl}} = F_{\mathrm{fl}} = F_{\mathrm{rr}} = F_{\mathrm{fr}} = 225\ \mathrm{kN}$, shows a strong correlation between the relative error and the reproducibility of the experiments. For example, the best results are obtained for F_{E},F_{A}, and F_{M} with an error of approximately between 1.8 and 2.5%. The standard deviation of the experiments is 0.3 and 1.7% of the used forces.

Worst results are observed when F_{C} is predicted. The relative error in this cases is more than 6%. At this point the error seems to be caused by the low reproducibility. The standard deviation at point C is approximately 8% of the used forces.

The discussion shows that a linear model is suited to model the forces of the press. For control without feedback an accuracy of 12% can be expected, the error can be further reduced

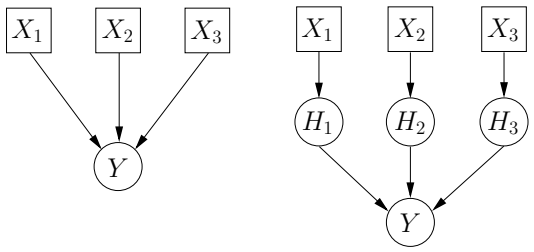


Figure 7.17: Example for Bayesian network with untrainable parameters (left hand side) and model based on Bayesian regression

when the deviation between current and desired forces is used for further corrections.

In the next subsection a model for the welding process is discussed. The main problems to be solved are occurring nonlinearities and the fact that not all configurations for the discrete nodes are observed.

7.1.3 Laser beam welding

The process of laser beam welding takes place directly after preforming. One of its effects is that the leaks, occurring at the end of preforming, are sealed, so that calibration can be started. The parameters are discussed in section 6.1.3, a good overview is given by figure 6.6.

When looking at the data three parts can be distinguished. In the main part the number of joints is kept constant, defocussing and the joint-offset are set to zero. In this part of the data the angle, the joint type (Either lap edge joint or lap seam joint), and the velocity are changed. Additionally, some of the blanks are contaminated by hydroforming medium, and some are not.

In the second part the influence of the velocity and the number of joints on the quality of the joint is tested. The rest of the parameters is not changed.

In the third part the influence of defocussing and offset is tested.

It is important to map the different parts of the data also to the model. If that is not done, this might lead to un-trainable parts of the Bayesian networks which can easily be identified. Imagine, for example, that the influence of three binary parameters $X_1 \cdots X_3$ on a continuous parameter Y or on a binary parameter X_4 is tested. The idea of the test is to keep two parameters constant, e.g. fix their value to 1, and change the remaining parameter. This would result in a test similar to table 7.6.

Even if all three parameters $X_1 \cdots X_3$ have an crucial impact on Y, a model as shown on the left hand side of figure 7.17 contains un-trainable parameters. The reason is that node Y has

Table 7.6: Example for training data

X_1	X_2	X_3	Y
1	1	1	y_1
⋮			⋮
1	1	1	y_n
0	1	1	y_{n+1}
⋮			⋮
0	1	1	y_{2n}
1	0	1	y_{2n+1}
⋮			⋮
1	0	1	y_{3n}
1	1	0	y_{3n+1}
⋮			⋮
1	1	0	y_{4n}

three discrete parents. Thus, it has 8 different parameters for the mean and covariance, one set of parameters for each possible configuration. Since only 4 of them are present in the training set, the parameter belonging to the remaining 4 configurations will remain more or less untrained. Thus the model will show an arbitrary response when one of the un-presented configurations is tested.

The model on the right hand side avoids this disadvantage. The effect of each variable is trained independently. The price is that the model is unable to learn interactions between two or more variables. Similar effects occur when the influence on a discrete variable has to be learnt. In this case a table with the conditional probabilities $\mathrm{P}(x_4|x_1, x_2, x_3)$ is assigned to the node X_4. Once again the entries for unseen configuration cannot be trained.

It should be mentioned, that the example presented above is not pathological. In product engineering a common problem is to find out the influence of several factors on the quality of the product. Particularly, if many factors have to be tested, it is too expensive to test all configurations. In many cases it is reasonable to assume that interactions between three and more variables have no significant influence on the result (see introductions to quality management, e.g. [Pfe93; Mar94]). This leads to the idea of test plans. Suppose that two settings per parameter are sufficient for the test plan and that the engineer assumes that the combination of $X_1 \cdots X_3$ has no effect on the result. In this case it is possible to set a forth parameter X_4 equal to the product $\cdot$ of X_1, X_2, X_3.

The product $\cdot$ is defined as a commutative and associative operation with $+ \ \cdot \ + = +$,

Table 7.7: Example for a test plan

X_1	X_2	X_3	$X_4 = X_1 \cdot X_2 \cdot X_3$	Y
+	+	+	+	y_1
⋮				⋮
+	+	+	+	y_n
-	+	+	-	y_{n+1}
⋮				⋮
-	+	+	-	y_{2n}
+	-	+	-	y_{2n+1}
⋮			⋮	
+	-	+		y_{3n}
-	-	+	+	y_{3n+1}
⋮				⋮
-	-	+	+	y_{4n}
+	+	-	-	y_{4n+1}
⋮				⋮

$+ \cdot - = -$, and $- \cdot - = +$. This results in the test plan outlined in table 7.7 where the effects of X_4 and $X_1 \cdot X_2 \cdot X_3$ cannot be distinguished. This example should reveal two main points. First, that missing configuration might result from well defined test plans, and second that according to the test plan the effect of special interactions cannot be identified and therefore it is not worthwhile modeling them.

In the process of laser beam welding it is therefore necessary to model the influence of the three data parts independently. In the first part the influence of the velocity, the angle and the type of joint on the force is modeled. An F-test [Rin97] shows that the combination of the angle and the type of joint is significant. Thus it is necessary to combine the influence even if not all configurations are tested. To avoid the problem of un-trainable parameters the deterministic node H_2 (confer figure 7.18), which has four different states, is introduced. Its conditional probabilities are defined according to table 7.8 and are not changed during training. The idea is that each of the four observed configurations is mapped to an own state of H_2. The two unobserved states, a setting angle of plus and minus 30° together with the lap edge joint, are mapped with a probability of 0.48 to the observation made for plus/minus 30° for the lap seam edge and with a probability of 0.48 to the observations made for an angle of 0 together with the lap edge joint. It is not proven that this mapping is correct, but the results are in a reasonable order of magnitude, that is a tensile force of approximately 4000 N is predicted. The influence of the velocity is not linear, as figure

Table 7.8: Probability of node H_2

Angle	Type of joint	State H_2	Probability
0	Lap edge joint	1	0.97
-30°	Lap edge joint	1	0.01
+30°	Lap edge joint	1	0.01
0	Lap seam joint	1	0.01
-30°	Lap seam joint	1	0.01
+30°	Lap seam joint	1	0.01
0	Lap edge joint	2	0.01
-30°	Lap edge joint	2	0.97
+30°	Lap edge joint	2	0.01
0	Lap seam joint	2	0.01
-30°	Lap seam joint	2	0.48
+30°	Lap seam joint	2	0.01
0	Lap edge joint	3	0.01
-30°	Lap edge joint	3	0.01
+30°	Lap edge joint	3	0.97
0	Lap seam joint	3	0.01
-30°	Lap seam joint	3	0.01
+30°	Lap seam joint	3	0.48
0	Lap edge joint	4	0.97
-30°	Lap edge joint	4	0.01
+30°	Lap edge joint	4	0.01
0	Lap seam joint	4	0.01
-30°	Lap seam joint	4	0.48
+30°	Lap seam joint	4	0.48

6.7 shows. There is a maximum of the force F, if the velocity v is approximately 4 $\mathrm{m/min}$. For high velocities the weld penetration is too low, which results in a lower tensile-strength. If the velocity is too small the weld penetration for the lap edge joint gets too high. That means that also in this case a low tensile-strength is obtained. For a velocity of approximately 4 $\mathrm{m/min}$ an optimal value is reached. Hence a simple connection between a node v to F_{H1} is not sufficient. Thus an additional node for v^2 is used, so that a polynomial is used for modeling the relationship between v and F.

The relationship between the number of joints and the tensile force is tested only for the lap edge joint for a constant angle of 0. The results are shown in table 6.8.

As there is again a maximum of the tensile-strength for a velocity of about 4 $\mathrm{m/min}$, the applied modeling technique is the same as for the influence of the angle and the type of joint. Node H_1 is used to calculate the difference between one and two joints. The difference is added to F_{H1} and assigned to node F_{H2}. Assignment means that the initialization of the parameters of node H_1 and F_{H2} is done, so that $F_{H2} \sim H_1 + F_{H1}$. For example the initial weight vector for F_{H2} is equal to $[1\ 1]$.

The third part models the influence of the offset and defocussing. Regarding figure 6.8, it turns out that this influence can be modeled more or less as a binary process. The idea is to compare both offset and defocussing with a threshold. If the input value, e.g. the offset, exceeds the threshold the quality of the joint decreases dramatically. According to figure 6.8 the threshold for the offset is somewhere between 0.3 and 0.6 mm, the threshold for the admitted defocussing between -4 and -2 mm. The threshold is controlled by the two binary discrete nodes $offset_{\mathrm{d}}$ and $defocussing_{\mathrm{d}}$. The binary, discrete node H_3 combines the two values. If one threshold is exceeded, the welding might fail. This combination is realised deterministically to avoid wrong predictions if both thresholds are exceeded.

It was mentioned that some of the parameters of the introduced model are not trained. There are two reasons. First, the parameters of the input nodes are not trained at all. The main reason is that the test data are not uniformly distributed. Thus, a training of the input nodes would lead to predictions being equal to the value mostly seen during training. Thus the prediction of values for the input nodes would strongly depend on the selected training examples of the test plan. Sometimes the parameters of hidden nodes are also clamped to achieve a special behavior of the model. This helps to reduce the overall number of parameter to 44. In the following, the results of the model are discussed.

The process of laser-beam welding is tested with 48 different configurations, whereas the tests for each configuration are repeated six times. To test the model, training took place with 47

Table 7.9: Relative error

Offset	Defocussing	Nb. of joints	Velocity	Angle	Joint type	Tensile strength
66.4%	84.6%	38.7%	22.1%	50.4%	15.5%	17.04%

configurations. Predictions are figured out for the remaining configuration.

Table 7.9 shows the relative error, but the results are sometimes misleading. To avoid division by zero only cases with a current value not equal to zero are included.

Prediction of the tensile strength The classification into good and defective joints using defocussing and offset works as intended. Only for two configurations (Offset = 0 respectively 0.3 mm, defocussing = -2 mm) strong deviations are observable. The prediction of the tensile strength F is 2985 N (3007 N) instead of $F = 4876\ \mathrm{N}$ (4384 N). The main reason is the lack of data. To test the ability to make predictions for unknown data the tested data sets are not used for training. As there are only three blocks with defocussing = -2 mm, two of them are used for training. One of them leads to a proper joint, the other fails. This leads to a too low prediction.

The correct classification of the joint does not mean that an exact prediction of the force is possible. The first reason is the great dispersion ($\sigma = 506\ \mathrm{N}$) of the data used to test the influence of the defocussing and the offset. Together with the mean force of 2710 N this results in an unavoidable error of approximately 18%. Another source of error is that the failure of the joint is modeled as a binary event.

Data to explore the influence of the angle, the velocity, and the number of joints are modeled with greater accuracy. First there is less dispersion in the data ($\sigma = 144\ \mathrm{N}$), which results in a lower unavoidable error of 3.39% (in relation to a mean force $\bar{F} = 4247\ \mathrm{N}$ for good joints). In a comparative model, where defocussing and offset are disregarded, the force is modeled with a relative error of 5.37%.

Prediction of the offset Table 7.9 shows a relative error of 66.4% when predicting the offset. It should be noticed that all cases with an offset of 0 mm are not part of this calculation to avoid division by zero. In 31 of 48 blocks the tensile strength is explored for both defocussing and offset = 0 mm. The predicted offset for these cases is 0.04 mm. If a Bayesian network would be used to control laser beam welding this would not lead to a decreasing quality.

Since an offset of less than 0.3 mm does not lead to a decreased quality and an offset larger than 0.6 mm results in a faulty joint (Its not possible to make a statement for the interval between 0.3 and 0.6 mm as the offset was only tested for 0, 0.3 and 0.6 mm), the offset can be divided into equivalence classes. The first class includes all values between 0.0 and 0.3 mm, the second one all values larger than 0.6 mm. Using these equivalence classes there are only two wrong and

Table 7.10: Misclassifications, number of joints

Velocity	Misclassifications	Number of examples	Remarks
2	3	6	No data for different angles
3	7	20	
4	2	9	
5	3	9	
6	0	2	No data for different angles
7	1	2	No data for different angles

one questionable prediction. The first wrong prediction is made for an offset of 0.6 mm and a defocussing of -4 mm. As the defocussing is a sufficient reason for a defective joint, the model has no possibility to distinguish between different offsets. The second failure occurs for an offset of 0.6 mm and a defocussing of -2 mm. In this case a value of 0.2 mm is predicted which would lead to a joint in best order. The problem might be caused by an extrapolation, as an offset of 0.6 mm is an extremal value and all other examples with a defocussing of -2 mm lead to a proper joint. For a offset of 0.6 mm and a defocussing of 4.0 mm an offset of 0.39 mm is predicted. There is no mean to decide whether the predicted value is in the correct equivalence class.

