Application of Bayesian controllers to dynamic systems *

Rainer Deventer[1], Joachim Denzler[1], and Heinrich Niemann[1]

Friedrich Alexander Universität Erlangen Nürnberg, 91058 Erlangen, Germany

Abstract. Bayesian networks for the static as well as for the dynamic case have gained an enormous interest in the research community of machine learning and pattern recognition. Although the parallels between dynamic Bayesian networks and Kalman filters are well-known since many years, Bayesian networks have not been applied to problems in the area of adaptive control of dynamic systems.
In our work we exploit the well-known similarities between Bayesian networks and Kalman filters to model and control linear dynamic systems using dynamic Bayesian networks. The analytical models are compared with models being trained with step and impulse response. The experiments show that the analytical model as well as the trained model are suitable for control purposes, which leads to the idea of self adaptive controllers.
Keywords: Dynamic Bayesian Networks, Kalman filter, Dynamic System, Controller

1 Introduction

Bayesian networks (BN) for the static as well as for the dynamic case have gained an enormous interest in the research community of artificial intelligence, machine learning and pattern recognition. Recently, BN have been applied also to static problems in production, since production processes become more and more complex so that analytical modeling and manual design are too expensive. One example for the successful application of BN to a *static system* in production are the quality evaluation and process parameter selection in order to reach an acceptable quality level [**?**].

Although the parallels between BN and Kalman filters are well-known since many years, BN have not been applied to problems in the area of adaptive control of *dynamic systems*. Adaptive control of dynamic systems is one major problem in production processes. Compared to classical control methods BN have the advantage that the model (of the static or dynamic system) can be trained from examples if the model is not available in analytical form. During training missing information can be handled which makes BN superior to other self adaptive systems like artificial neural networks. Finally, BN can also calculate the most suitable input which leads to a desired output given the information that is entered as evidence in the BN.

* This work was funded by the ,,Deutsche Forschungsgemeinschaft" (DFG) under grant number SFB 396, project-part C1

In this paper it is shown that BN can also act as controller. We exploit the well-known similarities between BN and Kalman filters to model and control linear dynamic systems using dynamic Bayesian networks (DBN). We show, how the model is used to calculate appropriate input signals for the dynamic system to achieve a required output signal. The desired value is entered as evidence. Then, marginalization results in the most likely values of the input nodes.

The performance of the controller is evaluated using the steady state error, the time until the desired value is reached, the integral of the squared error and overshoot. This is done for both the reference and the disturbance reaction of the control loop.

The performance of a controller, inferred from a mathematical model, is compared with a controller, trained with impulse and step responses. The training is done by the well-known EM-algorithm and simplified by normal forms which are used to reduce the number of parameters of the model and thus time complexity and search space during the training.

Both the analytical and the trained model show a good performance, the desired value is reached with low overshoot and the deviation of the actual value from the desired value is below 3% for the trained models.

Even if we focus in the paper on the modeling of stationary, linear dynamic systems of second order, the extension to control nonlinear systems is straight forward using hybrid BN, i. e. a BN using both discrete and continuous nodes. In statistical terms this means that a mixture of Gaussians is used instead of only one normal distribution [**?**].

An additional advantage of our approach is the possibility to learn not only the parameters of a given structure, but both the structure [**?**] and probabilities [**?**] of the Bayesian network from examples.

This article is structured as follows. In Sect. **??** the term control system is defined and the analytical descriptions applied in control theory are introduced. Section **??** deals with Bayesian networks and it is shown how the parameters of the DBN are obtained using the similarities to Kalman filters, a special type of DBN. The usage of BNs to generate control signals is explained in Sect. **??**. The results obtained with this new type of controller are presented in Sect. **??**. The article finishes with a short summary.

2 Dynamic systems and control theory

2.1 The aim of a controller

Humans are regularly encountered with controlling tasks. Typical examples are driving a car. Here the driver has to take care of his speed to avoid being stopped by the police. Additionally the temperature of the heating in the car has to be controlled. A lot of controllers are also found in industry. Generally spoken it is the task of the controller to keep a value $\boldsymbol{q}$, e. g. the speed of a car as close as possible to a desired value $\boldsymbol{w}$, e. g. the maximal speed allowed.

If the environment is changing, e. g. the speed limit is changed or the car is slowed down due to a hill, the controller should change the input so that the difference between the desired value and the current value is reduced as fast as possible.

