# TRAINING HIDDEN NON-MARKOV MODELS

Claudia Isensee
Fabian Wickborn
Graham Horton
Institut für Simulation und Graphik
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39016 Magdeburg, Germany
E-mail: {claudia,fabian,graham}@sim-md.de

**KEYWORDS**

Hidden Markov Models, Discrete Phase-type Distributions, Stochastic Petri Nets .

**ABSTRACT**

Hidden Markov models (HMM) are well known in speech recognition, where they are trained to recognize spoken words and even whole sentences. They are used to find the parameters of a so-called hidden model (usually a DTMC) by training it with observed output sequences. This paper introduces an approach to train stochastic Petri nets with the methods of HMM. As opposed to a DTMC, a stochastic Petri net can model time-continuous stochastic processes with generally distributed state transitions. The training algorithm finds the parameters of the hidden models distribution functions only by training the model with the observed output. By using a more general modelling paradigm, more realistic models can be analysed using the methods of HMM. An experiment verifies the functioning of the method for an example model.

## INTRODUCTION

### Motivation

Finding the parameters of discrete stochastic models is often a tedious process: Collecting data, matching distributions, implementing approximations. Sometimes one cannot observe the actual process that one wants to model, but rather the results or the output of that process, since the actual model states are hidden. This observable output could be for example temperature records from the 18$^{th}$ century when the actual state of the system is the weather condition, which was not recorded. Observable output could also be the visible symptoms of a patient, where the hidden state is the actual disease or physical state. Diagnosing a disease or estimating the effects of medication based on the symptoms of the patient is very interesting.

Hidden Markov models (HMM) possess the ability to model hidden processes with observable outputs. One can find out the probability or the generating path of a specific output sequence (trace) and also train a hidden model to produce a specific output. The latter is equivalent to the parameterization of the model. The hidden model of HMM is usually a discrete-time Markov chain (DTMC). This restricts the modeling capabilities of the paradigm for example to geometric state duration distributions. We propose a method to train stochastic Petri nets (SPN) with the methods of HMM. SPNs are dynamic models that contain generally distributed state transitions and are used to model continuous processes with a time-dependent stochastic behavior.

Using SPNs, one can write down the structure of the hidden model with specific outputs of the model states and then find the transitions distribution parameters by training them with observed output sequences. With this new approach dynamic and more realistic hidden models can be parameterized with the methods of HMM, which expands the range of possible application areas.

### Previous Work

Hidden Markov Models and their application in speech recognition were first published around 1970, one of the first papers is (Baum et al. 1970). A comprehensive summary of the theory including algorithms and application examples can be found in (Rabiner 1989). In order to make the models more flexible, some research was done on explicit state duration densities (hidden semi-Markov models), which however complicated the solution algorithms (Russel and Moore 1985). Expanding all HMM states to a sub-HMM was also used to realize more general state duration distributions (expanded state HMM) (Russel and Cook 1987). This increased the number of free parameters, the sub-HMM topologies were not very flexible, and the performance tuned to speech-recognition systems. In (Wickborn et al. 2006) a method is described to find the output sequence probability and the corresponding internal state sequence for continuous stochastic models with generally distributed transitions, but the method is not applicable for the training of models. An idea that was presented there, how general models could be trained, is further investigated in this paper.

## BACKGROUND

### Hidden Markov Models

Hidden Markov Models (HMM) are sometimes also called signal models, and are basically discrete-time Markov chains (DTMC) that emit signals in every step.

They are widely used in speech recognition systems and sometimes in pattern recognition.

A Hidden Markov model can be described by a 5-Tupel $(S,V,A,B,\pi)$. $S$ is the set of states of the DTMC. $V$ is the set of output symbols. $A$ is the transition probability Matrix of the DTMC. $B$ is the output probability Matrix, with $b_{ij}$ being the probability to output symbol $v_j$ in state $s_i$. $\pi$ is the initial probability vector of the DTMC. A sequence of states of the DTMC is denoted as $Q=\{ q_1, q_2, q_3,..., q_T\}$ and an observed output sequence as $O=\{ o_1, o_2, o_3,..., o_T\}$ with $T$ being the maximum number of steps of the DTMC. A parameterization of a specific HMM is often denoted by $\lambda=(A,B,\pi)$ which fully defines the model.

There are three basic questions that can be answered for a HMM and a given output sequence (trace).

1. What is the probability of producing the given output sequence $O$ with the given model $\lambda$?

2. What is the most probable state sequence $Q$ of the model $\lambda$ that produced the observed output sequence $O$?

3. Maximize the probability with which the model $\lambda$ produces the output sequence $O$ by training the parameters of the model.