Prediction of defocussing For the discussion of the result it makes sense to distinguish between two equivalence classes again. The first class contains defective joints with defocussing of less than -4 mm. The second equivalence class contains examples with a defocussing larger than -2 mm. According to these equivalence classes, there is one misclassification for an offset of 0.6 mm and defocussing of -4 mm. As the given offset is sufficient to explain the defective joint the model has no means to distinguish the two different equivalence classes.

A questionable prediction is made for a defocussing of -4 mm and an offset of 0.3 mm. In this case a defocussing of -2.72 mm is predicted. This might be a wrong prediction (it is not possible to determine the exact position of the threshold), caused by a high force for a defective joint.

Number of joints As already seen in table 6.8, the difference between the forces for one or two joints is smaller than 0.5σ for $v = 3$ or $v = 5\mathrm{m/min}$. Thus, for these velocities it is nearly impossible to distinguish between different number of joints. The results are itemized in table 7.10 which shows the number of misclassification for different velocities.

Prediction of the velocity For the node, representing the velocity, only the mean is trained. The variance is fixed to ten which is approximately three times of the estimation. So the dispersion is large enough to guarantee prediction in the tested range and to avoid a strong influence of the examples on the prediction.

Table 7.11: Classifications of angle

		Predicted angle		
		0°	-30°	30°
Correct Angle	0°	20	8	8
	-30°	0	4	2
	+30°	0	1	5

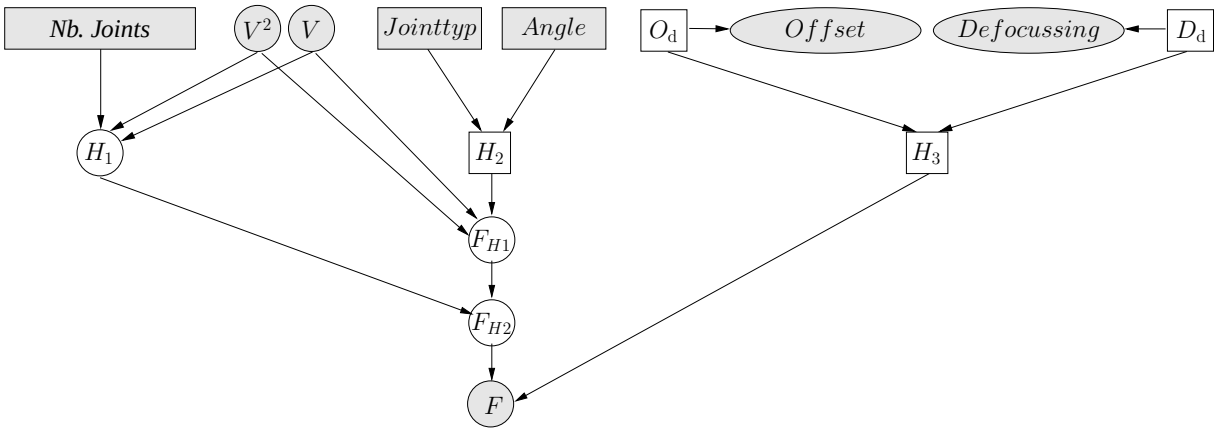


Figure 7.18: Model of laser beam welding

For the tested 48 blocks there are 8 cases with a prediction error larger than 1 $\mathrm{m/min}$. Two of them are cases with a defective joint where the velocity has no influence on the force. Three wrong predictions are made for lap seam joints which shows a very unregular dependency between velocity and force. For the remaining three cases the deviation is less than two $\mathrm{m/min}$, but there is no apparent reason for the deviation in these three cases.

Prediction of the angle For the angle misclassifications can be observed in 19 of 48 blocks. In most of the cases an angle of -30° (8 cases) or 30° (8 cases) is predicted instead of an angle of 0°. In four of these 16 misclassifications, predictions are made for a deficient joint, so that the angle has no influence on the joint. In four other cases, predictions are made for a velocity of 6 or 7 $\mathrm{m/min}$, so that a lack of data might be the cause of the error. A complete overview is given in table 7.11

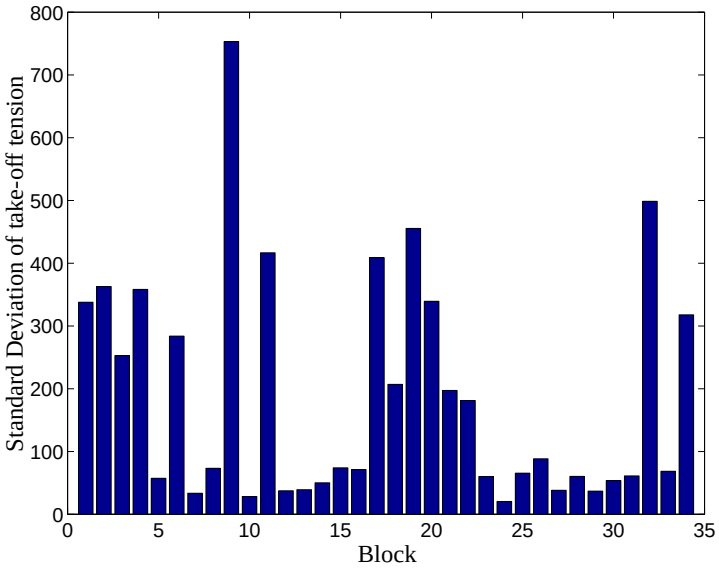


Figure 7.19: Standard deviation of take-off tension F

7.2 Injection moulding

In chapter 6.2 the executed test-plan is shortly discussed. Due to this test-plan only 34 configuration (15 inner experiments, 16 outer experiments, one central point and two possible operating points) are tested. Usually each setting is repeated 6 times (for some configurations outliers are removed, so that only four or five data sets are available), the experiments for the operating points are repeated 60 times. The standard deviation of the take-off tension is depicted in figure 7.19, points 1-15 are taken from the inner experiments, 16 – 31 are collected executing the outer experiments. The standard deviation of the operating points is given by bars 32 and 33, bar number 34 represents the standard deviation of the central point. The mean of the standard deviation is 223.2 N. In the calculation of the mean standard deviation, the values in figure 7.19 are weighted according to the number of experiments.

From $5^4 = 625$ possible configurations only 34 are observed. A comparison with a fraction of the test-plan for two arbitrary variables shows that even for a subset of two variables (compare table 7.12) only 11 (2 combinations for the operating point has to be added) from 25 configurations are observed. Thus the suggestion of subsection 7.1.3 to combine only variables if all of their configurations are observed is not applicable.

For most of the presented models in this subsection it cannot be expected that correct predictions are made for yet un-presented combinations (compare [DN01]). The model evaluation

Table 7.12: Fraction of executed test plan

Var_1	Var_2	F
2	2	
2	4	
4	2	
4	4	
1	1	
1	5	
5	1	
5	5	
3	3	

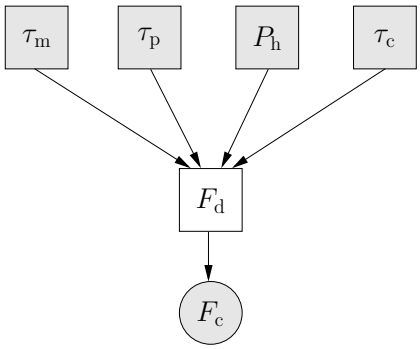


Figure 7.20: Discrete model of injection moulding

discussed in this section is therefore done on the training-set. In section 7.2.2 a new test plan is suggested. This test-plan restricts the number of different settings for each variable to three to reduce the number of experiments. The second principle is to use fully factorized test plans for a subset of the variables and to keep the rest of the variables unchanged.

7.2.1 Results

In a first experiment a model with only discrete input nodes (compare figure 7.20) is used. The results of the experiments with this model are listed in table 7.13 (compare also [EAD+03]). The temperature of the plastic is predicted with an relative error of 0.85%. In comparison to the mean melt temperature of $269.16°\text{C}$, this corresponds to an average deviation of $2.3°\text{C}$, 46% of the difference between two neighbored settings. The mean relative error of the preheating temperature is 11.82%, which means ($\overline{\tau_{\mathrm{p}}} = 171.04°\text{C}$) an average error of approximately $20.2°\text{C}$.

Table 7.13: Relative error of different models for injection moulding

Model	τ_{m}	τ_{p}	τ_{c}	P_{h}	F	Var_1	Var_2
Model figure 7.20	0.85%	11.82%	8.73%	12.27%	4.22%		
Linear model	2.31%	42.18%	14.93%	32.59%	10.25%		
Hybrid model figure 7.21	1.92%	40.46%	12.44%	28.45%	6.94%		
Work for injection	0.50%	13.57%	6.41%	7.63%	4.20%	6.79%	
Cushion	0.84%	11.92%	8.65%	11.82%	4.25%	57.49%	
Cushion and plasticizing	0.88%	11.85%	8.62%	12.00%	4.12%	57.04%	0.18%
Max cavity temperature	0.87%	12.49%		12.83%	4.04%	9.43%	

This is approximately 45% of the difference between two neighbored settings. The relative error of the cavity temperature is 8.73%, comparable to approximately $8.9°\mathrm{C}$ ($\overline{\tau_{\mathrm{c}}} = 102.15°\mathrm{C}$). This error is equal to 89% of the difference between two neighbored settings. This result indicates that the temperature of the cavity has only a minor influence on the tensile strength F (compare [ZE98]). The error for the prediction of the holding pressure is 12.27% or approximately 5.95 bar. This error is equal to 59% of the difference between to neighbored settings for the holding pressure. That is all the input variables are predicted with an accuracy smaller than the difference between two neighbored settings.

The relative error when predicting the tensile strength is 4.22%. In comparison to the mean force $\bar{F} = 3846\ N$, this error is approximately 162.3 N, which is smaller than the mean standard deviation within the data blocks.

A comparison of the discrete model to the linear model (see table 7.13), shows that a linear model is not adequate for the manufacturing process. The reason is that a manufacturing process, considered non-linear by the engineers, cannot be modeled by a pure linear model.

Also the hybrid model, depicted in figure 7.21, shows worse results than the discrete model. Thus the discrete model is used as base for the expansion. To enhance the prediction of the

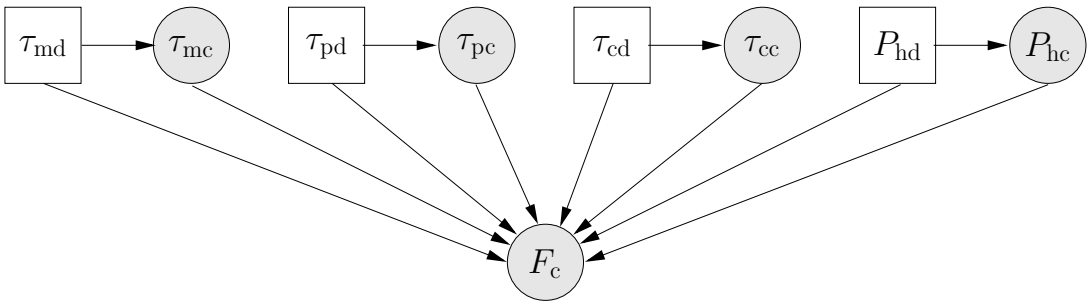


Figure 7.21: Hybrid model of injection moulding

tensile strength by additional measurements several variables are available. To keep the number of models to be tested small, a first selection is made, according to the correlation between the tensile strength and the measurements (see table 6.11 for the selected variables). In a first attempt the maximal cavity temperature is added. The used model is displayed in figure 7.22, the error is reduced by 0.18%. amount of additional information is relatively small. This is only a slight

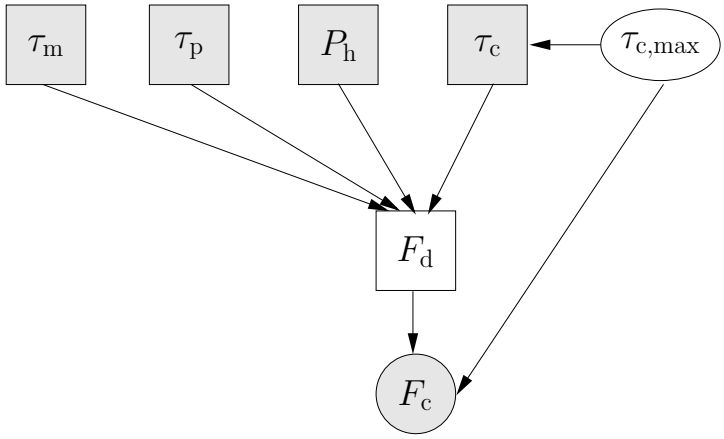


Figure 7.22: Model including the influence of the feed temperature

reduction, but the reader should keep in mind that the discrete model already uses all available input variables, so that the amount of additional information is relatively small. Additionally the error produced by the discrete model is smaller than the standard deviation, so that a great part of the error is due to scattering. The problem is that no predictions for un-presented data are made for validation, so that there is a large risk that over-adaptation to the training data has happened.