This section is structured as follows. First the main elements of control loops are introduced together with a block diagram which shows the controller and the dynamic system to be controlled. Afterwards a mathematical description of the dynamic system is given. The used description is taken from control theory, to ensure a broad field of application. Additionally this description is employed to deduce the structure and parameters of a Bayesian network used for control purposes. Using normal forms described in section **??** results in a reduction of the number of parameters to be learned.

The main elements of the driver example discussed at the beginning can be found in Fig. **??**, where all variables are one dimensional. The desired value w is compared with the output q of the system, which means the desired speed is compared with the current speed. The difference between desired and current value e is used as input for the controller which calculates the input for the actuator. The changed input value of the dynamic system results in a new output, where the manipulation reaction describes how the system responds to new inputs.

Additionally the controller has to deal with influences from the environment. The reaction of the controlled system to the disturbing value z' is modeled by the disturbance reaction. As the disturbance value z' usually can not be measured it is easier to deal with the system's reaction z to the disturbance variable directly. From now on z is added to the output y of the system

$$q(t) = y(t) + z(t) \tag{1}$$

and thus regarded as disturbance variable instead. As the block diagram

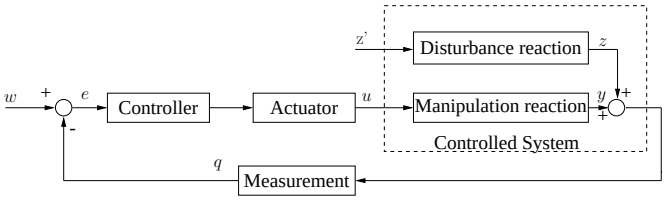

Fig. 1. Block diagram of a controlled system

shows conventional controller uses the error as input. In contrary a model based controller is able to predict the system's reaction and calculate the new input accordingly. It will be shown how a Bayesian model can act as such a controller for a dynamic process and which steps can be taken to simplify the training of it.

To obtain general results a general model widely used in control theory is applied. The usage of this model will not only provide us with a general structure, but will also reduce the number of parameters to be estimated during training.

To judge the quality of a controller a measure is needed. In control theory a lot of measures are known and the selection depends on the application. A metric that is frequently used integrates the squared error

$$Q = \int_0^\infty e^2(t)dt \tag{2}$$

with $e(t)$ as the difference between the desired value $w(t)$ and the actual value $q(t)$. This measure has to be adapted in two ways. First we have only limited time. So the upper limit is changed from infinity to te_{max}, where te_{max} is reached when $e(t)$ is lower than one or three percent of the desired value. If the remaining steady state error is greater than the regarded error limit, te_{max} is reached when the output signal is converged. Convergence is assumed when four positions after decimal point does not change for 20 time slices. As we work in a time discrete space the error is not defined for all t but only at discrete time steps. Thus the error sum

$$Q_d = \sum_{t=0}^{te_{max}} \Delta T e^2(t) \tag{3}$$

is taken as quality measure with ΔT as time between two time slices. Additionally the so called steady state error e_r, i. e. the remaining error when the output signal is converged is used as quality measure.

The next section deals with a mathematical description of the manipulation reaction (cf. Fig. **??**). The aim is a comparison with a Kalman filter and later the identification of a suitable Bayesian network for modeling the behavior of such a system.

2.2 Controlled systems

A dynamic system may be regarded as a black box with several input and output signals $\boldsymbol{u}$ and $\boldsymbol{y}$ respectively, where the output does not depend solely on the input signal, but additionally on an internal state $\boldsymbol{x}$. Linear, time-invariant systems are regularly described by differential equations

$$\sum_{i=0}^{n} a_i \frac{d^i \boldsymbol{y}(t)}{dt^i} = \sum_{j=0}^{m} b_j \frac{d^j \boldsymbol{u}(t)}{dt^j} \tag{4}$$

whereas only systems with $m <= n$ are physical realizable. It is always possible to transform a differential equation of n-th order to n coupled differential

equations of first order

$$\frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{5}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{E}\boldsymbol{u}(t), \tag{6}$$

called the state-space description. The transition matrix $\boldsymbol{A}$ describes the transition from one state $\boldsymbol{x}$ to the next, $\boldsymbol{B}$ the influence of the input $\boldsymbol{u}$ on the state. The output $\boldsymbol{y}$ depends on the state, as described by $\boldsymbol{C}$ and on the input $\boldsymbol{u}$, depicted by $\boldsymbol{E}$. A second order system may overshoot when the desired value is increased and need a long time to converge to a new value. In this article 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 Sect. **??**.

