# Control of Dynamic Systems Using Bayesian Networks *

RAINER DEVENTER, JOACHIM DENZLER, HEINRICH NIEMANN

Chair for pattern recognition
Martensstr. 3
D 91058 Erlangen
Telephone: +49 (0)9131 85-27825
{deventer, denzler, niemann}@informatik.uni-erlangen.de

**Abstract.** Bayesian networks 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. 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 know similarities between Bayesian networks and Kalman filters to model and control linear dynamic systems using dynamic Bayesian networks. We show, how the model is used to calculate appropriate input signals for the dynamic system to achieve a required output signal. First the desired output value is entered as additional evidence. Then marginalization results in the most likely values of the input nodes.

The experiments show that with our approach the desired value is reached in reasonable time and with great accuracy. Additionally, oscillating systems can be handled. The benefits of the proposed approach are the model based control strategy and the possibility to learn the structure and probabilities of the Bayesian network from examples.

**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 [3].

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 solve inverse problems, i.e. in any case the most likely parameters of the system can be computed given information about some parameters of the system that is entered as evidence in the BN.

In this paper it is shown that BN can also act as controller. We exploit the well know 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. Additionally, any other parameter of the dynamic system can be inferred given the remaining parameters as evidence. This procedure together with the well known training algorithms for BNs allows the realization of self adaptive controllers which is particularly important for nonlinear processes.

We focus in the paper on the modeling of stationary, linear dynamic systems of second order. The extension to control nonlinear systems, though, is straight forward using hybrid BN, i. e. a BN using both discrete and continu-

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

ous nodes. In statistical terms this means that a mixture of Gaussians is used instead of only one normal distribution [16].
We show how systems of higher order can be handled by breaking down the higher order system to a sequence of second order systems. The structure of the DBN and its parameters are inferred from an analytical description that we get from classical control theory in order to evaluate our results. However, for real world examples, the parameters of the BN are learnt from examples.

The experiments show that with our approach the desired value is reached in reasonable time and with great accuracy. Additionally, oscillating systems can be handled. The benefits of the proposed approach are the model based control strategy and the possibility to learn the structure [4] and probabilities [1] of the Bayesian network from examples.

This article is structured as follows. In section 2 the term control system is defined and the analytical descriptions as they are used in control theory are introduced. Section 3 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 section 4. The results obtained with this new type of controller are presented in section 5.

## 2 Dynamic systems

A dynamic system may be regarded as a black box with several input and output signals $\boldsymbol{u}$ respectively $\boldsymbol{q}$, 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

$$\sum_{i=0}^{n} a_i \frac{d^i \mathbf{q}(t)}{dt^i} = \sum_{j=0}^{m} b_j \frac{d^j \mathbf{u}(t)}{dt^j}. \tag{1}$$

It is always possible to transform a differential equation of n-th order to n coupled differential equations of first order,

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{2}$$

$$\mathbf{q}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{E}\mathbf{u}(t) \tag{3}$$

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 $\boldsymbol{q}$ depends on the state, as described by $\boldsymbol{C}$ and on the input $\boldsymbol{u}$, depicted by $\boldsymbol{E}$. To get a good impression of such a system it is helpful to regard the step response of such a system, that is regarding the output signal $\boldsymbol{q}(t)$ when $\boldsymbol{u}(t)$ is changed from 0 to 1 at t = 0. This step response is depicted in figure 1 for a system of second order.

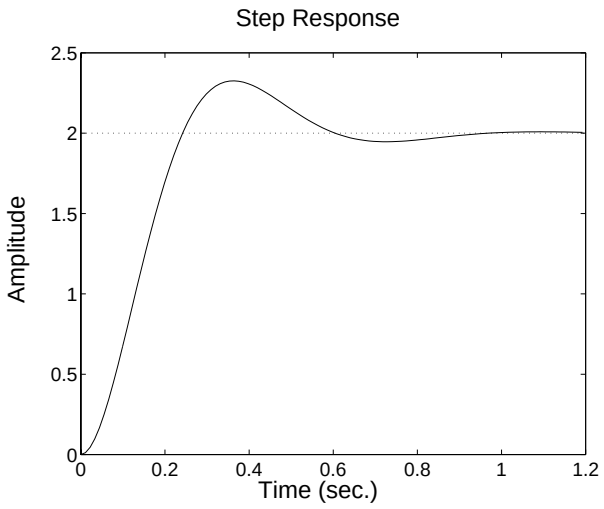


**Fig. 1.** Step response of a second order system

Regarding figure 1 the reader will notice that a second order system may overshoot and needs 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 filter as described in section 3.2.