When exact prediction of the input variables is required, the work needed for injection should be added (see figure 7.23). The model including the cushion is best for the prediction of the preheating temperature, the model is similar to the one used for adding the injection work, only the injection work is replaced by the cushion. The results in table 7.13 show that all variables might be used to improve predictions, but the difference to the discrete model is small in most of the cases. The main problem is that due to the test plan a large part of the parameters are untrained. The hybrid model and the linear model might avoid this problem. But these two models provide the worst results; it is not tested whether these two models provide similar results when tested with cross validation.

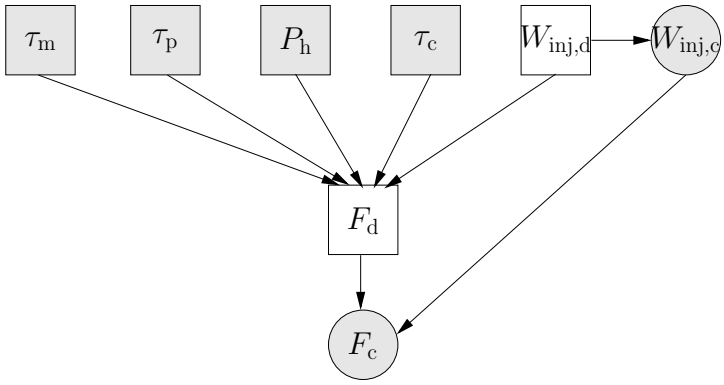


Figure 7.23: Model including the influence of the feed temperature

7.2.2 Test plan for injection moulding

The last section has revealed that the suggested models lack robustness. A fully factorized test plan would solve the problem, but this would severely increase the costs. To keep the experimental costs low, it is suggested to pick three variables and execute a fully factorized test plan for those three variables. The rest of the variables is kept constant. For an example see table 7.14. Value 2, used in this table, should be close to the operating point. Testing only three settings for

Table 7.14: Fraction of executed test plan

τ_{p}	τ_{m}	P_{h}	v	τ_{c}	F
1	1	1	2	2	
1	1	2	2	2	
1	1	3	2	2	
1	2	1	2	2	
⋮				⋮	
3	3	3	2	2	

each variable is forced by cost-pressure. On the one hand picking more than three variables for a test plan is very expensive. On the other hand it makes sense only in very rare cases as the combined influence of more than three variables can be neglected in most of the cases. Neglecting some of the possible combinations of three variables could further reduce the cost. This type of tests is closely related to the Bayesian network, so that the test plan results also in valuable hints for the structure to be used.

Chapter 8

Real-time

A controller has to react in real-time. That is the response time is limited by the sampling period ΔT. Particularly for nonlinear models as discussed in chapter 5 real-time requirement is hard to meet. In this chapter the reduction of time-slices is discussed which is one important step toward real-time. But further steps are necessary, e.g. the usage of approximate inference.

The run-time used for inference in a Bayesian network depends on the number of nodes n_N of the Bayesian network. For a dynamic Bayesian network, the number of nodes n_N is a multiple of the number of time-slices $t_{\max}$, and hence it is desirable to reduce the number of time-slices.

The experiments, discussed in the previous chapters, are performed with 10 time-slices for the representation of the past and 15 time-slices for present and future. The large number of time-slices has an effect on the run-time as well as on the time used for training. In particular for hybrid, dynamic Bayesian networks the time needed for inference

$$T_{\text{inf}} \approx n_{\text{s}}^{t_{\max}} \tag{8.1}$$

depends on the number of states n_{s} per time-slice and on the number of time-slices $t_{\max}$. Even if a reduction of the number of states n_{s} has a large effect on the run-time, this parameter is more or less dependent on the model and on the required accuracy. Hence there seems to be no general way to speed up the controller by reducing the number of states n_{s}. This remark does not mean that a reduction of n_{s} makes no sense in special cases, for an example see section 5.2. Reducing the number of time-slices $t_{\max}$ is much more promising. In the next two sections it is shown that the number of time-slices, needed for the representation of the past and future, can be severely reduced.

Another promising approach would be the usage of approximations, as discussed e.g. in

[BK98a]. It is not discussed in this thesis.

8.1 Number of time-slices

8.1.1 Dependency on the number of nodes used for the future

The number of time-slices is an important influencing factor on the run-time of the inference algorithm. The nodes, representing the future, have two main tasks.

First, the estimated inputs u_{t+i} are part of the calculation of the new input u_{new}, as described by equation (4.11). Sometimes it makes sense to use more than one input u_{t+i} for the calculation of the input to damp the control input u_{new}. Signals as displayed in figure 4.10(a) usually make no sense. The reason is that the actuator is not able to follow such high frequencies. A suitable number of estimated input signals is application dependent. The reader should keep in mind that there might be other methods to damp u_{new} than the usage of additional signals for the calculation of u_{new}.

The second function of the future nodes is the storage of the desired value w. It has to be tested whether a reduction of the number of time-slices has a negative influence on the results. Therefore the same experiments as in chapter 4 are executed. The only difference is the number of nodes n_{future} used for the representation of the future. No further reduction is possible at the limit $n_{\text{future}} = 2$. No evidence is given for the first time-slice of the future, as it is determined by the past (things might be different for jump Markov systems). The second time-slice of the future is needed to store the desired value w. Thus it is not possible to use less than two time-slices for the future.

The number of time-slices for the past is fixed to 10, the number of estimated signals u_{t+i} for the calculation of u_{new} is fixed to four. It is therefore possible to compare the results with the results of chapter 4.

Additionally, to the criteria discussed in chapter 4, the time for the training and the time for one inference step is listed. In our experiments the training is executed with 40 examples and 5 iterations of the EM-algorithm. The measurement of the evaluation-time includes the following steps:

1. Entering all observations for in- and output to a dynamic Bayesian network with $n_{\text{past}} + n_{\text{future}}$ nodes. This net contains no information about the desired value and is used for the estimation of the disturbance.

2. Estimation of the disturbance.

3. Entering the last n_{past} observations as the observed past and the desired value as the observed value in the future. To make the estimation of input signals more robust the estimation of the disturbance, obtained at step one, is used as additional evidence.

4. Estimation of signals u_{t+i} for the calculation of u_{new}.

The evaluation-time thus contains all measurements necessary for the calculation of the new signal. The measurings are done on a PC triggered with 2.4 GHz.

The results for system 2 (see table 4.1 for a definition of the test-systems and table 4.2 for an explanation of the criteria used to evaluate the systems) are listed in table 8.1. For the experiments, described by the last two columns, the number of signals used for the calculation of u_{new} is reduced to two respectively one. They only show that the controller works as intended for a low number of time-slices, e.g. the differences in the squared error sum are caused by the low number of signals u_{t+i} used for the calculation of u_{new}.

Table 8.1: Results of experiments with difference equation (system 2), 10 nodes used for the past

Number of future nodes	13	11	9	7	5	3	2
$I_{\text{d}}(z^d = 0)$	9.6685	9.6741	9.6785	9.6815	9.6800	8.5862	4.9942
$e_\infty(z^d = 0)$	0.0205	0.0172	0.0139	0.0106	0.0071	0.0033	0.0083
Overshoot	0.5405	0.5442	0.5482	0.5525	0.5584	0.1300	0.1203
$I_{\text{d}}(z^d = 1)$	0.1500	0.1500	0.1497	0.1493	0.1493	0.1404	0.1032
$e_\infty(z^d = 1)$	0.0244	0.0212	0.0173	0.0130	0.0095	0.0110	0.0170
$t_s(z^d = 0, 1\%)$ [s]	0.4500	0.4500	0.4500	0.4500	0.4500	0.3500	0.1500
$t_s(z^d = 0, 3\%)$ [s]	0.4000	0.4000	0.4000	0.4000	0.4000	0.2000	0.0500
$t_s(z^d = 1, 1\%)$ [s]	0.3000	0.3000	0.3000	0.3000	0.3000	0.3000	0.1500
$t_s(z^d = 1, 3\%)$ [s]	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.1500
Traintime [s]	279.6955	254.1279	230.0103	207.1388	181.9450	155.0763	143.4986
Evaltime [s]	0.6007	0.5503	0.4942	0.4428	0.3889	0.3332	0.3051

The effect of n_{future} on the run-time and training-time is depicted in figures 8.1 and 8.2. As the experiments are done for networks without discrete nodes, only a linear effect is observed. For hybrid dynamic Bayesian networks with discrete nodes a larger effect is expected.

The effect on the accuracy of the controller is depicted in figures 8.3 and 8.4. The results for $n_{\text{future}} < 4$ are omitted as the number of nodes that are used for the calculation of u_{new} is decreased in comparison to the other examples. Figure 8.3 shows the effect on the squared error sum. At a first glance, the reduction of n_{future} seems to have a negative effect, but the effect is only visible at the second position after decimal point.

Figure 8.4 depicts the effect of reducing n_{future} on the steady state error e_∞. Decreasing n_{future} has a positive effect on the steady state error. A further reduction until $n_{\text{future}} = 3$ has