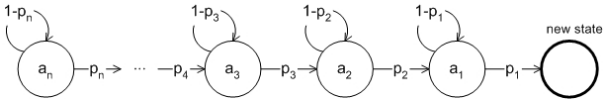
The first question can be efficiently answered by the Forward Algorithm (Rabiner 1989). The most likely state sequence can be determined by the Viterbi Algorithm. These two tasks will not be of interest in this paper and are only mentioned for completeness.

The third task of training a HMM model is solved by a kind of EM (Expectation Maximization) algorithm, the so-called Baum-Welch Algorithm (Baum et al. 1971). This algorithm takes an initial model parameterization $\lambda=(A,B,\pi)$ and iteratively improves it. The probability to produce the given output sequence O is increased in every step, but the algorithm does not necessarily find a globally optimal solution. Therefore the initial parameterization is of great importance.

### Discrete Phases

Discrete Phase-type distributions are a method to describe a non-Markovian probability distribution function through a discrete-time Markov chain segment. They were first thoroughly described and formalized in (Neuts 1981). It is also possible to approximate general distribution functions through such discrete phase-type distributions. This makes it possible to turn a discrete stochastic model (for example a Petri net) containing general transitions into a Markov chain. An approximation algorithms has been proposed in (Isensee and Horton 2005).

The structure of a discrete phase-type approximation as used in this paper is shown in Figures 1. The phases *1…n* represent the initial state with their initial probabilities $a_i$. The $p_i$ represent the state transition probabilities of the DTMC segment. The time to absorption in the *new state* mimics the non-Markovian distributions function.




Figures 1 : Structure of Discrete Phase-Type Approximation

## TRAINING NON-MARKOVIAN MODELS

### General Idea

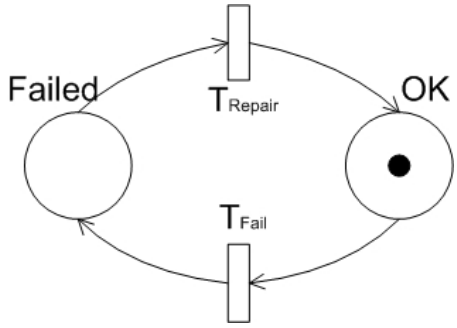
The general idea for the training of non-Markovian models is a stepwise approach. First the model to be trained is to be specified in its structure with transitions and output probabilities for the symbols and the states. The better this initial guess for the transitions distributions is, the better is the chance to find a good fit later. Then the model is turned into a DTMC by replacing the non-Markovian distributions with DPH of a chosen order.

The now Hidden Markov Model can be trained using the well known Baum-Welch Algorithm and a given output sequence or set of output sequences. The algorithm can be modified to not change the output probabilities and to leave the system structure intact by preserving zero entries in the transition probability matrix. Whether some state transitions might be deleted due to the unsupervised training process or whether states might change their roles, has to be investigated.

When the model has been successfully trained and the model structure preserved, one should be able to extract the also trained phase type distribution, which corresponds to a particular general transition in the model. One can now investigate the time to absorption in the DPH and even try and match a known general distribution function to it, which can then be used as the parameterization for the original model.

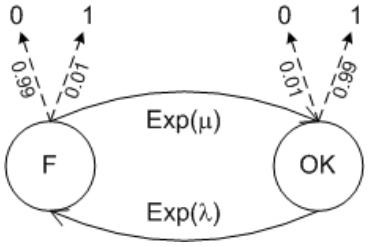
### Model Initialization

The presented approach is mainly a possibility to train parameters of real life models with a defined structure. Therefore the general structure of the model with its states and transitions needs to be known to obtain useful results. The system structure can be modeled for example using stochastic Petri nets with generally distributed firing times as described in (Bobbio et al. 1998). One restriction is that the model needs to have a finite state space, since it will later be turned into a Markov chain. An example Petri net of a machine model with the states *OK* and *Failed* corresponding to the places of the net is shown in Figures 2. The transitions are $T_{Fail}$ and $T_{Repair}$.