As a simple example a car with mass $M$ which is accelerated by a force $F$ and slowed down by friction and a spring is regarded. The friction is proportional to the product of a constant $b$ and the velocity $v = \frac{dx}{dt}$, the excursion $x$ of the spring causes a force $kx$. Thus, the following equation holds:

$$M\frac{d^2x(t)}{dt^2} + b\frac{dx}{dt} + kx = F. \tag{4}$$

By substitution $x_1 = \frac{dx}{dt}$ and $x_2 = x$ the example can be transformed to a system of differential equations of first order, called the state space representation.

$$\begin{bmatrix} \frac{dx_1(t)}{dt} \\ \frac{dx_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{b}{M} & -\frac{k}{M} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{M} \\ 0 \end{bmatrix} F(t) \tag{5}$$

More details about linear control theory are found in nearly every introduction, e. g. [12].

## 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

$$p(x_1, x_2, \cdots, x_n) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i|pa(i)). \quad (6)$$

with $pa(i)$ as 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. $p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}(\mu_{\mathbf{X_0}} + \boldsymbol{W}_{\mathbf{X}}\boldsymbol{y}, \boldsymbol{\sigma}_{\mathbf{X}})$ where $Y = Pa(X)$ are the parent nodes of $X$, $\boldsymbol{\mu}_{\boldsymbol{X_0}}$ is the mean when no parent exists or all parent have zero values. $\boldsymbol{W}_X$ is a weight matrix 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 [9], 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 $C$:

$$p(X \backslash C) = \int_C p(X) dC. \quad (7)$$

A more detailed description of the algorithms used for BNs is given in [9], for a more detailed introduction see [10] or [6].

### 3.1 Dynamic Bayesian networks

For many purposes a static description is sufficient. 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 equations 2 and 3. 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. a time discrete representation is used.

For each modeled point of 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 eq. 2 the state $\boldsymbol{x}_{t+1}$ depends on the input $\boldsymbol{u}_t$ and the state $\boldsymbol{x}_t$.

In a DBN the states are regarded as normally distributed, i. e. $p(\boldsymbol{x}_{t+1} \mid pa(\boldsymbol{x}_{t+1})) = p(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{A}_{BN}\boldsymbol{x}_t + \boldsymbol{B}_{BN}\boldsymbol{u}_t, \boldsymbol{\sigma})$. For the evaluation a DBN can be regarded as a static BN with equal parameter for all time slices respectively between the time slices. A deeper introduction is found in [8]. Well known DBNs are Hidden Markov Models and 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 equations 2 and 3 will be developed. As a result we will get the structure, the weight matrices and the mean values of a DBN. In section 4 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 section 2. Furthermore they are a special case of DBNs, so the results obtained for Kalman filter, e. g. in [5], may be used without any changes.

The state $\boldsymbol{x}(t)$ of a homogeneous systems, i. e. $\boldsymbol{u}(t) = 0$, is calculated as follows:

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_0)\boldsymbol{\Phi}(t, t_0) \quad (8)$$

$$\boldsymbol{\Phi}(t, t_0) = \sum_{i=0}^{\infty} \boldsymbol{A}^i \frac{(t - t_0)^i}{i!}. \quad (9)$$

As DBNs represent a time discrete system $\boldsymbol{\Phi}$ cannot be used directly as weight matrix in a DBN. Time discrete systems are described by difference equations

$$\boldsymbol{x}_{k+1} = \mathbf{A}_{BN}\boldsymbol{x}_k + \mathbf{B}_{BN}\boldsymbol{u}_k \quad (10)$$

that are solved by

$$\mathbf{A}_{BN} = \boldsymbol{\Phi}(t_{k+1}, t_k) \quad (11)$$

$$\mathbf{B}_{BN}\boldsymbol{u}_k = \int_{t_k}^{t_{k+1}} \boldsymbol{\Phi}(t_{k+1}, \tau)\boldsymbol{B}\boldsymbol{u}(\tau) d\tau. \quad (12)$$