In (**??**) and (**??**) $\boldsymbol{A}$ is an n x n matrix, where n is the order of the differential equation. For single input single output systems, i. e. both y and u are scalars instead of vectors, $\boldsymbol{B}$ and $\boldsymbol{C}$ are vectors of length n and $\boldsymbol{E}$ 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 (**??**) leads to the consideration 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.

2.3 Normal forms

As mentioned in the last section there is no unique state space description for a given differential equation. E. g. imagine a dynamic system with gain K. To model such a system the fortification can either take place between the input $\boldsymbol{u}$ and the state $\boldsymbol{x}$ or between the state $\boldsymbol{x}$ and the output $\boldsymbol{y}$.

Control theory distinguishes three different normal forms. As this paper is no introduction to control theory only the observable canonical form, being used for the training of the BNs, is discussed.

In the observable canonical form the four matrices get a special form. First the regarded differential equation is normalized, so that the parameter $a_n = 1$. Then the state space description is changed, so that the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{E}$ are equal to

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & 0 & -a_2 \\ \multicolumn{7}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ 0 & 0 & \cdots & & 1 & 0 & -a_{n-2} \\ 0 & 0 & \cdots & & 0 & 1 & -a_{n-1} \end{bmatrix} \tag{7}$$

$$\boldsymbol{B} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{bmatrix} \tag{8}$$

$$\boldsymbol{C} = [0\,0\,0\cdots 1] \tag{9}$$

$$\boldsymbol{E} = b_n. \tag{10}$$

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 $2n + 1$ which is equal to the number of the parameter in the differential equation if $n = m$ and a_n is normalized to 1. Thus no redundancy is left. More details about linear control theory are found in nearly every introduction, e. g. [**?**] and [**?**] are useful for understanding normal-forms.

3 Bayesian networks

Modeling with Bayesian Networks is equivalent with learning a probability distribution $p(X_1, X_2, \cdots, X_n)$ which represents the data as well as possible. Assuming independencies between the variables, the joint distribution simplifies to

$$p(x_1, x_2, \cdots, x_n) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i|pa(i)). \tag{11}$$

with $pa(i)$ being the instantiation of $Pa(X_i)$. This means that the distribution of a node X_i depends only on its parents $Pa(X_i)$. There are several types of BNs, which can be distinguished by the type of nodes used. We restrict ourselves to normally distributed, continuous nodes i. e.

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{X_0} + \boldsymbol{W}_X \boldsymbol{y}, \boldsymbol{\sigma}_X), \tag{12}$$

where $Y = Pa(X)$ are the parent nodes of X, $\boldsymbol{\mu}_{X_0}$ is the mean when no parent exists or all parent have zero values. The weight matrix $\boldsymbol{W}_X$ is used to characterize the influence of Y on X. $\boldsymbol{\sigma}_X$ is the covariance matrix of the normal distribution. The restriction to normally distributed nodes enables us to use the inference algorithms described in [**?**], avoiding time consuming sampling procedures. Additionally there is no need to bother about convergence problems. This is important as a controller has to react in real-time.

One of the most important operations on BNs is the calculation of marginal distributions. Given a full distribution $p(X)$ with $X = \{X_1, \cdots, X_n\}$ an arbitrary distribution $p(X \backslash C)$ with $C \subset X$ can be calculated by integration over all variables in C:

$$p(X \backslash C) = \int_C p(X) dC. \tag{13}$$

A more detailed description of the algorithms used for BNs is given in [**?**], [**?**] or [**?**].

3.1 Dynamic Bayesian networks

For many purposes a static description is sufficient, see e. g. [**?**]. But there are a lot of applications when time is an important factor, i. e. the distribution of a variable $X(t)$ depends not only on other variables, but also on its own value at a previous time step, e. g. systems described by (**??**) and (**??**). For such cases dynamic Bayesian networks are developed, which are able to monitor a set of variables at arbitrary, but fixed points of time, i. e. time is no longer a continuous variable.

For each modeled point in time a static Bayesian network is used. These time slices are linked to represent the state of a variable at different points in time. Regarding the state space description of (**??**) the state $\boldsymbol{x}_{t+1}$ depends on the input $\boldsymbol{u}_t$ and the state $\boldsymbol{x}_t$.

In our application the states are regarded as normally distributed, i. e.

$$p(\boldsymbol{x}_{t+1} \mid pa(\boldsymbol{x}_{t+1})) = p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{A}_{BN}\boldsymbol{x}_t + \boldsymbol{B}_{BN}\boldsymbol{u}_t, \boldsymbol{\sigma}). \tag{14}$$

For the evaluation a DBN can be interpreted as a static BN with equal parameter for all time slices respectively between the time slices. A deeper introduction is found in [**?**]. Well-known DBNs are Kalman filter which are mostly used in control theory for tracking and prediction of linear dynamic systems.