Figures 2 : Example Petri Net of a Machine Model

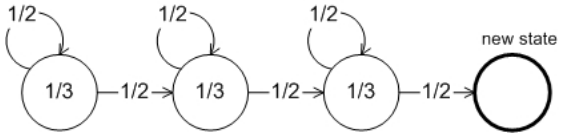
If additional information about the transitions is available, these should be initialized as good as possible, since the training algorithm used later is dependent on the initial model conditions. Furthermore, output symbols with probabilities need to be specified on the basis of the states of the system. This can also happen on the reachability graph of the Petri net, where the discrete states of the system are explicitly visible. The reachability graph with specified output probabilities for the output symbols *0* and *1* and two exponential distributions as initial parameterization is shown in Figures 3.




Figures 3 : Initialized Reachability Graph of Example Petri Net

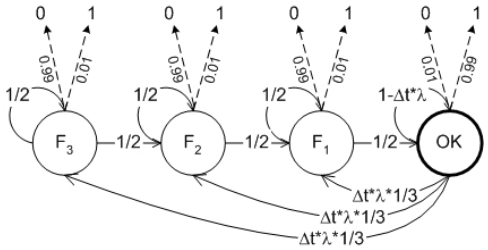
**Replacing General Distributions**

When all timed transitions of the SPN have an exponential distributions, as in GSPNs (Bobbio et al. 1998), then the reachability graph of the Petri net is equivalent to a Markov chain. In order to map the non-Markovian transitions to a Markov chain, one can replace them by so called discrete phase-type distributions (DPH). We use one minimal structure, since any acyclic DPH of the same order (number of phases) can be transformed into one of this specific structure with the minimum necessary number of parameters (Bobbio et al. 2002). Figures 4 shows a DPH of order three with uniformly chosen initial parameters. Theoretically the condition holds, that the larger the order of the DPH is, the better is the fit of the general distribution. Nevertheless does the number of phases probably have an influence on the time needed for the fit, or the amount of data needed, since by adding one phase, the number of independent variables to be trained is increased by two. Furthermore do some types of distributions require fewer phases to be fit accurately than others (Isensee and Horton 2005).




Figures 4 : Example DPH with Arbitrary Parameters

Some discrete states of the model will be replaced by several states of the DTMC. Since the probabilities of the output symbols are specified on the discrete state level of the SPN, these same values can just be copied to each corresponding state in the DTMC. The resulting DTMC for the model (Figures 3) and the DPH (Figures 4) is shown in Figures 5, assuming that the transition $T_{Fail}$ is exponential and $T_{Repair}$ is non-Markovian. The rates of the exponential distributions were transformed into transition probabilities by discretizing them with a chosen time step $\Delta t=0.1$.

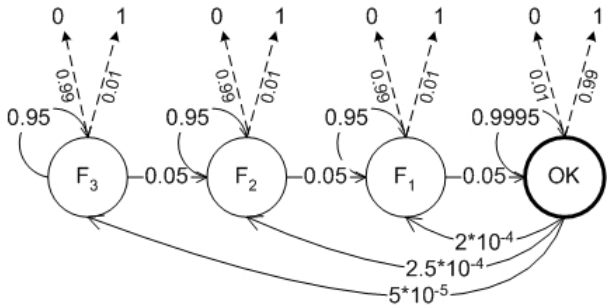



Figures 5 : Resulting HMM of Example Petri Net

**Baum-Welch**

The resulting DTMC with output probabilities for certain symbols can be interpreted as a hidden Markov model. One can now use the existing Baum-Welch Algorithm to train the HMM to produce the given output sequence or set of output sequences with a maximum probability. As opposed to the applications of HMM so far, the roles of the states in the DTMC need to stay fixed during the fitting process. Only if this condition is met, one can find and extract the DPH parameters from the trained model. Since the Baum-Welch Algorithm is an unsupervised training algorithm, it can also change the roles of DTMC states, which is a known property of the algorithm.

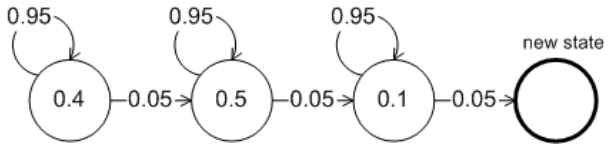
An existing implementation of the Baum-Welch Algorithm was modified to preserve the output probabilities. This was found necessary in some initial experiment to fix the roles of the states. The system structure is also left intact by not changing any zero entries in the transition probability matrix of the DTMC. This means, that no new state transitions are introduced into the model. Whether existing state transitions are preserved or might be destroyed in the trained model, needs to be investigated further. A deleting of transitions would significantly change the structure of the model and would complicate interpretation if not make it impossible. The result of the training process could be a HMM as shown in Figures 6.




