Hugo Lewi Hammer

Topics in stochastic simulation, with an application to seismic inversion

Thesis for the degree of philosophiae doctor

Trondheim, February 2008

Norwegian University of
Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of Science and Technology

Thesis for the degree of philosophiae doctor

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Mathematical Sciences

ISBN 978-82-471-7426-5 (printed ver.)
ISBN 978-82-471-7443-2 (electronic ver.)
ISSN 1503-8181

Theses at NTNU, 2008:73

Printed by Tapir Uttrykk

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of *philosophiae doctor* (PhD) at the Norwegian University of Science and Technology (NTNU). The research was funded by the Department of Mathematical Sciences, NTNU.

First of all I would like to thank Håkon Tjelmeland for being such an excellent supervisor and always helpful person. Thanks to all my colleagues at the Department of Mathematical Sciences and especially to the members of the statistics group. The administrative staff, and in particular Anne Kajander, has been very helpful. A special thanks to the computer support group, and in particular Per Kristian Hove, for providing state of the art computer system and always being helpful whenever needed. Also thanks to all the graduate students and postdocs at the department for making it a nice place to work.

I would also like to thank my friends all around Norway and in particular my friends in Trondheim during the PhD period: Håvard Berland, Steinar Kragset, Kjetil Midthun, Geir Solgaard and Jofrid Vårdal. I will, among many other things, always remember our beautiful mountain trips. I am grateful to my family at home for all your love and support and being so proud of me. Finally I am returning home :-)

Hugo Lewi Hammer

Trondheim, February 2008

Thesis Outline

The thesis consists of the following papers, which can be read independently of each other, though it is natural to read Paper III before Paper IV.

I **Control variates for the Metropolis–Hastings algorithm**.
With Håkon Tjelmeland. Accepted for publication in *Scandinavian Journal of Statistics.*

II **Directional Metropolis–Hastings algorithms on hyperplanes**.
With Håkon Tjelmeland. Report.

III **Approximate forward–backward algorithm for a switching linear Gaussian model – with application to seismic inversion**.
With Håkon Tjelmeland. Report.

IV **Empirical comparison of two Bayesian lithology–fluid prediction algorithms**.
With Marit Ulvmoen. Report.

Paper I to III are written for a statistical audience, whereas Paper IV is primarily directed to geoscientists.

Background

When doing stochastic modelling we typically need to evaluate integrals of the form

$$\mathrm{E}(f(\mathbf{x})) = \int f(\mathbf{x})\pi(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{1}$$

where $\mathbf{x}$ is a stochastic vector of dimension n, $f(\cdot)$ a function and $\pi(\cdot)$ a probability density. There are typically three directions to go in solving the integral, analytically, numerically or by stochastic simulation.

Finding an analytic solution to the integral is naturally the best. There exist several integrands where this is possible, at least when $\mathbf{x}$ is of low dimension. When $\mathbf{x}$ is of high dimension possible alternatives are for example the multivariate normal distribution (Johnson and Wichern, 1998) and Hidden Markov models (HMM) in combination with the Forward–backward (FB) algorithm (Scott, 2002). For HMM the resulting posterior distribution will be a non-stationary Markov chain and the transition probabilities can be computed exactly and efficiently using the FB algorithm.

Numerical techniques, like approximating the integral with a Riemann sum, work only for low dimensions of $\mathbf{x}$ since the number of evaluations of the integrand increases exponentially with the dimension of $\mathbf{x}$. For many situations, stochastic simulation is the only alternative. In stochastic simulation the integral is typically estimated by the sample mean

$$\widehat{\mathrm{E}(f(\mathbf{x}))} = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i), \tag{2}$$

where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are realizations from $\pi(\cdot)$. There exist both exact and approximate simulation techniques. The exact techniques are normally only available for low dimensions of $\mathbf{x}$. Importance sampling (Liu, 2001) is an important class of simulation techniques and there exist both approximate and exact versions. Exact versions are typically only available for low dimension. For higher dimensions there exist approximate sequential techniques and in particular particle filtering (Gordon et al., 1993) applied on non-linear HMM, also referred to as state space models, is popular.

A general and frequently used simulation technique is Markov chain Monte Carlo (MCMC) simulation and in particular the Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970). The MCMC algorithms simulate from a stationary Markov chain which has $\pi(\cdot)$ as its limiting distribution. The algorithms are in theory able to simulate from almost any distribution. Among disadvantages with the algorithms are that the realizations are dependent, and from $\pi(\cdot)$ only in the limit when the number of iterations goes to infinity. Evaluation of the convergence and mixing properties of the constructed Markov chain are important but often a challenging task. In the early days of MCMC the focus where on finding general algorithms that worked for a large set of distributions. Typical examples where algorithms that made small moves in samples space in each iteration like single site and random walk algorithms. Later, as one wanted to consider more and more complicated distributions, algorithms became more and more specialised for the application and distribution in question. There exists a large amount of different MCMC algorithms. Below we mention some of the most central.