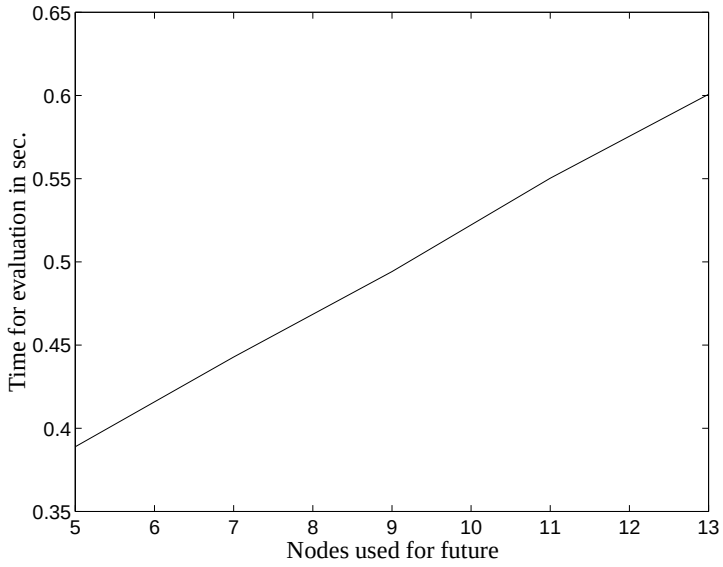


Figure 8.1: Time for the generation of the new input signal

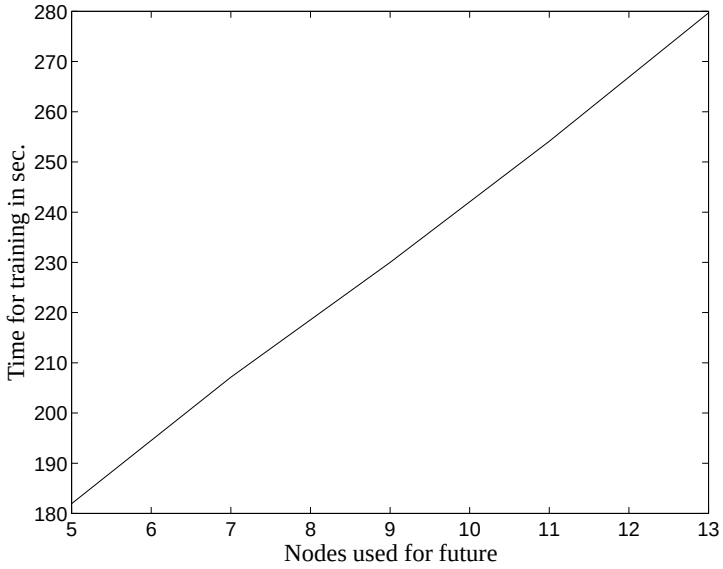


Figure 8.2: Time for training depending on n_{future}

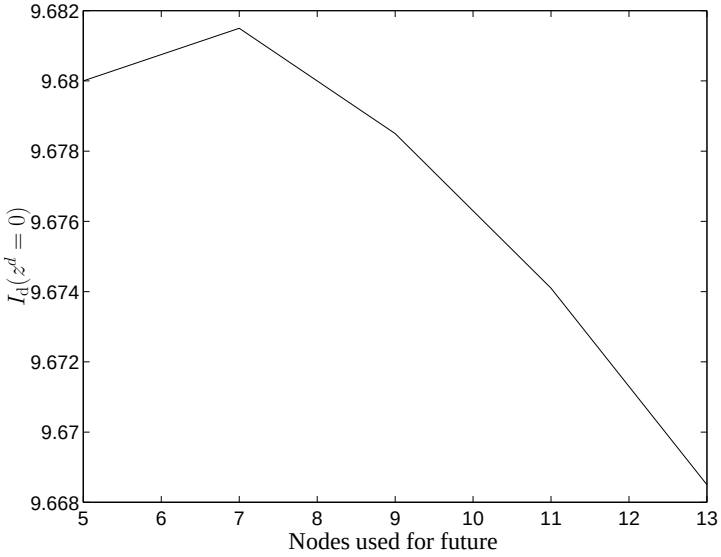


Figure 8.3: Squared error sum depending on number of time-slices

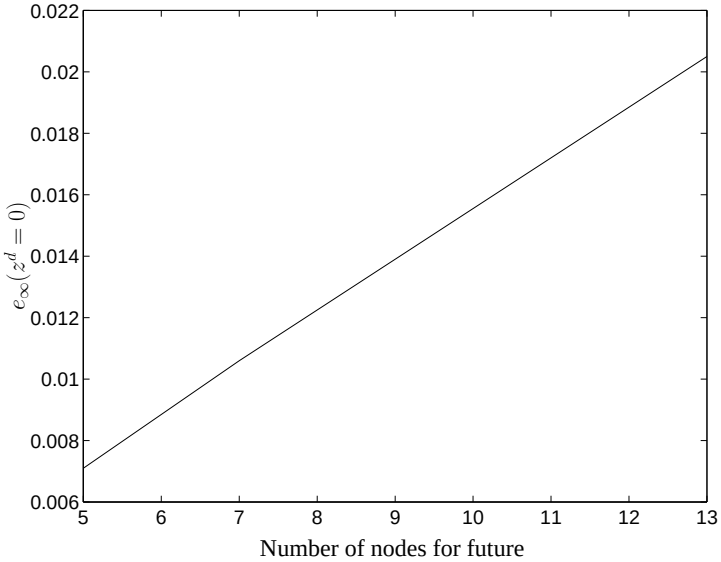


Figure 8.4: Steady state error depending on the number of nodes for the future

Table 8.2: Dependency of the accuracy on the number of nodes used for the future

Number of future nodes	13	11	9	7	5	3	2
$I_{\mathrm{d}}(z^d = 0)$	9.8833	9.8842	9.8838	9.8843	9.8829	8.9384	4.9675
$e_\infty(z^d = 0)$	0.0007	0.0010	0.0009	0.0005	0.0000	0.0007	0.0011
Overshoot	1.1388	1.1403	1.1402	1.1406	1.1403	0.2144	0.0040
$I_{\mathrm{d}}(z^d = 1)$	0.1575	0.1583	0.1578	0.1556	0.1522	0.1511	0.1133
$e_\infty(z^d = 1)$	0.0199	0.0215	0.0190	0.0111	0.0028	0.0110	0.0185
$t_s(z^d = 0, 1\%)$ [s]	0.6500	0.6500	0.6500	0.6500	0.6500	0.4000	0.0500
$t_s(z^d = 0, 3\%)$ [s]	0.4500	0.4500	0.4500	0.4500	0.4500	0.2000	0.0500
$t_s(z^d = 1, 1\%)$ [s]	0.5500	0.5500	0.5500	0.8500	0.8500	0.7500	0.6500
$t_s(z^d = 1, 3\%)$ [s]	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.1500
Traintime [s]	280.9525	255.7706	239.2738	281.2450	209.0050	157.5998	158.9270
Evaltime [s]	0.6029	0.5502	0.5092	0.9710	0.5654	0.3379	0.3317

the same effect. Setting n_{future} to the theoretical limit $n_{\mathrm{future}} = 2$ makes no sense, the steady state error increases to $e_\infty = 0.0083$. When n_{future} is set to two, there is only one node left for encoding the desired value. As a result the information might be missing that the desired value has to be kept constant.

Similar results are obtained for the experiments with system 3, listed in table 8.2. Also for system 3 a reduction of the number of nodes has nearly no impact on the accuracy. As a consequence of the discussion in this section n_{future} is set to five, the number of signals for the calculation of u_{new} is fixed to four. This setting enables a comparison between the results in this and the following section, and chapter 4.

8.1.2 Dependency on the number of nodes used for the past

The first n_{past} time-slices are used for the estimation of the past and the disturbance. Before the experiments are discussed, the lower limit for a SISO-system from the theoretical point of view is deduced. For the state space model, a dynamic Bayesian network to model a third order system is depicted in figure 8.5. Note that the number of state-nodes is equal to the order of the system. The structure is based on the assumption that the normal form, described in section 3.1.1, is used. The reason is that normal forms are essentially for good training results.

Provided that the in- and outputs are known for all time-slices, the third state $X^s_{3,t}$ can be estimated at time t based on the measurement of the output Y^m_t. For time-slice $t+1$ the first and the third state,$X^s_{1,t+1}$ and $X^s_{3,t+1}$, are assessable accurately. For the estimation of the second state $X^s_{2,t+1}$ information about the first state $X^s_{1,t}$ is missing. At $t+2$ enough information is available for the estimation of all states.

For higher order systems similar considerations lead to the result that at the first time-slice

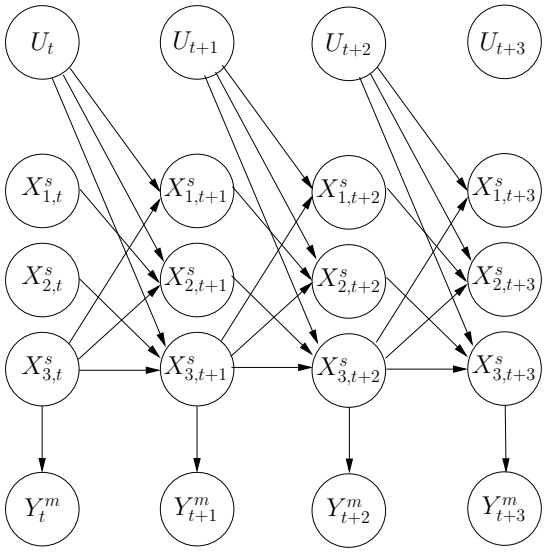


Figure 8.5: State space model of third order

only the last state can be estimated, for the second time-slice the first and the last one. For the estimation of all states of a system of n-th order n time-slices are needed.

If the structure of the DBN is based on the difference equation approach, the same result is obtained. To figure out the minimal number of time-slices equation (3.34) is helpful. It is necessary to have access to all signals $u_{t-1} \cdots u_{t-n}$ and $y^m_{t-1} \cdots y^m_{t-n}$ to make a prediction for Y^m_t. That is also for the difference equation n time-slices are necessary to make exact predictions for a system of n-th order.

The results of the experiments with system 2 are itemized in table 8.3. The first line shows that reducing n_{past} has nearly no effect on the squared error sum. The relationship between the number of time-slices for the past and the squared error sum $I_{\text{d}}(z^d = 0)$ is displayed in figure 8.6. But a reduction also has drawbacks. The steady state error $e_\infty(z^d = 0)$ raises from 0.0071 to 0.0105 for $n_{\text{past}} = 3$. If the steady state-error $e_\infty(z^d = 1)$ for the disturbance reaction is observed, it gets obvious that the controller is not working properly for $n_{\text{past}} = 3$. The steady state error for $n_{\text{past}} = 3$ raises to 0.0347. This value is 3 to 18 times larger than the steady state error for more time-slices. The results obtained for system 3, listed in table 8.4, agree with the results for system 2.

Reducing the number of nodes leads to nearly no changes in the squared error sum, but to a larger steady state error. However, a reduction to $n_{\text{past}} = 4$ seems possible, particularly when the run-time is taken into account The results discussed so far are obtained with the inference

Table 8.3: Dependency of the results on the number of nodes used for the past, system 2

Number of past nodes	10	8	6	5	4	3
$I_{\mathrm{d}}(z^d = 0)$	9.6800	9.6832	9.6836	9.6842	9.6861	9.7001
$e_\infty(z^d = 0)$	0.0071	0.0075	0.0080	0.0079	0.0084	0.0105
Overshoot	0.5584	0.5578	0.5574	0.5567	0.5566	0.5523
$I_{\mathrm{d}}(z^d = 1)$	0.1493	0.1444	0.1426	0.1432	0.1380	0.1217
$e_\infty(z^d = 1)$	0.0095	0.0045	0.0022	0.0019	0.0079	0.0347
$t_s(z^d = 0, 1\%)$ [s]	0.4500	0.4500	0.4500	0.4500	0.4500	0.4500
$t_s(z^d = 0, 3\%)$ [s]	0.4000	0.4000	0.4000	0.4000	0.4000	0.4000
$t_s(z^d = 1, 1\%)$ [s]	0.3000	0.3000	0.5500	0.5500	0.5500	0.5500
$t_s(z^d = 1, 3\%)$ [s]	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500
Traintime [s]	181.9450	157.2597	132.0516	119.3287	106.6231	94.1040
Evaltime [s]	0.3889	0.3359	0.2846	0.2557	0.2291	0.2021

Table 8.4: Dependency of the results on the number of nodes used for the past, system 3

Number of past nodes	10	8	6	5	4	3
$I_{\mathrm{d}}(z^d = 0)$	9.8853	9.8815	9.8792	9.8813	9.8824	9.8796
$e_\infty(z^d = 0)$	0.0005	0.0007	0.0024	0.0016	0.0013	0.0032
Overshoot	1.1435	1.1401	1.1424	1.1439	1.1439	1.1439
$I_{\mathrm{d}}(z^d = 1)$	0.1556	0.1445	0.1373	0.1465	0.1481	0.1301
$e_\infty(z^d = 1)$	0.0111	0.0052	0.0286	0.0203	0.0163	0.0410
$t_s(z^d = 0, 1\%)$ [s]	0.6500	0.6500	0.6500	0.6500	0.6500	0.6500
$t_s(z^d = 0, 3\%)$ [s]	0.4500	0.4500	0.4500	0.4500	0.4500	0.4500
$t_s(z^d = 1, 1\%)$ [s]	0.8500	0.8000	0.6000	0.6000	0.6000	0.6000
$t_s(z^d = 1, 3\%)$ [s]	0.2500	0.2500	0.2500	0.5000	0.5000	0.5000
Traintime [s]	282.0064	162.3054	136.7756	123.4775	112.0902	98.5522
Evaltime [s]	0.5778	0.3556	0.2992	0.2703	0.2433	0.2142

algorithm discussed in section 2.3.2. The drawback of this algorithm are numerical instabilities [LJ99]. It is not possible to use this algorithm for dynamic Bayesian networks with both discrete and continuous nodes. For networks with more than 5 time-slices the result of inference is Not a Number (NaN). As a consequence the stable algorithm, discussed in [LJ99], is examined in the next section.

8.2 Stable Inference algorithm

The stable inference algorithm avoids switching between the moment and canonical characteristic so that the matrix inversion in this step is avoided. Additionally the representation of a potential is changed. In the stable algorithm the continuous nodes are divided in head and tail nodes, the potential represents a distribution of the head nodes given the tail nodes, so that the dimension of the covariance matrix gets smaller.