Figures 6 : Trained Example HMM

**Extraction**

The extraction of the DPH from the trained HMM is basically the reverse step of the replacement of general transitions. If the model structure and state roles have been preserved during the previous training process, the location of the DPH in the DTMC is known, and its parameters can be extracted by normalizing the incoming transition probabilities. The extracted DPH of the trained HMM in Figures 6 is shown in Figures 7.

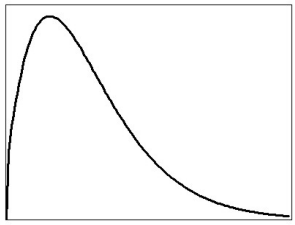



Figures 7 : Extracted DPH from Example HMM

The reachability graph of the original model can contain several state transitions that refer to the same transition in the Petri net. If these are replaced by DPH, the resulting trained DPH need to be combined in order to find a representation for the Petri net transition. This could for example be done by weighting the different fits parameters.

**Backfitting**

The last step in determining the parameterization of the general transitions in the Petri net is to investigate the time to absorption in the extracted DPH. The shape of the output of the example DPH in Figures 7 is shown in Figures 8. The shape clearly resembles a PDF (probability density function) of a Weibull distribution.



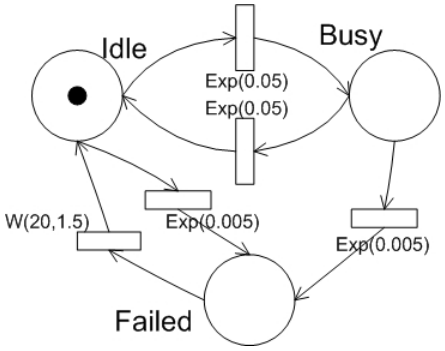
Figures 8 : Shape of Time to Absorption in Example DPH

The more interesting analysis is to try and match the phase-type distribution to a known general distribution function with specific parameters. Assuming a discretization time step of $\Delta t=0.1$, which was also used for the discretization, the closest match is a Weibull distribution with a scale parameter of approximately $\alpha=5$ and a shape parameter of about $\beta=1.5$. This finding of distribution type and parameters can be done by using common input modeling methods.

Here the optimization algorithm for the fitting of the DPH parameters, described in (Isensee and Horton 2005), was adapted and used to find the parameters of a general distribution function for a specific DPH output. The algorithm can also perform the fit for several known distribution types, compare the resulting error values and return the most likely distribution type and parameters. The distribution can now be used for the corresponding transition in the initial Petri net.

**EXPERIMENTS**

The experiments were performed with the model shown in Figures 9. The Petri net shows a web server that can be either working in the states *Idle* or *Busy* and that can be in state *Failed*, and under repair. The output that the system produces is actually the response to a ping request sent once every time unit. To the outside user in the web the actual state of the web server is not visible, he can only observe the answer to the ping request. By analyzing the observed traces he tries to determine the actual probability or distribution characteristics of the server being offline.

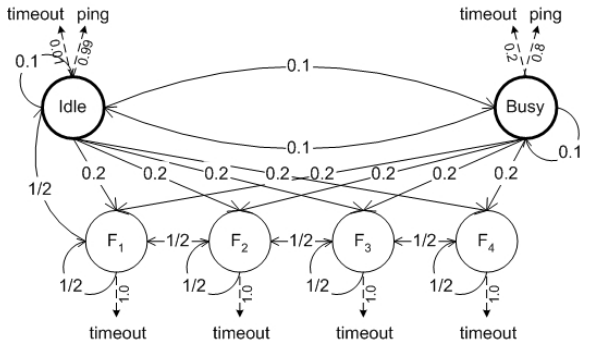



Figures 9 : Petri Net of Web Server Model

It is known, that in state *Idle* the web server responds with a probability of *0.99* and in state *Busy* only with a probability of 0.8. When it is *Failed*, the web server does not answer at all, so the probability for a timeout is 1.0. Only the repair time is assumed to have a non-Markovian distribution.

The output sequences for the training of the model were created by running a discrete event simulation of model using the parameters shown in Figures 9. According to the state of the model the simulation produces an output symbol every $\Delta t=1$ time units, and records it in an output file. Five stochastically independent runs were performed to produce five independent output sequences. They have a maximum possible length of *T=1,410,100* symbols.