3.2 Kalman filter

Our aim is to develop a controller which uses a DBN as model to generate the control signals. As a first step the model used for systems described by (**??**) and (**??**) will be developed. As a result we will get the structure, the weight matrices and the mean values of a DBN. In Sect. **??** this DBN is used to calculate the necessary input signals via marginalization.

In control theory Kalman filters are a well-known method for tracking and prediction of stationary, linear systems as described in Sect. **??**. Furthermore they are a special case of DBNs, so the results obtained for Kalman filter, e. g. in [**?**], may be used without any changes.

DBNs represent a time discrete system whose state transition

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}_{BN}\boldsymbol{x}_t + \boldsymbol{B}_{BN}\boldsymbol{u}_t \tag{15}$$

is described by the matrix $\boldsymbol{A}_{BN}$. In this equation t is used as index to denote a time discrete system. To calculate $\boldsymbol{A}_{BN}$ it is useful to regard the state transition of a homogeneous system, i. e. $\boldsymbol{u}(t) = 0$,

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0)\boldsymbol{\Phi}(t, t_0) \tag{16}$$

$$\boldsymbol{\Phi}(t, t_0) = \sum_{i=0}^{\infty} \boldsymbol{A}^i \frac{(t - t_0)^i}{i!}. \tag{17}$$

Assuming $t = t_{k+1}$ and $t_0 = t_k$ with a constant time difference $\Delta T = t_{k+1} - t_k$ the matrix $\boldsymbol{A}_{BN}$

$$\boldsymbol{A}_{BN} = \boldsymbol{\Phi}(t_{k+1}, t_k) \tag{18}$$

depends only on the time difference ΔT and $\boldsymbol{A}$. Thus $\boldsymbol{A}_{BN}$ is constant for all k. Taking into account the influence of the input on the state leads to

$$\boldsymbol{B}_{BN}\boldsymbol{u}_t = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}(t_{k+1}, \tau)\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{19}$$

which simplifies to

$$\boldsymbol{B}_{BN} = \Delta T \sum_{i=0}^{\infty} \frac{\boldsymbol{A}^i \Delta T^i}{(i+1)!}\boldsymbol{B}. \tag{20}$$

when only systems with a constant $\Delta T = t_{k+1} - t_k$ and a constant input $\boldsymbol{u}_t = \boldsymbol{u}(\tau)$ within one timeslice are considered. To build a DBN, which incorporates these equations $\boldsymbol{B}_{BN}$ is used as weight matrix between the input nodes and the state nodes. The 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 means that the state at time $t + 1$ is calculated by

$$\boldsymbol{x}_{t+1} = [\boldsymbol{\Phi}(\boldsymbol{\Delta T})\ \boldsymbol{B}_{BN}] \cdot \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{u}_t \end{bmatrix}. \tag{21}$$

In a BN the mean $\boldsymbol{\mu}$ is equal to $\boldsymbol{\mu} = \boldsymbol{\mu}_0 + \boldsymbol{W}\boldsymbol{y}$. Thus $\boldsymbol{\mu}_0$ has to be set to zero and $W = [\boldsymbol{\Phi}(\boldsymbol{\Delta T})\ \boldsymbol{B}_{BN}]$. The output depends linearly on the state and is not time dependent, thus the matrix $\boldsymbol{C}$ and $\boldsymbol{E}$ may be used unchanged also in a time discrete system.

As a further consequence the dimension of the hidden state nodes is equal to the order of the differential equation describing the system.

3.3 Structure of the Bayesian network

Until now only the control transfer function of the system is modeled, i. e. the reaction when the manipulated variable $\boldsymbol{u}$ is changed. Experiments not described in this paper show that a system where only input, output and state variables are modeled, is sufficient for building a controller as long as only the manipulation reaction of the system is concerned. If disturbance variables occurs an additional slice for the disturbance variables has been used. According to (**??**) the disturbance value $\boldsymbol{z}$ has to be added to $\boldsymbol{y}$ to come to the measured output $\boldsymbol{q}$. The weight of the links is set to one. This results in Fig. **??**. The additional node z has two different functions. When a perfect model is used the only task of z is to model the disturbance variable. In our test scenario a reference value $w = 10$ and a disturbance variable of $z = 1$ is used. In this case the state x must take on a value, so that $y = q - z = 9$.