If we restrict ourselves to systems with a constant $\Delta T = t_{k+1} - t_k$ and assuming that the input remains constant during a timeslice, then $\boldsymbol{\Phi}(t_{k+1}, t_k) = \boldsymbol{\Phi}(\Delta T)$ stays constant for all $k$ and equation 12 simplifies to

$$\boldsymbol{B}_{BN} = \Delta T \sum_{i=0}^{\infty} \frac{\boldsymbol{A}^i \Delta T^i}{(i+1)!} \boldsymbol{B}. \quad (13)$$

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

$$\boldsymbol{x}_{t+1} = [\boldsymbol{\Phi}(\boldsymbol{\Delta T})\ \boldsymbol{B}_{BN}] \cdot \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{u}_t \end{bmatrix}. \tag{14}$$

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. In figure 2 two time-slices of the second order system used to model the example of section 2 are shown. Please note that the rectangles in this picture do not represent any random variable, but the weight matrices and how they are used to calculate the means of the nodes, represented by circles.

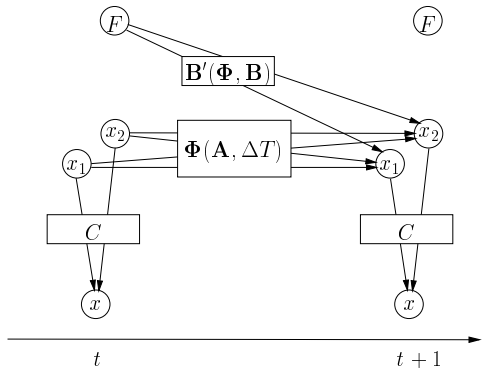


**Fig. 2.** Weight matrices in a Kalman filter

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

## 4 Calculation of control signals

In section 3.2 we showed how to set the weight matrices and mean values of the DBN, the questions about the time-difference $\Delta T$ 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 figure 3 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 figure 3 these are the nodes $\boldsymbol{u}(t-2)$ up to $\boldsymbol{u}(t)$, for our experiments[1] we used 20 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 $x$. Now it is the task of the DBN to calculate a signal that can be used to change the system's output to the desired signal and to keep that output constant. To instruct 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 $\boldsymbol{q}(t+1)$ till $\boldsymbol{q}(t_{max})$. To control the plant it would be best to calculate the signal which leads with a maximal probability to the new desired value. For reasons of simplicity we used the marginal value of $\boldsymbol{u}(t+1)$ instead of calculating the value of $\boldsymbol{u}(t+1)$ which leads with a maximal probability to the given evidence. The new input was 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 thrown away. Then the current signal is entered at time $t$. The future values may remain unchanged if the desired value is not changed.

It remains the question, what $\Delta T$ 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 an eigenfrequency of $\omega_0 = \frac{1}{T_2}$. The minimal sampling rate is twice the frequency to be measured. Therefore the maximal value for $\Delta T$ should be

$$\Delta T_{max} = \frac{T_2}{2}. \tag{15}$$

Our first experiments are done with very low 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 reasons are that a low covariance at the input nodes,

[1] The experiments were done with Matlab, the Control toolbox 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