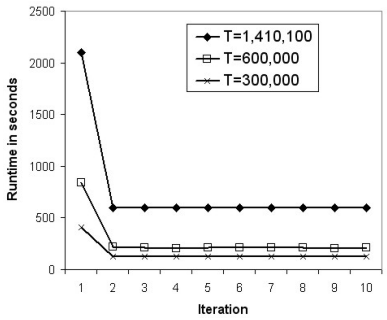
The reachability graph of the Petri net was turned into a Markov chain by replacing the general transition with a phase-type distribution of order four. The initial parameterization of the HMM for the model training process was chosen as shown in Figures 10. The initial state probabilities were set uniformly to *1/6.*




Figures 10 : Initial Parameters of Example HMM

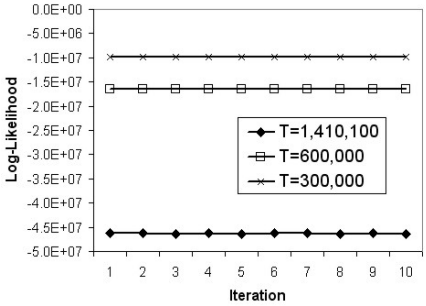
This initial model was trained iteratively using the Baum-Welch Algorithm with fixed output probabilities. Even though the algorithm itself is an iterative algorithm (iterations are called epochs), one training run did not result in a good fit, but several successive calls of the algorithm improved the model fitness. The chosen implementation terminates when the relative difference between the current likelihood and the mean likelihood of the previous fits falls below a threshold of *1e-6.* It was not tested, whether a stricter stop criterion would eliminate the need of several successive runs.

The experiments were performed using the five stochastically independent output sequences of different length. *Exp1* used the maximum possible length of *T=1,410,100*, *Exp2* used traces of length *T=600,000* and *Exp3* length *T=300,000*. The results for the processing time of the first 10 iterations are shown in Figures 11. The runtime decreases significantly with the length of the symbol sequences, as expected. The average runtime for *Exp1* is 6*00s*, for *Exp2 200s* and for *Exp3* about *120s*. The first iteration is more expensive in all cases, since this trains an arbitrarily initialized model, which requires more so-called epochs.



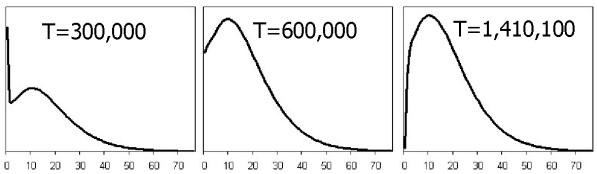

Figures 11 : Runtime per Iteration of Baum-Welch Algorithm Using Different Trace Length

The development of the log-likelihood through the first 10 iterations is shown in Figures 12. A larger absolute value represents a lower log-likelihood. The first call of Baum-Welch leads to an initially steep improvement in the initial log-likelihood of *4e-1*, which is not shown in the graph, since this contains the resulting trained models log-likelihood values. The log-likelihood for the longer traces is much lower than that of the shorter ones, since it is computed from a product of more probability values. The further iterations do not seem to improve the log-likelihood of the models, but the resulting fits for the non-Markovian distributions do. The cycling through the traces decreases the danger of getting stuck in a local minimum for all of the three experiments.




Figures 12 : Log-Likelihood for Successive Baum-Welch Iterations for Different Trace Length

The shape of the non-Markovian distribution for the repair time after 50 iterations can be seen in Figures 13. The fit that resulted from the training with the longest traces is the best one. It already resembles the Weibull distribution, which was used for creating the output symbol sequences. Further experiments showed that about 100 iterations were needed to find a good approximation for the W(20,1.5) distribution for the longest traces in *Exp1*. For the medium length traces *Exp2* about 300 iterations were needed to find a good fit. The shortest traces in *Exp3* did not yield comparable fits to the other experiments after 300 iterations, and did not improve further.




Figures 13 : Fits for Repair Time after 50 Baum-Welch Iterations with Different Sequence Length

This behavior of the algorithm is due to the following fact. The longer the traces, the more information is contained in them. The shortest traces obviously do not carry enough to train the tested model. The training using medium length traces needed more iterations, but about as much time in total as the training with the long traces.

The experiments show, that the approach above described does work. It is possible to train non-Markovian models using the methods of HMM. The use

of several output traces instead of just one increases the probability to find a globally good solution.

## CONCLUSION

### Summary

This paper introduced an approach to find the parameters of non-Markovian stochastic Petri nets using methods of hidden Markov models, specifically the Baum-Welch algorithm. By replacing the state transition by a DPH and training its parameters, a known mathematical distribution function can be found for the transition in the Petri net. This makes it possible to parameterize continuous stochastic processes by using some observable output of their hidden states, where before the hidden models were usually restricted to DTMCs. This expands the range of possible applications to more realistic models beyond speech recognition. The experiments showed that the approach works, and that only output sequences, which contain a sufficient amount of data, work for training the models.