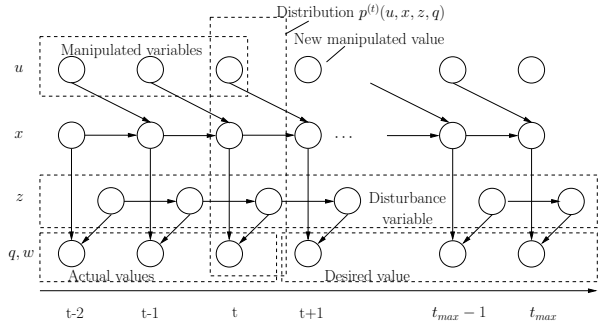

Fig. 2. Principal structure of a BN used for control purposes

When trained models are used z has the additional function to make up the differences between the model and the reality.

To simplify the training the observable canonical normal form is used. When (**??**) and (**??**) are applied to (**??**) and (**??**) to calculate an adaptation to time discrete systems the typical form of the normal form is destroyed. But control theory shows that also for time discrete systems normal form exists. For the observable canonical form only the last column has to be adapted, i.e. only the parameters a_i and b_i changes. Now each zero can be interpreted as a missing link, each time a position in $\boldsymbol{A}$ is equal to one a weight can be clamped. Thus the usage of normal forms has two advantages. First the number of input nodes connected to the output is reduced which results in higher speed. Second the weights of all nodes being connected to the output node is known. Thus it can be clamped and is not changed during training. Indeed this has been the reason to use the observable canonical form. The remaining knowledge about normal forms is used to find good initializations for $\boldsymbol{A}_{BN}$ and $\boldsymbol{B}_{BN}$.

It remains the question about the weight between $\boldsymbol{z}_t$ and $\boldsymbol{z}_{t+1}$. Here it is assumed that environmental changes are relatively slow, so that the statistical properties of $\boldsymbol{z}_t$ are approximately equal to $\boldsymbol{z}_{t+1}$. Thus a weight of one is taken.

4 Calculation of control signals

In Sect. **??** we showed how to set the weight matrices and mean values of the DBN, the questions about the time-difference ΔT and the covariance matrices are still open. We first introduce the structure and the generation of the input signal before we deal with the remaining parameters. For the generation of the input variable $\boldsymbol{u}$ a DBN with a fixed number of time-slices as depicted in Fig. **??** is used.

To generate the manipulated value $\boldsymbol{u}_{t+1}$ the first part of the input nodes is used to enter the history. In Fig. **??** these are the nodes $\boldsymbol{u}_{t-2}$ up to $\boldsymbol{u}_t$, for our

experiments[1] we used 10 nodes for the representation of the past. Moreover the observed output values are stored and entered as evidence using the nodes $\boldsymbol{q}_{t_0}$, the oldest stored output value, till $\boldsymbol{q}_t$. The state cannot be observed, so no evidence is given for the random variable $\boldsymbol{x}$. Now it is the task of the DBN to calculate a signal that changes the system's output to the desired signal and to keep that output constant. To tell the system to do so the desired value $\boldsymbol{w}$ is also entered as evidence. This means the desired future values for the output nodes are treated as they were already observed and entered as evidence for all the nodes $\boldsymbol{q}_{t+2}$ till $\boldsymbol{q}_{t_{max}}$. No evidence is given for $\boldsymbol{q}_{t+1}$ as this value is determined by an already calculated input. To control the plant it is necessary to calculate the value $\boldsymbol{u}_{t+1}$ which leads to the desired value $\boldsymbol{w}$. This is done by marginalization. The new input is passed to the simulation of the dynamic system and the resulting output is calculated. Then a complete cycle is finished. The used input and the resulting output are added to the history and the next input is calculated. To ensure that the calculation of the input signal is not limited to a certain amount of time the evidence is shifted to the left after each time step, i. e. the oldest input and output values are deleted. Then the current signal is entered at time t. The future values may remain unchanged if the desired value is not changed. This works well for slow systems, for systems with the ability to oscillate this would result in oscillating input signals. Thus it is necessary to damp the input. To do so a weighted sum of $\boldsymbol{u}_t$ to $\boldsymbol{u}_{t+k}$ is used. For our experiments, described in Sect. **??**, a weighted sum of 4 nodes are used.

It remains the question which ΔT is appropriate for the dynamic system. According to control theory a system $K\boldsymbol{u}(t) = \boldsymbol{q}(t) + T_1 \frac{d\boldsymbol{q}}{dt} + T_2^2 \frac{d^2\boldsymbol{q}}{dt^2}$ of second order has a natural angular frequency of $\omega_0 = \frac{1}{T_2}$. The minimal sampling rate has to be at least twice the frequency to be measured.