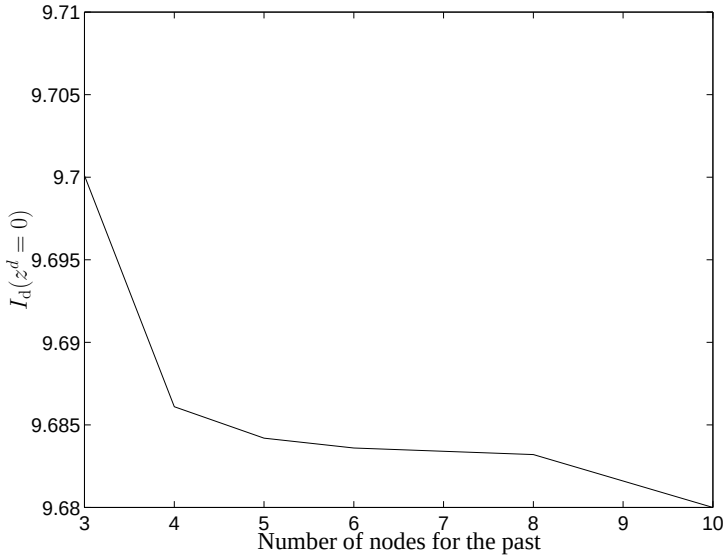


Figure 8.6: Squared error sum depending on number of time-slices

In this section the measurements for system 2, with $n_{\mathrm{past}} = 6$ and $n_{\mathrm{future}} = 5$ is repeated with the new inference algorithm to get an impression whether changing the inference algorithm has an impact on the run-time. A comparison between table 8.5 and table 8.3 shows that the usage of the stable inference algorithm has nearly no impact on the quality of the controller defined by the squared error sum, overshoot and steady state error. This coincides with the expectation as the used inference algorithm should have no impact on the calculated distribution. Comparing

Table 8.5: Runtime of the stable inference algorithm

Experiment number	1	2	3	4	5	6	7	8	9	10	Mean
$I_{\mathrm{d}}(z^d = 0)$	9.68	9.68	9.68	9.68	9.68	9.68	9.68	9.68	9.69	9.68	9.68
$e_\infty(z^d = 0)$	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0.01
Overshoot	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56
$I_{\mathrm{d}}(z^d = 1)$	0.14	0.15	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14
$e_\infty(z^d = 1)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$t_s(z^d = 0, 1\%)\,[s]$	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
$t_s(z^d = 0, 3\%)\,[s]$	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.40
$t_s(z^d = 1, 1\%)\,[s]$	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55
$t_s(z^d = 1, 3\%)\,[s]$	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Traintime [s]	726	736	754	699	698	698	697	697	700	696	710
Evaltime [s]	4.13	4.13	3.89	3.89	3.90	3.90	3.89	3.89	3.93	3.89	3.94

the time used for the training of the Bayesian network and the time for evaluation shows that the (current) implementation of the inference algorithm described in [LJ99] is approximately 14

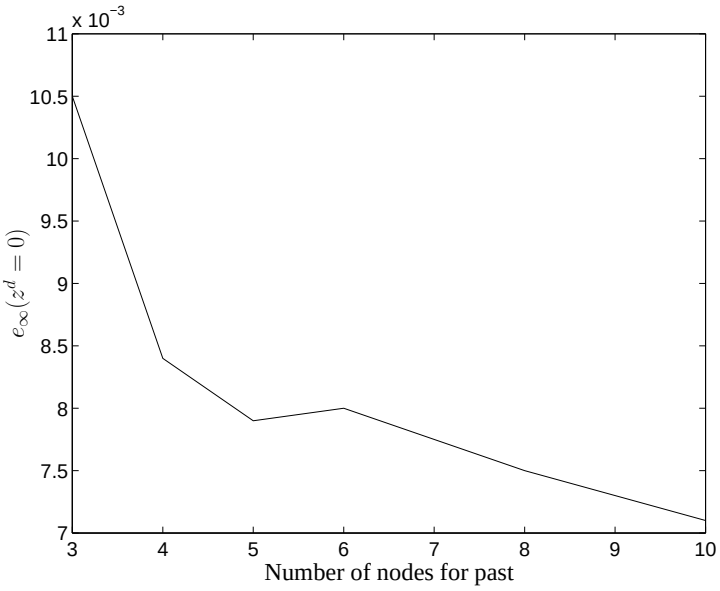


Figure 8.7: Steady state error depending on the number of nodes for the past

times slower (comparison of the evaluation time) than the implementation of the former algorithm [Lau92]. Therefore, it is advantageous to use the former version as long as no warnings (a warning is given e.g. when the matrix to be inverted is badly conditioned) are displayed during runtime.

Chapter 9

Outlook and Summary

9.1 Outlook

The thesis is divided into two main parts. The first part deals with the application of Bayesian networks as a controller. In chapter 4 and section 5.2 preconditions and suitable models are discussed that enable a dynamic Bayesian network to act as controller. Chapter 8 examines the run-time and simple means for its reduction.

The second part addresses modeling of manufacturing processes like hydroforming and injection moulding. In both areas large progress is achieved, yet there are still a lot of possibilities for further optimization. Section 9.1.1 analyzes methods to improve the Bayesian controller. An additional subject is the usage of alternative approaches to extend the possibilities of stochastic control.

9.1.1 Usage of Bayesian networks as a controller

In this section three different aspects are discussed. First, the Bayesian controller is regarded from the practical point of view. Problems which might occur due to differences between simulation and possible application are covered. The second paragraph deals with possible extensions of the system so that as many systems as possible can be controlled. Finally, alternative stochastic approaches are discussed.

Practical application The Bayesian controller is examined with simulated dynamic systems of second and third order. As *training signals* the impulse-, step-, and sine-responses (different frequencies are used) are applied. Using a broad range of frequencies simplifies the adaptation of the parameters. But signals collected during practical employment of the system have a different

characteristic. Usually a controller keeps the output constant. Provided that no disturbance occurs, the input signal is also constant. It is not possible to learn the dynamic of the system using constant signals. As long as only constant in- and output signals are collected, a dynamic system can be characterized by its gain. Possible solutions might be to store signals which are gathered during the occurrence of a disturbance or when the plant is started up.

A second point to be discussed is the robustness of the training process facing differences between the *structure of the Bayesian network* and the plant to be modeled. To explore the feasibility of Bayesian control, it was assumed that the order of the system and, as a result, the structure of the dynamic Bayesian network is known. Structure means either the number of state nodes or the order of the applied Markov model. To determine the order of the system several points of view are important. Control theory offers different methods to estimate the order of the system (see [Unb00], chapter 4.3). Some of them are applied before system identification takes place. Additionally it might make sense to simplify the model, for example to use a model of smaller order than the plant (see [Unb97a], chapter 9 for regularly used simplifications). The smaller number of state nodes results in less nodes. Therefore training and evaluation time is shortened. Assuming the equality of some time constants is another possibility to reduce the search-space and simplify system identification.

The application of structure learning algorithms comprises some open questions. It is not known how the special requirements of the manufacturing domain are taken into account. Particularly adding additional hidden nodes or learning a hybrid Bayesian network can be considered as a complex problem.

Before the EM-algorithm is applied, a suitable *sampling rate* has to be set. In this thesis the sampling rate is deduced from the natural angular frequency of the system and the sampling theorem. But this is no final solution as the formula given for the calculation of the natural angular frequency requires the knowledge of the time-constants of the system and is restricted to systems of second order. Both conditions are usually not met. A minimization of the sampling period does not solve the problem as it complicates the system identification and shortens the maximal inference time (The reader should keep in mind that, in order to meet real-time requirements, ΔT is the maximal inference time). Unbehauen suggests (confer [Unb00], section 4.4.2) to approximate the transition function by theoretical considerations or deterministic test-signals to figure out T_{63}, the time when the output has reached 63% of its maximal value. A sampling period between $\Delta T = 1/6\ T_{63}$ and $\Delta T = 1/10\ T_{63}$ is proposed.

At last the *run-time* has to be discussed. In chapter 8 it is shown that run-time can be reduced by decreasing the number of time-slices. For hybrid Bayesian networks the number of mixture

components is proportional to $n_{\mathrm{s}}^{t_{\max}}$. That is the reduction of the number of time-slices is an effective means to reduce run-time. But as time-measurements in chapter 5 show, solely cutting down the number of time-slices is not sufficient. For discrete dynamic Bayesian networks a possible approach is discussed in [BK98a; BK98b]. A more promising approach might be moment matching. The idea is to reduce the number of components by approximating a mixture of Gaussians by a mixture of Gaussians with a lower number of mixture components. This seems to be promising particularly for the state-space model introduced in section 5.2. In the dynamic Bayesian model depicted in figure 5.16 the discrete nodes represent different operating points. Usually one state is much more probable than the remaining states. When the system changes from one operating point to the next only two states are more likely than the others. Thus it is expected that the reduction of mixture components results only in a low error. A similar principle is applied in the modeling of possible failures within a system of 5 tubes [KL00]. The main idea is that the states representing none or only one error are more probable than the states representing the occurrence of multiple errors at the same time. That means that the idea that some states are unlikely is used to reduce the number of mixture components.

Four different problems

- Usage of real training signals.
- Identification of the structure of the system.
- Identification of the sampling rate
- Real-time inference

are discussed which might cause problems when transferring the theoretical approach in this thesis to a practical application. For all of them possible approaches to solve them are discussed, therefore it is worthwhile to continue research in the domain of Bayesian control.

Improvements of the system As test systems only SISO systems were employed. But in reality also a lot of MIMO systems have to be controlled. Schulz [Sch02] discusses as example a helicopter, a hydraulic cascade, a distillation column, and a steam generator. That is *MIMO systems* are an important extension. At least the state-space approach is also used for MIMO systems, but in reality some problems can occur. One possible source of problems is the inference algorithm. Using the inference algorithm introduced in [Lau92] might result in numerical problems. The matrix $\boldsymbol{K}$ occurring in the canonical characteristics is initialized, so that it has rank one ("Note that $\boldsymbol{K}$ has rank one and is therefore typically not positive definite [Lau92]"). But when the canonical characteristics is transformed to moment characteristics an inversion of

$\boldsymbol{K}$ is required. Furthermore the calculation of marginal potentials is only guaranteed to be finite if the according sub-matrix of $\boldsymbol{K}$ is positive definite. Thus numerical problems are expected. The implementation of the recent inference algorithm [LJ99] in the BN-toolbox is slower than the implementation of the former one. Thus it is advisable to use the former algorithm as long as it provides numerical stable results. The second problem is the training process. For the state-space model some weights are clamped; i.e, they are excluded from training. As a result the search-space is narrowed which leads to a faster and more stable training. For MIMO systems the normal form which is used to restrict the search-space can no longer be used. Thus it has to be checked whether the normal forms available for MIMO systems [Sch02] can be used in a similar way.

The application of normal forms is one way of introducing a-priori knowledge into the model. A second source of information is an analytical description of the plant. But usually a complete analytical description of the system to be controlled is not given. Therefore an automatic way to include partial or qualitative [Kui94] knowledge about the system is needed. Possible approaches are:

- Imprecise or incomplete knowledge might be used for a better initialization and a refined training process. For example the weight of a link $Y_i \rightarrow Z$ might be known exactly. But in the current implementation of the BN-toolbox it is only possible to clamp a complete weight-vector of a node Z. That is the weights of all links $Y \rightarrow Z$ must be known.

- Fixing a set of parameters, e.g. variance or weights of some nodes.

- When structure learning is applied a-priori knowledge might be used to include or exclude some edges from the search process or to provide a node ordering for the learning process.

- For overdamped systems the number of nodes included in the calculation of u_{new} can be reduced.

A main point of control is *stability* of the controlled system. For linear SISO systems good results are obtained by using a difference equation model. But for the state-space approach the training failed in one case, i.e. convergence was not achieved (see table 4.4). As the suggested model for nonlinear systems are based on a state-space model a guarantee for convergence is still of importance, particularly for the nonlinear case. Similarly a guarantee is missing that the steady state error is reduced to zero.

Alternative approaches There are other means for stochastic inference beside Bayesian networks. Well known inference algorithms are different *sampling* procedures, e.g. Gibbs sampling.

Gibbs sampling is implemented for example in BUGS. In [DDN00b] it is shown that BUGS is also suitable for modeling of preforming and calibration. There are three advantages in comparison to the inference process used for the Bayesian network.

- The inference process is not restricted to conditional Gaussian distributions.
- Usage of deterministic nodes which are helpful to model nonlinearities.
- Run-time can be reduced by reducing the number of iterations.

The last item raises the question how many iterations are necessary before convergence is reached. Brooks and Roberts [BR97] write: "Ideally we would like to analytically compute or estimate a convergence rate and then take sufficient iterations for any particular desired accuracy but this is not possible in general. In fact for Markov chains it is extremely difficult to prove even the existence of a geometric rate of convergence to stationarity."

Closer to Bayesian networks are *decision networks*, introduced e.g. in [CDLS99]. In decision networks two types of nodes are added. First decision nodes that have no parents are added. The decision nodes have an influence on the utility nodes which represent the utility of each state; i.e., the utility nodes depend on the states and on the decision nodes. In [CDLS99] the decision nodes are defined to be discrete. Before the approach can be applied to control problems an extension to continuous decision nodes is necessary.

9.1.2 Modeling manufacturing processes

There remain several challenges when modeling manufacturing processes. The main problem is the usage of *test-plans*. The problem is clearly identified in section 7.1.3. However, the suggested solution is only applicable when a fully factorized test-plan is employed to identify the main influence factors. That means that all configurations of X_i, X_j have to be examined if the combined influence of $X_i X_j$ is considered important. It does not mean that all possible configurations of all variables have to be examined.

Section 7.2 shows that the idea to identify subsets of variables, where all configurations are observed, is not applicable to all test plans. For injection moulding the restrictions will be taken into account during the collection of new data. But it remains subject of research whether there is a general mapping from each test-plan to a structure of a Bayesian network.

The principle of *piecewise approximation* is mostly applied in the approximation of one-dimensional functions. But, as there are also Taylor series for multi-dimensional functions, it is

likely that an extension for the multi-dimensional case exists. To apply the suggested approximation to multi-dimensional functions an algorithm is needed to figure out suitable points u_a, so that the expansion of a function f at a point u_a can be calculated. For the one-dimensional case visualization is applied. This is no appropriate solution for the multi-dimensional case. The second problem is to calculate Taylor series at hidden states x^s. This would have a large impact on hybrid dynamic Bayesian networks used for control. At the moment the selected operating point depends only on the input. The selection of the operating point depending on the state would cover a wider range of nonlinearities.