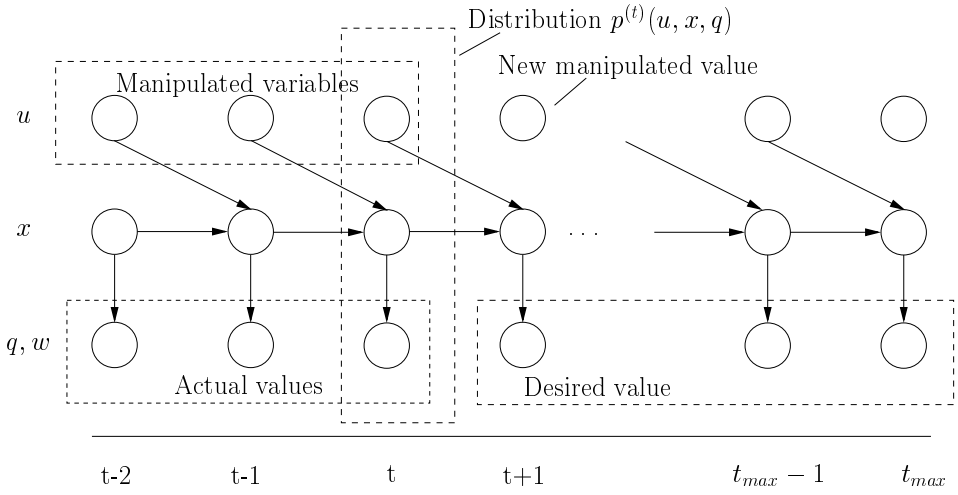


**Fig. 3.** Principal structure of a BN used for control purposes

together with zero mean values results in a very low probability that $u(t) \neq 0$. 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. For a further improvement we used the EM algorithm (see [2], [13], [15] or [14]) to learn the mean of the input nodes. We tested the reference action of the control loop on systems with proportional response with second order delay. This systems are described by a differential equation of second order.

## 5 Experiments

Our system was tested with a Bayesian network with the structure given in figure 3. The old input- and output signals together with the desired values are entered as evidence. Then the marginal value for $\boldsymbol{u}(t+1)$ is calculated and used as input for a system of second order that was simulated by Control, a Matlab toolbox. These systems are defined by the differential equations

$$K\boldsymbol{u}(t) = \boldsymbol{q}(t) + T_1 \frac{d\boldsymbol{q}}{dt} + T_2^2 \frac{d^2\boldsymbol{q}}{dt^2} \qquad (16)$$

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 that these systems are easy to control. To test our controller the desired value was changed from 0 to 20 and the system's response was regarded. The quality of the controller depends on the time needed until the new output value is reached, the deviation between the desired value and the actual value and on the overshoot. Figure 4 describes the behavior of a system with $K = 2$, $T_1 = 1$, $T_2 = 0.1$ and $\Delta T = 0.05$. It gets clear that there is no overshoot, nearly no deviation from the desired signal and the time until the desired signal is reached is below one second.

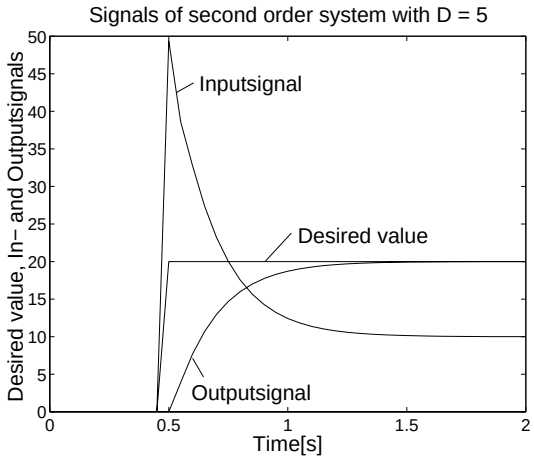


**Fig. 4.** Signals of a strong damped second order system

If $D < 1$, the system has the tendency to overshoot and to oscillate (cf. figure 5, with $K = 2, T_1 = T_2 = 0.1$ and $\Delta T = 0.05$). Also in that case the output signal is acceptable, but the input signal oscillates. Such a signal cannot be used in practical situations. Therefore it is necessary to damp down the input signal. This can be done by not only using $\boldsymbol{u}(t+1)$ as input signal, but a weighted sum of signals of past and predicted signals. Regarding the

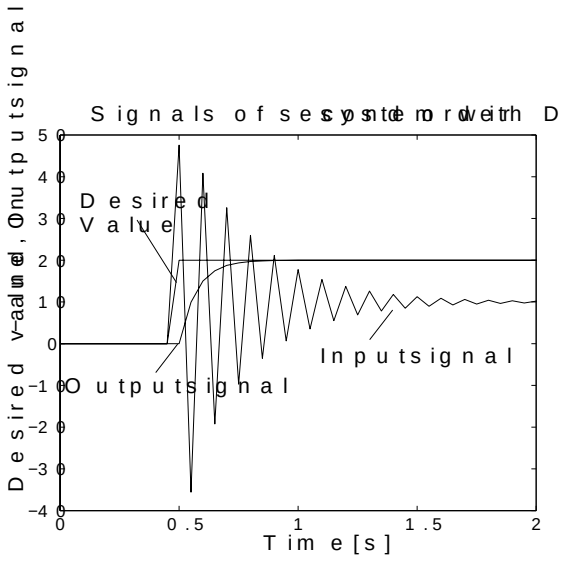


**Fig. 5.** Oscillating input

output signal resulting from the damped input at figure 6 it gets clear, that this does not change the accuracy. Only the time needed to take on the new desired value is increased.