Our first experiments are done with very small covariances, because we used an accurate model based on an analytical description and are not interested in loosing information due to great covariances. Please note that zero covariances are not possible, due to matrix inversion during evaluation. As a consequence we got an accurate model that was unable to calculate appropriate control signals. The reason is that small covariances at the input nodes, together with zero mean values results in a high probability for an input close to zero which can not be used for control purposes. Therefore we changed the covariance of the input node to a maximum to tell the system, that there is no a-priori information about the correct value of the input signal. The other covariances remain unchanged to keep the accurate modeling behavior.

[1] The experiments were done with Matlab, Simulink to simulate the dynamic systems and the BN-Toolbox, an expansion of Matlab which is freely available at http://www.cs.berkeley.edu/~murphyk/Bayes/bnt.html

5 Experiments

Our experiments were done with three different systems of second order which are simulated by Simulink. These systems are described by differential equations

$$K\boldsymbol{u}(t) = \boldsymbol{q}(t) + T_1\frac{d\boldsymbol{q}}{dt} + T_2^2\frac{d^2\boldsymbol{q}}{dt^2} \tag{22}$$

and their behavior depends mainly on the two parameters T_1 and T_2. If the damping $D = \frac{T_1}{2T_2}$ is greater than one the system has no tendency to overshoot which means these systems are easy to control. Systems with $0 < D < 1$ have the ability to oscillate and overshooting can be regarded when the input signal is changed. Our test systems are described in Table **??**. At the beginning of

Table 1. Description of test systems

Nr.	K	T_1	T_2	Description
1	2	1	0.1	Damped system with gain two which has no tendency to overshoot
2	2	0.1	0.1	System with $D < 1$ which means that there is a tendency to overshoot
3	10	0.05	0.1	System with high gain and a large tendency to overshoot

each test the desired value is set to 0 and shortly afterwards it is changed to 10. After convergence the disturbance input z is changed to 1. Thus at the first moment the output changes to a value close to 11. Then the input is changed, so that the output reaches the desired value once again. The signals are stored, so that the quality measures can be calculated.

5.1 Experiments with calculated models

First experiments were done with models whose parameters were retrieved analytically. The reason for these experiments are to show that Bayesian networks used in the manner described in Sect. **??** can be really used for control purposes and on the other hand to get comparative values. The used quality measures are described in Table **??**, the results of our experiments in Table **??**. Experiments with the third system are done twice with different sampling rates. As expected it can be seen that a higher sampling rate improves the result. The signals of system two can be seen at Fig. **??**. At the beginning a steep raise of both the input and the output signal can be observed. The output signal reaches its maximum of 10.56 and after 0.5 s the error is below the 1% level. At $t = 4.1$ s the disturbance variable is changed to $z = 1$ and half a second later the effect of the disturbance disturbance is nearly vanished.

In all four cases the system shows a very good performance, the desired value is reached fast and with nearly no deviation. Also when the disturbance value is changed from 0 to 1 the system reacts as intended. The input is changed so that the desired value is reached once again.

Table 2. Used quality measures

Quality measure	Description
$Q_d(p\%, z = d)$	Squared error sum as defined in (**??**). Calculation stopped when the deviation between w and q is smaller than $p\%$. Tests done with a disturbance variable $z = d$.
Overshoot	The difference between the maximal output value q_{max} and w.
e_r	The remaining error $e = q - w$ after convergence takes place.
$te_{max}(z = 0, p\%)$	Raise time until the output has changed from $q = 0$ to $q = w$ when no disturbance occurs.
$te_{max}(z = 1, p\%)$	Settling time until the error is smaller than $p\%$ starting with the occurrence of the disturbance input.