9.2 Summary

This thesis deals with the problem of modeling and control of static and dynamic systems. Thus it is situated at the intersection of control theory, manufacturing, and stochastic modeling. It is therefore necessary to give a brief introduction to all domains.

In chapter 2 Bayesian networks that are selected as means for stochastic modeling are introduced. This chapter covers the main aspects of discrete, hybrid, and dynamic Bayesian networks, including definitions, inference algorithms, and training.

Chapter 3 gives an overview about control theory. Main points are the description of linear, dynamic systems by difference equations and the state-space description. Later on, in chapter 4, they are used to deduce the structure of a Bayesian network. Even if linear systems offer a broad applicability, also nonlinear systems have to be discussed. Typical nonlinearities like saturation, hysteresis curve, dead-zone and the two-point element that are introduced in several books about nonlinear control [LW00; Föl93; Unb97b] are mentioned. The aim is to develop prototypical models for frequently occurring nonlinear units which can be combined to more complex units.

At the end of chapter 3 two traditional controllers, namely the approach by Ziegler and Nichols, and the Dead-Beat controller are introduced. They are used to compare the new approach of a Bayesian controller to controllers in practical use.

In chapter 4 the *Bayesian controller* is introduced. As a starting point, both the state-space description and the difference equation model that are regularly used in control theory are mapped to dynamic Bayesian networks. These models are also employed for testing the new approach. The usage of general models has the following advantages

- The approach is independent from the intended application. Even if some domain-knowledge is used, the models can easily be transferred to other applications.

- The approach can thoroughly be tested by simulation. This reduces the effort for modeling as there is no need to start expensive plants. Moreover the risk of failure is reduced when the solutions are transfered to the final application.

- Knowledge from control theory can easily be incorporated into the Bayesian model.

In control theory dynamic systems are described by differential equations. One mathematical model is obtained by rewriting the differential equation to n differential equations of first order. This model is called the *state-space description*, the state of the system is represented by hidden state nodes. The differential equations are transformed to an equivalent description in discrete time. The obtained model is similar to a Kalman filter; i.e., the state $\boldsymbol{x}_{t+1}^s$ depends on the former state $\boldsymbol{x}_t^s$ and on the input u_t. The output depends on the state, and in special cases also on the input. The state-space model is mapped to a structure of a dynamic Bayesian network. The similarity to the Kalman filter is used to deduce the weights and means of the dynamic Bayesian network (see section 3.1.2 for Kalman filters and section 4.1 which discusses the relation between Kalman filters and dynamic Bayesian networks).

To get an efficient controller, the first time-slices are used for the representation of the past; i.e., the evidence for the first time-slices comprises in- and output. This information is used to estimate the disturbance as the difference between the model and the observation. In the state-space model additionally the state of the system is estimated.

For the future no information but the desired value is given. The desired value is entered as evidence and afterwards the manipulated value is calculated by marginalization over all variables except the required input. The estimated disturbance is included in the calculation as its estimation is propagated from the past to the future. This propagation is based on the assumption that the characteristic of the disturbance changes slowly. To guarantee a good performance a special relation of the covariances is essentially.

- The variances of the input nodes are set to a large value. This setting reflects the assumption that the controller must respond to a changed disturbance or desired value by changing the input.

- The variance of the nodes representing the disturbance is below the variance of the input and greater or equal to the variance of the output nodes. Particularly the variance of the disturbance node in the first time slice is greater than the variance of the output nodes. This setting enables the estimation or the disturbance when input and output are given.

- The variance of the output is set to a small value, because it is assumed that the model is

correct. Deviations between the model and the observation are explained by an adequate estimation of the disturbance.

The relation between the variances is fixed, that is the variances are not adapted during training.

In a first step (see section 4.1) the *feasibility of a Bayesian controller* is tested. To judge the performance of a controller the sum of the squared error, the overshoot, the steady state error and the settling time is used (compare table 4.2).

The feasibility test shows that the desired value is reached in all cases with a deviation smaller or equal than 2 ‰ of the desired value within a settling time of smaller than 0.7 s (Test systems with a natural angular frequency of 10 s are used).

To come to a self adaptive system *training* is necessary. It is performed with the EM-algorithm which is included in the BN-toolbox. The EM-algorithm used for training only guarantees that a local maximum of the log-likelihood is reached. To obtain a large accuracy of the model several measures are taken concerning the training data:

- The time-series used for training are exchanged after five iterations.
- Time-series of different type (step-response, impulse response and sine-response) and frequencies are applied.

In order to obtain an optimal training result the search-space for the state-space model is reduced by the usage of the *observable canonical form* which allows to fix the weights from the state nodes $\boldsymbol{X}^s$ to the output node Q. As also the weight from the disturbance node Z^d to the output Q is fixed, the node Q is excluded from training. The parameters of the disturbance node are also fixed, because it is assumed that the characteristic of the disturbance changes slightly from one time-slice to the next. Thus two layers are excluded from training.

The overshoot of the trained controller is greater than for the analytical controller. This leads to a smaller squared error for the reference reaction. However, the less accurate model leads to a larger squared error $I_{\mathrm{d}}(z^d = 1)$ for the disturbance reaction and to a greater settling time t_s. The steady state error remains with three (of 30) exceptions below the 1%-level. The most critical result is the missing convergence in one case.

This severe drawback is remedied by the *difference equation* model. It is obtained by transforming the differential equation to a difference equation. This is done by approximation of the derivatives by differences of function values. The state of the system is represented by regress to former function values; i.e., the calculation of y^m_{t+1} is based on information about y^m_{t-i} and u_{t-i}. Thus the usage of difference equations leads to a higher order Markov model. For the implementation of higher order Markov models two different methods are suggested:

- Expansion of the BN-toolbox, so that also higher order Markov-models can be unrolled to usual Bayesian networks.
- Adding redundant nodes to the model, so that former inputs and outputs are represented in two different time-slices.

For the difference equation model it is supposed that there is no disturbance during training. Thus $q_t = y_t^m$, the observed output is entered twice as evidence. Thus no hidden nodes are left which leads to a stable training result. The steady state error e_∞ is in all cases smaller or equal than 7‰ of the desired value and convergence is achieved in all cases. In comparison to the state-space system the number of training-iterations is reduced from 20 to 5, but convergence of the log-likelihood can be observed after the second iteration.

Chapter 4 is restricted to linear systems. This restriction is eliminated by the usage of *hybrid Bayesian networks* as discussed in chapter 5. The idea is to approximate a *nonlinear function* by multiple Taylor series. When the model is employed a discrete node selects the suitable Taylor series. For test purposes the model is applied to the saturation, and to the hysteresis curve. The saturation is modeled with a relative error smaller than 3%, the hysteresis is also modeled with high accuracy, but a training of the hysteresis model is not possible.

The linear approximation is combined with linear dynamic models, both with the difference equation model and the state-space model. For nonlinear systems the difference equation model provides no acceptable results, particularly the training is unstable. The combination of the saturation model with the state-space model leads to acceptable results, the steady state error is between 5 and 6‰ of the desired value. But at the moment a practical employment is not possible due to the large evaluation time of approximately 40 s to calculate the new input signal.

Most of the *manufacturing processes* are nonlinear. Chapter 7 discusses the application of Bayesian networks to several subprocesses of hydroforming, e.g. *preforming* and *calibration*. For preforming the pressure between the blanks is predicted with a relative error smaller than 3%, for the calibration the relative error is between 30 and 55%, but the predicted curve is close to the original data, the error is largest at the bursting point. Another important point is that both models show generalizability. That is they are able to make predictions for yet unknown data, a feature which is usually associated with neural networks. But not all of the presented models offer generalizability.

The counter example is the model for injection moulding. The analysis of the data shows that only 5% of the possible configurations are tested. This low number of examples leads to the fact that not all configurations of discrete parents are observed. Thus there are parameters in the conditional probability tables which cannot be trained, as there are no examples. Literature about

quality management points out that the missing configurations are due to deliberate test-plans. For simple test-plans subsets of variables, where all configurations are tested, can be identified and used as possible parents. Following this instruction a model for the *welding process* is developed (see section 7.1.3 and [DDNK03]). This model is also able to make predictions for unknown inputs. Thus the aim of generalizability is reached in almost all cases. The only exception is the model for *injection moulding* which is therefore not suited to deduce a suitable operating point for injection moulding. The reason for the failure is that there is no set of two variables where all configurations are observed.

The thesis finishes with a discussion about *real-time*. The number of time-slices can be severely reduced which leads both to a reduction of the training and the evaluation time. For models of second order the number of time-slices used for the future can be reduced to two. In one time-slice no evidence is given for the output node, the second time-slice is used to enter the desired value as evidence. There is nearly no impact on the steady state error and on the settling time. Also the number or time-slices used for the representation of the past can be reduced. The minimal number depends on the order of the system to be modeled. But it seems from advantage to use one or two additional time-slices to reduce the steady state error.

As discussed in section 9.1, there are a lot of problems to be solved, but the thesis offers a stable base for Bayesian control. At the beginning the Bayesian controller might be used for slow processes with no available mathematical model, so that there is a need for a self-adaptive controller with no hard real-time requirements.

Bibliography

[AJA+89] S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. Sørensen, A. Rosenfalck, and F. Jensen. MUNIN - An expert EMG assistant. In J. E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, chapter 21, pages 255 – 277. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 89.

[BH96] J. S. Breese and D. Heckerman. Decision-Theoretic Troubleshooting: A Framework for Repair and Experiment. Technical Report MSR-TR-96-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, 1996.

[BK98a] X. Boyen and D. Koller. Approximate learning of dynamic models. In *Proc. of the 11th Annual Conference on Neural Information Processing Systems, Denver, Colorado*, pages 396 – 402, December 1998.

[BK98b] X. Boyen and D. Koller. Tractable Inference for Complex Stochastic Processes. Technical report, Stanford University, Computer Science Department, March 1998.

[BR97] S. Brooks and G. Roberts. Assessing Convergence of Markov Chain Monte Carlo Algorithms. *Statistics and Computing*, 8(4):319 – 335, 1997.

[Bre69] L. Breiman. *Probability and stochastic processes: With a view towards Applications*. Houghton Mifflin Company, Boston, 1969.

[Bre73] L. Breiman. *Statistics: With a view toward application*. Houghton Mifflin Company, Boston, 1973.

[Bun94] W. L. Buntine. Learning with Graphical Models. Technical report, Artificial Intelligence Research Branch at NASA Ames, 1994.

[CAC95] *Communications of the ACM*, 38(3), March 1995.

[CDLS99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer, New York, 1999.

[CH92] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9:309 – 347, 1992.

[CM98] G. F. Cooper and S. Moral, editors. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, July 1998.

[DDN00a] R. Deventer, J. Denzler, and H. Niemann. Control of Dynamic Systems Using Bayesian Networks. In L. N. de Barros et al., editors, *Proceedings of the IBERAMIA/SBIA 2000 Workshops (Atibaia, São Paulo, Brazil)*, pages 33 – 39, São Paulo, November 2000. Tec Art Editora. ISBN: 85-901664-1-4.

[DDN00b] R. Deventer, J. Denzler, and H. Niemann. Non-linear modeling of a production process by hybrid Bayesian Networks. In Werner Horn, editor, *ECAI 2000 (Berlin)*, pages 576 – 580. IOS Press, August 2000.

[DDN02a] R. Deventer, J. Denzler, and H. Niemann. Application of Bayesian Controllers to Dynamic Systems. In A. Abraham and M. Koeppen, editors, *Hybrid Information Systems, Proceedings of the First International Workshop in Hybrid Intelligent Systems, HIS 2001*, pages 555 – 569, Heidelberg, 2002. Physica.

[DDN02b] R. Deventer, J. Denzler, and H. Niemann. Using Non-Markov models for the Control of Dynamic Systems. In *Engineering of Intelligent Systems (EIS)*, pages 70 (complete paper on CD–ROM). ICSC-NAISO Academic Press, September 2002.

[DDN03a] R. Deventer, J. Denzler, and H. Niemann. Bayesian Control of Dynamic Systems (to be published). In A. Abraham and L. C. Jain, editors, *Recent Advances in Intelligent Paradigms*, chapter 3. Springer Verlag, 2003.

[DDN03b] R. Deventer, J. Denzler, and H. Niemann. Bayesian controller versus traditional controllers. In M. Mohammadian, editor, *International Conference on Computational Intelligence For Modelling Control & Automation*, 2003.

[DDNK03] R. Deventer, J. Denzler, H. Niemann, and O. Kreis. Using Test Plans for Bayesian Modeling. In P. Perner and A. Rosenfeld, editors, *Machine Learning and Data Mining in Pattern Recognition*, pages 307 – 316, Berlin, 2003.