In the Gibbs sampler (Geman and Geman, 1984; Gelfand and Smith, 1990) the variables are updated from the full conditionals $\pi(x_i|\mathbf{x}_{-i})$, where $\mathbf{x}_{-i}$ are all the variables in $\mathbf{x}$ except x_i. The algorithm is also referred to as the single site Gibbs sampler. Alternatively one may jointly update more than one component of $\mathbf{x}$ from the resulting full conditional. Such algorithms are referred to as the block Gibbs sampler. The Gibbs sampler can be seen as a special case of the MH algorithm, where the full conditionals are proposal distributions with resulting acceptance probability one.

In auxiliary MCMC algorithms an auxiliary variable $\mathbf{s}$ is introduced and a joint distribution $\pi(\mathbf{x}, \mathbf{s})$ is defined so that the marginal distribution for $\mathbf{x}$ still are distributed according to the distribution of interest, $\pi(\cdot)$. The auxiliary variable $\mathbf{s}$ is included to simplify the definition of large updates. A typical example would be to include an $\mathbf{s}$ such that $\pi(\mathbf{x}|\mathbf{s})$ and $\pi(\mathbf{s}|\mathbf{x})$ are possible to sample from so that a block Gibbs sampler can be used. The Swendsen–Wang algorithm (Swendsen and Wang, 1987) for example using this idea. Other examples of auxiliary MCMC algorithms are simulated tempering (Marinari and Parisi, 1993; Geyer and Thompson, 1995), multi–grid sampling (Goodman and Sokal, 1989) and slice sampling (Edwards and Sokal, 1988; Neal, 2003).

An important class of MCMC algorithms is what is referred to as reversible jump algorithms (Green, 1995; Waagepetersen and Sørensen, 2001). For these algorithms, the proposals are generated through a parametrized deterministic function. This scheme is mostly used to simulate from distributions of varying dimension. A class of algorithms that is using the reversible jump setup for distributions of fixed dimension is directional MH algorithms. Here, the proposals are generated along lines in sample space. By making proposals in other directions than defined by the coordinate algorithms with better convergence and mixing properties can be obtained. Here the parameter in the algorithm represents the direction of the line. There exist several direction MH algorithms. For example Chen and Schmeiser (1993) generate proposals along lines with uniform directions, Roberts and Rosenthal (1998) present an algorithm that moves along a direction centered at the gradient direction evaluated in the current point. Eidsvik and Tjelmeland (2006) also let the direction be a function of the current point.

Single site MCMC algorithms work poorly in terms of convergence and mixing properties whenever strong dependency is present between any of the components in $\mathbf{x}$. Proposing changes for many variables simultaneously in a MCMC algorithm potentially results in better algorithms. Such algorithms are typically referred to as block or grouping MCMC algorithms. For example, Liu et al. (1994) argue that doing block updates work better than single site updates when it comes to the Gibbs sampler and there is strong dependence in $\mathbf{x}$. Often models are constructed such that full conditionals for large blocks are either multivariate Gaussian or such that the FB algorithm can be used. Alternatively auxiliary variables also are used to achieve this. To avoid very low acceptance rates for block MH algorithms the proposals are typically generated from approximations to the full conditionals.

Computer power is often a limiting factor when it comes to estimation of $\mathrm{E}(f(\mathbf{x}))$ using stochastic simulation. There have been developed some techniques to reduce the variance of the estimate of $\mathrm{E}(f(\mathbf{x}))$ without increasing the amount of realizations generated and thereby save computer time. Typical examples are Rao–Blackwellization, antithetic variables and the use of control variates (Liu, 2001). There also exist examples where variance reduction techniques are used inside the MCMC framework. Casella and Robert (1996) makes use of Rao–Blackwellization and Pinto and Neal (2001), Mira et al. (2003) and Atchadé and Perron (2005) construct control variates.

Summary

In the following we summarise the four papers in this thesis.

Paper I constructs control variates for variance reduction in the estimation of mean values using the MH algorithm. Traditionally, mean values are estimated by the sample mean of the accepted states from the MH algorithm. The rejected proposals are not used in the estimation and this seems like a waste of information. We present a framework for construction of zero mean control variates which are a function of all the accepted and rejected states. The control variates can generally be used for the HM algorithm and for the reversible jump algorithm. We are able to construct the zero mean control variates by letting the variates be a weighted sum of the current state and the proposal in each iteration. We also develop control variates that in addition to the current and the potential new state are a function of the acceptance /rejection decision. In our simulation examples, we get the highest variance reduction for the control variate that is a function of just the accepted and rejected states. The variance reductions are typically between 15% and 35%, but in one of our examples we get variance reductions as high as 45%.

The algorithm considered in Paper II generalizes the direction MH algorithm in Eidsvik and Tjelmeland (2006) by generating proposals in hyperplanes in stead of along lines. The generalized algorithm is compared with the algorithm in Eidsvik and Tjelmeland (2006) and more traditional MH algorithms in a simulation example. The generalized algorithm typically proposes longer jumps then the other algorithms and this indicates better mixing properties. However, with our implementation the generalised algorithm are more computation intensive per iteration and so the simpler algorithms in most cases are better when run for the same amount of computer time. An interesting area for future research is therefore to find directional Metropolis–Hastings algorithms that generate proposals in hyperplanes, but requires less computation time per iteration.