Table 3. Results of experiments with calculated and trained models

	Calculated models				Trained models		
Nr.	1	2	3.1 ($\Delta T = 0.05$)	3.2 ($\Delta T = 0.02$)	1	2	3.2 ($\Delta T = 0.02$)
$Q_d(1\%, z = 0)$	8.43	9.63	8.08	6.97	8.11/10.66	9.43	8.43/5.35
$Q_d(3\%, z = 0)$	8.43	9.62	8.06	6.97	8.06/9.58	9.42	8.40/5.18
$e_r(z = 0)$	-0.01	-0.01	0.00	0.00	0.07	0.01	0.09
Overshoot	-0.01	0.56	0.35	0.02	0.62	2.4	1.41
$Q_d(1\%, z = 1)$	0.15	0.18	0.22	0.12	0.2/0.40	0.28	1.19/1.31
$Q_d(3\%, z = 1)$	0.14	0.18	0.21	0.11	0.19/0.08	0.27	0.67/1
$e_r(z = 1)$	0.00	-0.01	-0.01	0.01	0.06	0.02	0.24/0.38
$te_{max}(z = 0, 1\%)$	0.5	0.45	0.7	0.26	2.21/31.45	1.16	1.37/8.96
$te_{max}(z = 0, 3\%)$	0.4	0.4	0.45	0.22	0.58/4.1	0.82	0.36/0.45
$te_{max}(z = 1, 1\%)$	0.45	0.5	0.55	0.48	0.55/22.55	0.92	8.92/8.93
$te_{max}(z = 1, 3\%)$	0.3	0.3	0.4	0.26	0.4/0.2	0.6	3.37/8.93

5.2 Experiments with trained models

The results described in the last section are based on analytically retrieved Bayesian networks. As our aim is to use Bayesian models as self adaptive controller we have tested our approach also with trained Bayesian networks. The training material is generated by simulation of the pulse and step response of the dynamic system. Particularly the first one is used in control theory for system identification in simple cases. In-depth analysis to find out which training signals would be best are still to be done, first experiences show that the used signals provide results above the average.

40 different time series are used for training and 20 iterations. In our experiments 20 iterations are usually sufficient for convergence. The experiments are repeated 10 times, to control the robustness of the results. Best results are obtained for system 2 (cf. Table **??**), an example is shown in Fig. **??**. As Table **??** shows the desired value is reached with an mean accuracy of 0.01 which means that there is almost no difference between the calculated and the trained model. To avoid that positive and negative values are averaged to zero the calculation of the mean of the steady state error is based on absolute values. In some cases a better result is obtained as in the mathematical model. The reason might be that the variations in the math-

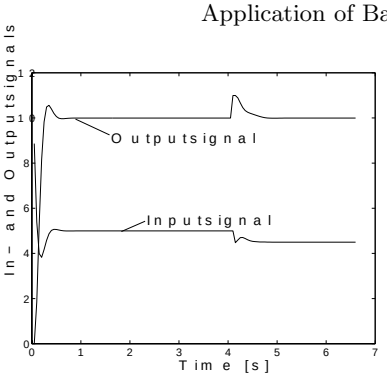

Fig. 3. Signals for system 2 controlled by a calculated controller

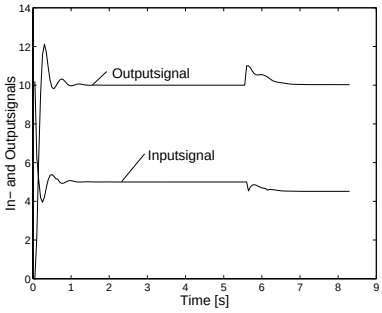

Fig. 4. Signals for system 2 controlled by a trained controller

ematical models are clamped to a fixed value which is sufficient for control purposes but is not proven to be optimal. When comparing the Quality Q_d of the trained and the calculated model the calculated model is only slightly better. The main difference is that the trained model generated a higher input signal at the beginning so that the error is reduced faster. This results in a higher overshoot and a longer settling time. In contrast the calculated controller tries to reach the desired value without any overshoot.

Also the disturbance reaction is fine for the trained system. A mean deviation of 0.2% of the reached value from the desired value is acceptable, even if in that case the mathematical model shows its superiority regarding the accuracy and the speed.

The results for system number 1 are slightly below system 2. The desired value is reached with a mean deviation of 0.7%, but two times the resulting steady state error is not below 1%. Thus in two cases the upper time te_{max} when calculating Q_d depends on the time of convergence, not on reaching the given accuracy. Therefore two different means are calculated. The first value is the mean of cases with $e_r < 1\%$, the second value is the mean of the remaining systems.

Also the disturbance reaction shows good performance. When the disturbance value is changed, the desired value is reached with an error of 0.6%. Concerning the performance the same observations as for the second system can be made. The error is reduced faster in the beginning, which results in a higher overshoot and a longer settling time. When the disturbance reaction is concerned the trained model behaves well, but the mathematical model is clearly better.

Experiments with the third system and $\Delta T = 0.05\,\mathrm{s}$ show no acceptable results. Three trials were made and when testing the reference reaction the deviation of the desired value is between 0.5% and 2.3%. But when a disturbing value appears a deviation between 7.1% and 9.5% is observable.