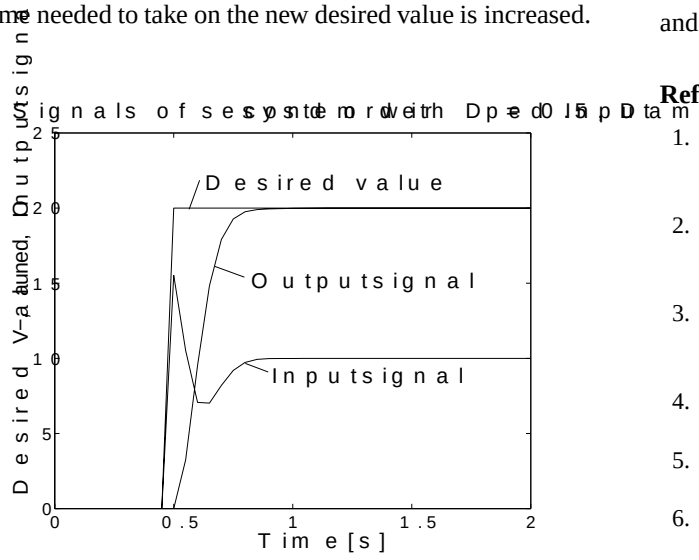


**Fig. 6.** Usage of damped input

### 5.1 Systems of higher order

Theoretically it is possible to model a system of higher order by using state nodes with the same dimension as the system's order. Experiments show that numerical problems occur when calculating the inverse of a potential's covariance matrix[2]. As each clique in a junction-tree (see [9] or [7]) contains the node itself and the node's parents the potential which contains the state nodes has at least twice the dimension of the state node. To cope with that problem a dynamic system of higher order can be split in several controlled systems of second order being connected in series. Thus the dimension of the potentials is limited and the matrix inversion is numerically stable.

## 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. Using the desired value as evidence it was shown that the marginal distribution of the input nodes may be used as input signal for the controlled system. The resulting steady state desired value deviation disappears if the mean of the input nodes is learnt, and a great dispersion is used. There is nearly no literature about controllers based on BNs, Welch [17] 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. X. Boyen and D. Koller. Approximate learning of dynamic models. In *Proc. of the 12th Annual Conference on Neural Information Processing Systems, Denver, Colorado*, 1998.
2. 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.
3. R. Deventer, J. Denzler, and H. Niemann. Non-linear modeling of a production process by hybrid Bayesian Networks. In *ECAI 2000*, 2000.
4. N. Friedman, K. Murphy, and S. Russel. Learning the structure of dynamic probabilistic networks. In *12th UAI*, 1998.
5. A. Gelb, editor. *Applied optimal estimation*. MIT Press, Cambridge, Massacusetts, USA, 1994.
6. F. V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.
7. Frank Jensen, Finn. V. & Jensen. Optimal junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, July 1994.
8. U. Kjærulff. A computational schema for reasoning in dynamic probabilistic networks. In *8th UAI*, 1992.
9. S. L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *J. of the American Statistical Association*, 1992.
10. S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.


[2] The matrix inversion is part of the used message propagation algorithm described in [9]. The newest version described in [11] is not yet implemented.

11. S. L. Lauritzen and F. Jensen. Stable Local Computation with Conditional Gaussian Distributions. Technical report, Aalborg University, Department of Mathematical Sciences, 1999.
12. David C. Luenberger and David G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Application*. John Wiley, 1979.
13. Daniel Mc Michael, Lin Liu, and Heping Pan. Estimating the parameters of mixed Bayesian networks from incomplete data. In *Proc. Information Decision and Control 99, Adelaide, Australia*, February 1999. http://www.cssip.edu.au/∼heping/PanPapers.html.
14. Kevin P. Murphy. Fitting a Conditional Gaussian Distribution, October 1998. http://http.cs.berkeley.edu/∼murphyk/.
15. Kevin P. Murphy. Inference and Learning in Hybrid Bayesian Networks. Technical report, University of California, Computer Science Division (EECS), January 1998.
16. K. G. Olesen. Causal probabilistic networks with both discrete and continuous variables. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 15, vol 3*, 1993.
17. R. L. Welch and C. Smith. Bayesian control for concentrating mixed nuclear waste. In *15th UAI*, 1999.