[Dev94] R. Deventer. Extraktion und Simplifikation von Regeln aus Boltzmannmaschinen. Master's thesis, Universität Kaiserslautern, April 1994.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1 – 38, 1977.

[DN01] R. Deventer and H. Niemann. Arbeits- und Ergebnisbericht SFB 396, 1999 – 2001, C1 Prozessmodellierung. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2001.

[EAD+03] G. W. Ehrenstein, S. Amesöder, L. Fernández Diaz, H. Niemann, and R. Deventer. Werkstoff- und prozessoptimierte Herstellung flächiger Kunststoff-Kunststoff und Kunststoff-Metall-Verbundbauteile. In M. Geiger and G. W. Ehrenstein, editors, *DFG Sonderforschungsbereich 396 Berichts- und Industriekolloqium 15./16. Oktober 2003*, pages 149 – 178, Bamberg, 2003. Meisenbach.

[EZ98] G. W. Ehrenstein and G. Zhao. Arbeits- und Ergebnisbericht SFB 396, 1996 – 1998, B3 Prozessoptimierung. Friedrich-Alexander-Universität Erlangen-Nürnberg, 1998.

[FJdS99] A. Filippidis, L. C. Jain, and C. W. de Silva. Intelligent Control Techniques. In Jain and de Silva [JdS99], pages 1 – 24.

[FK03] N. Friedman and D. Koller. Being Bayesian about Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning Journal*, 50:95 – 126, 2003.

[FMR98] N. Friedman, K. Murphy, and S. Russel. Learning the Structure of Dynamic Probabilistic Networks. In G.F. Cooper and S. Moral, editors, *Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 139 – 147, San Francisco, 1998. Morgan Kaufmann.

[Föl93] O. Föllinger. *Nichtlineare Regelungen I.* Oldenbourg, München,Wien, 1993.

[Gel94] A. Gelb, editor. *Applied optimal estimation*. MIT Press, Cambridge, Massacusetts, USA, 1994.

[GH94] D. Geiger and D. Heckerman. Learning Gaussian Networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 235 – 243, San Francisco, 1994. Morgan Kaufmann.

[GL99] E. L. Grant and R. S. Leavenworth. *Statistical Quality Control*. McGraw-Hill, Boston, 1999.

[Had99] P. Haddawy. An Overview of Some Recent Developments in Bayesian Problem Solving. *AI Magazine, Special Issue on Uncertainty in AI*, 1999.

[HB95] E. Horvitz and M. Barry. Display of information for Time-Critical Decision Making. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 296 – 305, 1995.

[Hec97] D. Heckerman. Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery*, pages 79 – 119, 1997.

[HG95] D. Heckerman and D. Geiger. Learning Bayesian Networks. Technical Report MSR-TR-95-02, Microsoft Research, February 1995.

[HGC95] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197 – 243, 1995.

[HH98] D. Heckerman and E. Horvitz. Inferring Informational Goals from Free-Text Queries. In Cooper and Moral [CM98], pages 230 – 237.

[HHN91] D. Heckerman, E. J. Horvitz, and B. Nathwani. Toward normative expert systems I: The pathfinder project. *Methods of Information in Medicine*, 31:90 – 105, 1991.

[HJBHR98] E. Horvitz, D. Heckerman J. Breese, D. Hovel, and K. Rommelse. The Lumière Project: Bayesian User Modeling Inferring the Goals and Needs of Software User. In Cooper and Moral [CM98], pages 256 – 265.

[HMW95] D. Heckerman, A. Mamdani, and M. P. Wellman. Real-World Applications of Bayesian networks. *Communications of the ACM*, 38(3):24 – 26, March 1995.

[HR91] H. Hetzheim and G. Rommel. Fuzzy Logic für die Automatisierungstechnik? *Automatisierungstechnische Praxis*, 33(10), 1991.

[HRSB92] E. Horvitz, C. Ruokangas, S. Srinivas, and M. Barry. A decision theoretic approach to the display of information for time-critical decisions: The Vista project. In *Proceedings of the Conference on Space Operations and Automation and Research*, January 1992.

[IB98] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29:5 – 28, 1998.

[JdS99] L. C. Jain and C. W. de Silva, editors. *Intelligent Adaptive Control, Industrial Applications*. CRC Press, Boca Raton, 1999.

[Jen96] F. V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

[Jen01] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Statistics for Engineering and Information Science. Springer, Berlin, Heidelberg, 2001.

[JJD94] F. Jensen, F. V. Jensen, and S. L. Dittmer. From Influence Diagrams to Junction Trees. In *Tenth Conference on Uncertainty in Artificial Intelligence*, pages 367 – 373, San Francisco, 1994. Morgan Kaufmann.

[Jor99] M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, Massachusetts, 1999.

[Kit02] E. Kitzelmann. Bayes‘sche Netze, Grundlagen und Anwendungen, Juli 2002. Seminar Kommunikation und Sicherheit in komplexen soziotechnischen Systemen SS02, Uni Berlin.

[Kjæ90] U. Kjærulff. Triangulation of Graphs - Algorithms Giving Small Total State Space. Technical Report R 90-09, Aalborg University, Institute of Electronic Systems, Judex Datesystemer A/S, DK-9000 Aalborg, Denmark, March 1990.

[Kjæ92] U. Kjærulff. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference of Uncertainty in Artificial Intelligence*, pages 121 – 129. Morgan Kaufmann Publishers, San Mateo, California, 1992.

[Kjæ93] U. Kjærulff. *Aspects of Efficiency Improvement in Bayesian Networks*. PhD thesis, Aalborg University, Institute of Electronic Systems, 1993.

[KL00] D. Koller and U. Lerner. Sampling in Factored Dynamic Systems. In A. Doucet, J.F.G. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 21, pages 445 – 464. Springer, 2000.

[KN98] B. Kahles and H. Niemann. Arbeits- und Ergebnisbericht SFB 396, 1996 – 1998, C1 Prozessmodellierung. Friedrich-Alexander-Universität Erlangen-Nürnberg, 1998.

[Kre02] O. Kreis. *Integrierte Fertigung - Verfahrensintegration durch Innenhochdruck-Umformen, Trennen und Laserstrahlschweißen in einem Werkzeug sowie ihre tele- und multimediale Präsentation*. PhD thesis, Universität Erlangen-Nürnberg, Meisenbach, Bamberg, 2002. http://www.integrierte-fertigung.de.

[Kui94] B. Kuiper. *Qualitative Reasoning*. MIT Press, 1994.

[Lan96] R. Langmann. *Prozeßlenkung, Grundlagen zur Automatisierung technischer Prozesse*. Vieweg, Braunschweig, 1996.

[Lau92] S. L. Lauritzen. Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models. *Journal of the American Statistical Association*, Vol. 87(420):1098 – 1108, December 1992.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[Lau01] S. L. Lauritzen. Some Modern Applications of Graphical Models. Technical Report Research Report R-01-2015, Department of Mathematical Sciences, Aalborg University, 2001.

[Lei89] H.-G. Leimer. Triangulated Graphs With Marked Vertices. *Graph Theory in memory of G. A. Dirac, Annals of Discrete Mathematics*, 41:311 – 324, 1989.

[LJ99] S. L. Lauritzen and F. Jensen. Stable Local Computation with Conditional Gaussian Distributions. Technical Report R-99-2014, Aalborg University, Department of Mathematical Sciences, September 1999.

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. Local Computation With Probabilities on Graphical Structure and Their Application To Expert Systems. *Journal of the Royal Statistical Society*, pages 157 – 224, 1988.

[LSK01] U. Lerner, E. Segal, and D. Koller. Exact Inference in Networks with Discrete Children of Continuous Parents. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 319 – 328, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[Lu96] Y.-Z. Lu. *Industrial Intelligent Control, Fundamentals and Applications*. John Wiley & Sons, Chichester, 1996.

[LW00] H. Lutz and W. Wendt. *Taschenbuch der Regelungstechnik*. Verlag Harri Deutsch, Frankfurt am Main, 3. edition, 2000.

[Mar94] W. Marsing, editor. *Handbuch Qualitätsmanagement*. Carl Hanser Verlag, München, Wien, 1994.

[MLP99] D. W. McMichael, L. Liu, and H. Pan. Estimating The Parameters Of Mixed Bayesian Networks From Incomplete Data. In R. Evans, L. White, D. McMichael, and L. Sciacca, editors, *Proceedings of Information Decision and Control 99*, pages 591 – 596, Adelaide, Australia, February 1999. Institute of Electrical and Electronic Engineers, Inc.

[Mur98a] K. P. Murphy. Fitting a Conditional Gaussian Distribution. Technical report, University of California, Computer Science Division (EECS), October 1998.

[Mur98b] K. P. Murphy. Inference and Learning in Hybrid Bayesian Networks. Technical Report CSD-98-990, University of California, Computer Science Division (EECS), January 1998.

[Mur99] K. P. Murphy. A Variational Approximation for Bayesian Networks with Discrete and Continuous Latent Variables. In Kathryn Blackmond Laskey and Henri Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 457 – 466, San Francisco, 1999. Morgan Kaufmann Publishers Inc.

[Mur02] K. P. Murphy. *Dynamic Bayesian networks: Representation, Inference and Lerning*. PhD thesis, University of California, Berkeley, 2002.

[Nom99] T. Nomura. Applications of Evolutionary Algorithms to Control and Design. In Jain and de Silva [JdS99], chapter 3, pages 41 – 62.

[Ole93] K. G. Olesen. Causal probabilistic networks with both discrete and continuous variables. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 3(15):275 –279, 1993.

[Pan98] C. Panknin. A Probabilistic, Knowledge Based Approach to Detection and Tracking of Non-Rigid Objects. Master's thesis, Friedrich-Alexander-Universität, Erlangen, August 1998.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1988. xISBN 0-934613-73-7.

[Pfe93] T. Pfeifer. *Qualitätsmanagement – Strategien, Methoden, Techniken*. Hanser, München, Wien, 1993.

[Pfe98] T. Pfeifer. *Fertigungsmeßtechnik*. Oldenburg, München, 1998.

[Pre92] H.-P. Preuß. Fuzzy Control – heuristische Regelung mittels unscharfer Logik. *Automatisierungstechnische Praxis*, 34(4), 1992.

[Rin97] H. Rinne. *Taschenbuch der Statistik*. Verlag Harri Deutsch, 1997.

[RM86] D. E. Rummelhardt and J. L. McClelland. *Parallel distributed processing, explorations in the Microstructure of Cognition*, volume 1. The Massachussets Institute of Technology, 1986.

[RS97] M. Ramoni and P. Sebastiani. Parameter Estimation in Bayesian Networks from Incomplete Databases. Technical report, Knowledge Media Institute, The Open University, 1997.

[Sch02] G. Schulz. *Regelungstechnik, Mehrgrößenregelung – Digitale Regelungstechnik – Fuzzy Regelung*. Oldenburg Verlag, München, Wien, 2002.

[sfb95] SFB 396, ,,Robuste, verkürzte Prozessketten für flächige Leichtbauteile“, Finanzierungsantrag 1996 – 98, 1995.

[SGS01] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, Cambridge, Massachusetts; London, England, 2001.

[She97] P. P. Shenoy. Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, 17(2-3):239 – 263, 1997.

[SJK00] C. Skaaning, F. V. Jensen, and U. Kjærulff. Printer troubleshooting using Bayesian networks. In *Proceedings of the 13th International Conference on Industrial and Engineering Applications of AI and Expert Systems*, pages 367 – 379, 2000.

[SL91] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, London, 1991.

[SMH+91] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. J. Horvitz, and H. Lehmann. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I: The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30:241 – 255, 1991.

[ST95] E. G. Schukat-Talamazzini. *Automatische Spracherkennung*. Vieweg, 1995.

[STBG96a] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. *BUGS 0.5 Bayesian inference using Gibbs Sampling*. MRC Biostatistics Unit, Institute of Public Health, Robinson Way, Cambridge CB2 2SR, 1996.

[STBG96b] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. *BUGS 0.5 Examples Volume 1*. MRC Biostatistics Unit, Institute of Public Health, Robinson Way, Cambridge CB2 2SR, 1996.

[STBG96c] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. *BUGS 0.5 Examples Volume 2*. MRC Biostatistics Unit, Institute of Public Health, Robinson Way, Cambridge CB2 2SR, 1996.

[TL98] S. Thrun and J. Langford. Monte Carlo Hidden Markov Models. Technical Report CMU-CS-98-179, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1998.

[Unb97a] H. Unbehauen. *Regelungstechnik I*. Vieweg, Braunschweig, Wiesbaden, 1997.

[Unb97b] H. Unbehauen. *Regelungstechnik II*. Vieweg, Braunschweig, Wiesbaden, 1997.

[Unb00] H. Unbehauen. *Regelungstechnik III*. Vieweg, Braunschweig, Wiesbaden, 2000.