Comparing this steady state error with the disturbance of 10% of the desired value clarifies that the disturbance reaction is not acceptable.

Thus the sampling rate was increased from $\Delta T = 0.05\,\mathrm{s}$ to $\Delta T = 0.02\,\mathrm{s}$. Increasing the sampling rate results in a great improvement of the performance. The deviation of the desired value drops to 0.9% as long as no disturbance appears. A greater problem is the disturbance reaction which results in a steady state error of 2.7% which means that the effect of the disturbance is eliminated only to approximately 80%. Additionally the Bayesian controller reacts relatively slowly in that case.

The greatest problem is the missing convergence in one case. It is well-known that the EM-algorithm may converge to a local minimum. To avoid such cases as much a priori knowledge as possible is used, e. g. the knowledge about normal forms. To avoid missing convergence in the future two remedies will be studied. First we consider the usage of models with no hidden nodes. On the other hand an online training during control can take place to further reduce the error.

The three examples points out that a Bayesian controller is a perfect mean for controlling linear system. When based on a mathematical description the desired value is reached, so that no overshoot can be observed. But also when working with trained models a high accuracy is shown and it seems worthwhile to expand the described approach to nonlinear system by the usage of hybrid Bayesian networks.

To ensure applicability in real world domains there are two important points to be improved. The first point is real-time. In the simulations described here the time for calculating the new input signal and the system's response is approximately 1.5 s. The first step to be done is to use compiled programs instead of interpreted ones. Later on approximative algorithms may be used to further improve the performance.

The second point is to improve robustness. One approach is to change the used inference algorithm. The usage of the algorithm described in [**?**] is intended.

6 Summary

Starting from an analytical description the structure of a dynamic Bayesian network was developed and an explanation was given how to calculate the parameters of a Bayesian network. These network are used as controller by using the desired value as evidence and using the marginal distribution of the input nodes as input signal for the controlled system.

The performance of a Bayesian controller based on trained and analytically retrieved models is compared regarding both the reference and the disturbance reaction of the controlled system. Even if the analytical model performs better the trained model shows a very good performance, e. g. the steady state error is below 1% in most of the cases. For the future there will

be several points of interest. To get a better impression of the practical applicability the performance of a Bayesian controller has to be compared with PID controllers which are widely used in industry and it has to be examined how the Bayesian controller can be optimized with respect to special constraints, e. g. no overshoot or limited input. Other important points are the ability to react in real time and the modeling of non-linearities.

Using Bayesian networks for control purposes is a relative new approach, so there is nearly no literature about controllers based on BNs. Welch [**?**] proves that BNs might be used in real time for control purposes, but he is restricted to static BNs and the choice between two possible actions.

References

1. Boyen X., Koller D. (1998) Approximate learning of dynamic models. In: Proc. of the 12th Annual Conference on Neural Information Processing Systems, Denver, Colorado
2. Deventer, R., Denzler, J., Niemann, H. (2000) Non-linear modeling of a production process by hybrid Bayesian Networks. In: Werner Horn (Ed.): ECAI 2000 (Berlin), IOS Press, Amsterdam, Berlin, Oxford, Tokyo, Washington DC, pages 576–580
3. Friedman, N., Murphy, K., Russel S. (1998) Learning the Structure of Dynamic Probabilistic Networks. In: 12th UAI.
4. Gelb A. (1994) Applied optimal estimation. MIT Press, Cambridge, Massacusetts, USA.
5. Jensen, F. V. (1996) An introduction to Bayesian networks. UCL Press.
6. Kjærulff U. (1992) A computational schema for reasoning in dynamic probabilistic networks. In: 8th UAI
7. Lauritzen, S. L. (1992) Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models. J. of the American Statistical Association
8. Lauritzen, S. L. (1996) Graphical Models. Oxford University Press.
9. Lauritzen, S. L., Jensen, F. (1999) Stable Local Computation with Conditional Gaussian Distributions. Technical report, Aalborg University, Department of Mathematical Sciences.
10. Luenberger D. C., Luenberger D. G. (1979) Introduction to Dynamic Systems: Theory, Models, and Application. John Wiley.
11. Olesen,K. G. (1993) Causal probabilistic networks with both discrete and continuous variables. IEEE Transaction on Pattern Analysis and Machine Intelligence, 15, Vol. 3.
12. Heinz Unbehauen, H. (1993) Regelungstechnik II. Vieweg.
13. Welch R. L., Smith C. (1999) Bayesian Control for Concentrating Mixed Nuclear Waste. In: 15th UAI.