### Outlook

The approach presented here is still new and opens up exciting new prospects and questions. Possible application areas of a working training method for non-Markovian models are any kind of hidden stochastic systems that can only by observed via their output: Disease recognition in a patient on the basis of the visible symptoms, consumer behavior estimation or failure protocols of running systems. Once trained these models could help in diagnosing, predicting behaviors and finding errors. Less spectacular, but also useful are applications, where noisy data is used for finding model parameters. The data does not need to be filtered, it can be used as is.

There are also several open issues regarding the training algorithm. The implementation of the training algorithm itself needs to be modified to be able to train with more than one trace, which was done artificially in this paper. More experiments are needed that test the feasibility on larger and more complex models. The replacing of several non-Markovian transitions will be needed for real life models. The merging of several trained DPH when a transition had to be replaced at several positions in a Markov chain is also interesting problem. The influence of the length of the DPH used for replacing on the training process and results is also interesting. Another open problem is finding the right amount of data needed for the training, so that it is sufficient and does not slow the training process too much. It needs to be tested whether it is possible to also train the output probabilities of the model. Under which conditions the Baum-Welch algorithm preserves the model structure also needs to be investigated.

## REFERENCES


Baum, L.E., T. Petrie, G. Soules, N. Weiss. 1970. “A Maximization Technique in the Statistical Analysis of Probabilistic Functions of Markov Chains”. In *The Annals of Mathematical Statistics*, vol.41, no.1, pp.164-171.

Bobbio, A., A. Puliafito, M. Telek, K. S. Trivedi. 1998. “Recent Developments in Non-Markovian Stochastic Petri Nets”. In *Journal of Systems Circuits and Computers*, vol. 8, no. 1, pp. 119-158.

Bobbio, A., A. Horváth, and M. Telek. 2002. “The scale factor: A new degree of freedom in phase type approximation”. In *Proceedings of 3rd International Performance & Dependability Symposium* (June)

Isensee, C. and G. Horton. 2005. “Approximation of Discrete Phase-Type Distributions”. In *Proceedings of the 38th Annual Simulation Symposium 2005*, San Diego, USA (Apr).

Neuts, M. F. 1981. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. The John Hopkins University Press, Baltimore.

Rabiner, L. R.. 1989. “A tutorial on hidden Markov models and selected applications in speech recognition”. In *Proceedings of the IEEE.* vol.77, no.2, pp.257–286, (Feb).

Russel, M. J. and R. K. Moore. 1985. “Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition”. In *Proceedings of the ICASSP’85*, Tampa, Florida. pp. 5-8 (Mar).

Russell, M. J. and A. E. Cook, “Experimental evaluation of duration modelling techniques for automatic speech recognition,” in *Proc. ICASSP, 1987*, pp. 2376–2379.

Wickborn, F., C. Isensee, T. Simon, S. Lazarova-Molnar, G. Horton. 2006. “A New Approach for Computing Conditional Probabilities of General Stochastic Processes”. Accepted to *39th Annual Simulation Symposium 2006*, Huntsville, USA (Apr).


## AUTHOR BIOGRAPHIES


**CLAUDIA ISENSEE** studied Computer Science at the Otto-von-Guericke-Universität Magdeburg and spent an exchange year at the University of Wisconsin, Stevens Point, where she graduated in 2002. She was awarded her German University diploma in September 2003. Since October 2003 she is working at the “Lehrstuhl für Simulation” at the Otto-von-Guericke-Universität Magdeburg. Her e-mail address is: `claudia@sim-md.de.`

**FABIAN WICKBORN** studied Computer Science at the Otto-von-Guericke-Universität Magdeburg. He was awarded her German University diploma in December 2004. Since January 2005 he is working at the “Lehrstuhl für Simulation” at the Otto-von-Guericke-Universität Magdeburg. Besides his work, he enjoys singing and being an aviator. His e-mail address is: `fabian@sim-md.de.`

**GRAHAM HORTON** studied Computer Science at the University of Erlangen, obtaining his Masters degree ("Diplom") in 1989. He obtained his PhD in Computer Science in 1991 and his "Habilitation" in 1998 at the same university, in the field of simulation. Since 2002, he is Professor for Simulation and Modelling at the Computer Science department of the University of Magdeburg. His email address is: `graham@sim-md.de.`