[WBS+01] A. Weckenmann, V. Bettin, R. Stöber, H. Niemann, and R. Deventer. Modellierungsverfahren zur Optimierung und Regelung verkürzter Prozessketten. In Georg Redeker, editor, *Qualitätsmanagement für die Zukunft – Business Excellence als Ziel*, pages 109 – 124, Aachen, 2001. Shaker Verlag. ISBN: 3-8265-8461-9.

[WS99] R. L. Welch and C. Smith. Bayesian control for concentrating mixed nuclear waste. In *Proceddings of the 15th conference of uncertainty in artificial intelligence*, pages 663 – 669, 1999.

[WSdS99] Q. M. J. Wu, K. Stanley, and C. W. de Silva. Neural control systems and applications. In Jain and de Silva [JdS99], chapter 4, pages 63 – 103.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and control*, 8:338 – 353, 1965.

[ZE98] G. Zhao and G. W. Ehrenstein. Arbeits- und Ergebnisbericht SFB 396, 1996 – 1998, B3 Prozeßoptimierte Herstellung von flächigen Kunststoff-Metall-Hybridstrukturen. Friedrich-Alexander-Universität Erlangen-Nürnberg, 1998.

[Zha98a] N. L. Zhang. Probabilistic Inference in Influence Diagrams. *Computational Intelligence*, 14(4):475 – 497, 1998.

[Zha98b] N. L. Zhang. Probabilistic Inference in Influence Diagrams. In Cooper and Moral [CM98], pages 514 – 522.

[ZN42] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Trans. ASME*, 64:759 – 768, 1942.

Appendix

Used mathematical symbols for Bayesian networks

Symbol	Description
$\mathbb{C}$	A clique, i.e. a set of random variables used as a node in a junction tree.
$\mathbb{C}_\mathrm{J}$	A set of cliques
$d_{\boldsymbol{x}}$	Expected probability for configuration $\boldsymbol{x}$
$\det(\boldsymbol{M})$	Determinant of a matrix $\boldsymbol{M}$
$Dom(X_i)$	Domain of the random variable X_i
$E[Y]$	Expectation of a random variable Y
$\boldsymbol{E}_\mathcal{G}$	Denotes the edges in a graph $\mathcal{G}$
e	Evidence entered in a Bayesian network
F	Force, variable used e.g. for the tensile strength
$\mathbb{F}(X)$	Family of the random variable X, $\mathbb{F}(X) = \{X\} \cup \mathbb{P}(X)$
$\mathbb{f}(X)$	Instantiation of $\mathbb{F}(X)$
$\mathcal{G}$	Graph, not necessarily a DAG
$g(\boldsymbol{x})$	Parameter of the canonical representation, depending on the configuration $\boldsymbol{x}$
$\boldsymbol{h}(\boldsymbol{x})$	Vector, parameter of the canonical representation, depending on the configuration $\boldsymbol{x}$
J_t	Junction tree for time-slice t in a DBN
$\boldsymbol{K}(\boldsymbol{x})$	Matrix, parameter of the canonical representation, depending on the configuration $\boldsymbol{x}$.
L'	Likelihood
L	Log-likelihood
$\boldsymbol{M}^T$	Transpose of the matrix $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	Inverse of matrix $\boldsymbol{M}$

Symbol	Description
$n_c(\boldsymbol{x})$	Denotes how often a configuration $\boldsymbol{x}$ is observed
$\widetilde{n_c}(\boldsymbol{x})$	Expectation, how often a configuration $\boldsymbol{x}$ is observed
n_N	Number of nodes in a Bayesian network
n_{past}	Number of time-slices used for the representation of the past
n_{future}	Number of time-slices used for the representation of the future
n_{s}	Number of states per time-slice in a Dynamic Bayesian network
N	Number of training examples
n_{ts}	Number of nodes in a time-slice
$\mathcal{N}$	Normal distribution
P	Probability (discrete random variables)
p	Probability distribution (continuous random variables)
$\mathbb{P}(X)$	Parents of a node for random variable X
$\mathbb{p}(X)$	Instantiation of the parents of random variable X
$\boldsymbol{R}$	Strong root in a junction tree
$S_{ZY,\boldsymbol{x}}$	ESS for the product of random variables Z and Y given the configuration $\boldsymbol{x}$ and the observations.
$\mathbb{S}$	Denotes a separator, i.e. a set of random variables used as a node in a junction tree. In analogy to the cliques $\mathcal{S}$ denotes a set of separators.
$t_{\max}$	Number of time slices in a DBN
T_{inf}	Inference time for a Bayesian network
$\boldsymbol{V}_{\mathcal{G}}$	Vertices of a graph $\mathcal{G}$
X, x	X denotes a discrete random variable, x its instantiation
$\boldsymbol{X}, \boldsymbol{x}$	Vector of discrete random variables
$\boldsymbol{X}^j$	j-th training example
Y, y, Z, z	Continuous random variables together with their instantiation
$\boldsymbol{\alpha}$	Parameter of a continuous node in a Bayesian network
$\boldsymbol{\beta}$	Weight of the potential definition in [LJ99]
$\boldsymbol{\Gamma}$	Covariance matrix, used in the definition of a continuous node in a Bayesian network
$\boldsymbol{\Gamma}_{\mathcal{G}}$	Continuous nodes in a Bayesian network
$\boldsymbol{\Delta}_{\mathcal{G}}$	Discrete nodes in a Bayesian network
$\boldsymbol{\theta}$	Parameters of a distribution, e.g mean and dispersion for a Gaussian distribution
$\boldsymbol{\xi}$	Mean of a potential (moment characteristics)

Symbol	Description
$\boldsymbol{\Sigma}$	Covariance of a potential (moment characteristics)
$\phi(\boldsymbol{X})$	Potential of a set of random variables $\boldsymbol{X}$
$\chi(\boldsymbol{x})$	Indicator function being 1 iff $P(\boldsymbol{x}) > 0$
Ω	Set of outcome of an experiment
ω^r	Result of an experiment
$\dot{\cup}$	Disjoint union
$\mid \boldsymbol{y} \mid$	Length of a vector $\boldsymbol{y}$
$\boldsymbol{X}_1 \perp\!\!\!\perp \boldsymbol{X}_3 \mid \boldsymbol{X}_2$	$\boldsymbol{X}_1$ is conditionally independent from $\boldsymbol{X}_3$ given $\boldsymbol{X}_2$
$\phi_{\mathbb{C}}^{\downarrow\mathbb{S}}$	Marginalization of a potential, here marginalization of the potential of a clique $\mathbb{C}$ to the separator $\mathbb{S}$

Symbols used for control theory

Symbol	Description
a_i^c, a_i	Output-coefficients for a differential (a_i^c) or difference (a_i) equation, describing a linear dynamic system
b_j^c, a_i	Input-coefficients for a differential (b_i^c) or difference (a_i) equation, describing a linear dynamic system
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$	Parameters of the state space model, sometimes used together with the index BN, like $\boldsymbol{A}_{BN}$. In this case these parameters are used to model a time-discrete system.
D	Damping of a dynamic system
$e(t)$	Error, difference between desired value and current value
e_∞	Steady state error
$F_w(s)$	Desired control transfer function
$G(s)$	Laplace transformed transfer function
$G_w(s)$	Control transfer function
I	Designates different quality measures
K	Gain of a transfer unit, e.g. K_s denotes the gain of the dynamic system
m	Maximal deriviation of the input signal in differential equation 3.4
n	Order of the differential or difference equation describing a linear dynamic system
o	Dimension of $\boldsymbol{y}^m$ of a dynamic system

Symbol	Description
$q(t), q_t$	Observed output signal of a dynamic system. $q(t)$ denotes the signal in a continuous time system, q_t signals in a time discrete system
r	Dimension of the input $\boldsymbol{u}$ of a dynamic system
$Re(s)$	Real part of a complex number s
s	Complex number, usually used within a Laplace-transformed transfer function
φ	Transfer unit
T_1, T_2	Time constants to describe a dynamic system of second order
t_s	Settling time
T_a	Rise time
T_d	Dead-time
T_u	Delay time
$u, \boldsymbol{u}$	Input to a dynamic system
u_{new}	Control input calculated from the estimated input for several points in time.
w	Desired value
$\boldsymbol{X}^s$	State in the state-space description
$\check{\boldsymbol{x}}^s$	Operating Point
y^m	Output of the model, to be distinguished from the observed output q
z^d	Disturbance input
δ	Short time duration
ΔT	Sampling period
Δu	Deviation from the operating point
ε	White noise term used to model the disturbance of a system
$\sigma(t)$	Step-function
$\mathcal{Z}$	Z-transformation

Used symbols in process models

Symbol	Description
C	Cushion
F	Force
P_{ihu}	Inner pressure between the blanks
P_{h}	Holding pressure
s_{pl}	Plasticizing stroke
s_{W}	Warpage
τ_{c}	Cavity temperature
$\tau_{\mathrm{c,max}}$	Maximal cavity temperature
τ_{f}	Feed temperature
τ_{m}	Melting temperature
τ_{p}	Preheating temperature
V	Volume
v	Velocity
W_{inj}	Injection work

Used abbreviations

Abbreviation	Explanation
BN	Bayesian network
DAG	Directed acyclic graph
DBN	Dynamic Bayesian network
ESS	Essential Sufficient Statistic
MIMO	Multiple input, multiple output system
MRAC	Model reference adaptive controller
SISO	Single input, single output system
STC	Self tuning controller

Index

Zusammenfassung

Zielsetzung der vorliegenden Dissertation ist es, dynamische Bayesnetze für die Modellierung und Regelung von statischen und dynamischen Prozessen zu verwenden, die hauptsächlich aus der Fertigungstechnologie stammen.

Hierzu werden die Zustandsraumbeschreibung und die Beschreibung dynamischer Systeme durch Differenzengleichungen auf dynamische Bayesnetze abgebildet. Diese Abbildung liefert neben der Struktur des Bayesnetzes auch die Mittelwerte und Gewichte der Knoten. Dadurch erhält man ein Modell, das das gleiche Verhalten wie das dynamische System zeigt.

Um zu einem Regler zu kommen, werden die ersten Zeitscheiben des dynamischen Modells für die Modellierung der Vergangenheit verwendet. Durch die Eingabe ehemaliger Ein- und Ausgaben kann die Störgröße anhand der Abweichung zwischen Ausgabe des Modells und der tatsächlich beobachteten Ausgabe geschätzt werden. Der Sollwert wird als Beobachtung in der Zukunft eingegeben. Durch Marginalisierung wird auf eine mögliche Eingabe geschlossen.

Um zusätzlich Nichlinearitäten zu modellieren, werden diese mit hybriden Bayesnetzen modelliert. Dabei wird die Eingabe gleichzeitig durch einen diskreten und einen kontinuierlichen Knoten modelliert. Die kontinuierlichen Knoten dienen zur Approximation der Nichtlinearität durch eine Taylorreihe. Der diskrete Knoten schaltet zwischen diesen Taylorreihen um. Dadurch wird eine Approximation mit mehreren Taylorreihen gleichzeitig vorgenommen. Dieses Prinzip wird erfolgreich auf die Modellierung der Teilprozesse Vorformen, Kalibrieren und Schweißen des Innenhochdruckumformens und auf den Spritzguss angewandt.

Anschließend wird die Modellierung der Nichtlinearitäten mit der Modellierung dynamischer Systeme kombiniert. Bei der Kombination der Nichtlinearitäten mit der Zustandsraumdarstellung wird fast die gleiche Genauigkeit erzielt, wie bei der Modellierung linearer Systeme. Es bleiben aber Probleme mit der Echtzeitfähigkeit des Systems. Durch Verwendung von weniger Zeitscheiben wird die Laufzeit zwar stark reduziert, bei nichtlinearen, dynamischen Systemen läßt sich damit aber keine Echtzeitfähigkeit erreichen.

Curriculum vitae

15.6.63	Born in Ludwigshafen am Rhein.
8.79 - 7.82	Apprentice as Meß-und Regelmechaniker at BASF AG.
8.82 - 6.85	Final examen, Berufliches Gymnasium (Technical grammar school).
9.85 - 10.86	Civil Service, Red Cross Mutterstadt.
10.86 - 6.94	Study of computer science at University of Kaiserslautern. Diploma-thesis about neural networks with the title 'Extraktion und Simplifikation von Regeln aus Boltzmannmaschinen' ('Extraction and simplification of rules from Boltzmann Machines').
10.94 - 3.96	Siemens AG Mannheim, Engineer.
4.96 - 3.98	Employed as computer scientist by University of Stuttgart.
5.98 - 1.99	Softwareengineer at Zühlke Engineering AG.
2.99 - present	Computer scientist employed by University of Erlangen-Nürnberg.
13.1.2004	Defence of thesis with the title: 'Modeling and Control of Static and Dynamic Systems with Bayesian Networks'.