Paper III considers a Hidden Markov model with two hidden layers. The model, which we call switching linear Gaussian (SLG) model, has its origin from the problem of seismic

inversion. The bottom layer of the model is a Markov chain and the intermediate layer and the observations are Gaussian. The model has clear similarities to what is known as switching linear dynamical systems and conditionally Gaussian state space model (Zoeter and Heskes, 2006; Bar-Shalom and Li, 1998; Barber, 2006). Because of the Markovian structure of the SLG model, the FB procedure can be used, but the computational efficiency of traditional FB algorithms are not present for the SLG model. The forward recursion involves a mixture of Gaussian terms where the number of terms grows exponentially when iterating forward. We are not able to handle this many terms on a computer and need to introduce an approximation by removing less important terms. Related to the problem of seismic inversion, the SLG model is a part of a seismic inversion (SI) model. The bottom and the intermediate layer of the SLG model, represent LF classes and elastic material parameters, respectively, along a vertical profile in a reservoir. The SI model contains an additional layer connecting the elastic parameters of the SLG model with pre–stack seismic data. The SI model is closely related to the models presented in Buland et al. (2003) and Larsen et al. (2006). To construct an efficient simulation algorithm for the SI model, we include the uppermost layer of the SLG model as an additional layer in the SI model. The layer can be seen as auxiliary variables for the simulation algorithm. We simulate from the SI model using MCMC simulation, where new values for the LF classes and elastic parameters are proposed using the approximate FB algorithm described above. In our simulation examples, the resulting MCMC algorithm converges fast and seems to have very favorable mixing properties.

Paper IV compare inversion results from the SI model MCMC algorithm in Paper III with inversion results from the simulation algorithm in Larsen et al. (2006). The algorithm Larsen et al. (2006) introduces an approximation in the likelihood part of the SI model and is in this way able to evaluate the resulting approximate posterior exactly and very efficiently. The consequences of the approximations introduced in the likelihood are on the other hand not clear and the objective of Paper IV is to evaluate the consequences. We achieve this by comparing the inversion results from the posterior distribution and the approximate posterior distribution in a synthetic but realistic case study. Central in Paper IV is how to best compare discrete distributions of high dimensions. By using several measures on the posterior marginal distributions we succeed in getting a good understanding of the consequences of the approximations. We observe that, dependent on the parameters in the model, typically will the approximate likelihood model preserves between 55 and 80% of the information in the original likelihood model. The consequences of the approximation increase when the amount of noise in the model increases. The approximation work better when most of the variability is in the rock physics model and it is little seismic noise, compared to the opposite.

References

Atchadé, Y. F. and Perron, F. (2005). Improving on the independent Metropolis-Hastings algorithm, *Statistica Sinica* **15**: 3–18.

Bar-Shalom, Y. and Li, X.-R. (1998). *Estimation and Tracking: Principles, Techniques and Software*, Artech House, Norwood, MA.

Barber, D. (2006). Expectation correction for smoothed inference in switching linear dynamical systems, *Journal of machine learning research* **7**: 2515 – 2540.

Buland, A., Kolbjørnsen, O. and Omre, H. (2003). Rapid spatially coupled AVO inversion in the Fourier domain, *Geophysics* **68**: 824–836.

Casella, G. and Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes, *Biometrica* **83**: 81–94.

Chen, M. and Schmeiser, B. (1993). Performance of the Gibbs, hit-and-run and Metropolis samplers, *Journal of computational and graphical statistics* **2**: 251–272.

Edwards, R. G. and Sokal, A. D. (1988). Generalization of the Fortuin–Kasteleyn–Swendsen–Wang representation and Monte Carlo algorithms, *Physical Review D* **38 issue 6**: 2009–2012.

Eidsvik, J. and Tjelmeland, H. (2006). On directional Metropolis–Hastings algorithms, *Statistics and Computing* **16**: 93–106.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling based approaches to calculating marginal densities, *Journal of the American Statistical Association* **85**: 398–409.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restortian of images, *IEEE Transaction Pattern Analysis and Machine Intelligence* **6**: 721–741.

Geyer, C. and Thompson, E. (1995). Annealing Markov chain Monte Carlo with application to ancestral inference, *Journal of the American Statistical Association* **90**: 909–920.

Goodman, J. and Sokal, A. (1989). Multigrid Monte Carlo methods: Conceptual foundations, *Physical Review D* **40 issue 6**: 2035–2071.

Gordon, N., Salmond, J. and Smith, A. (1993). A novel approach to non–linear/non–Gaussian Bayesian state estimation, *IEEE Proceedings on Radar and Signal Processing* **140**: 107–113.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computations and Bayesian model determination, *Biometrica* **57**: 97–109.

Hastings, W. (1970). Monte Carlo sampling using Markov chains and their applications, *Biometrica* **57**: 97–109.

Johnson, R. A. and Wichern, D. W. (1998). *Applied Multivariate Statistical Analysis*, Prentice-Hall.

Larsen, A. L., Ulvmoen, M., Omre, H. and Buland, A. (2006). Bayesian lithology/fluid prediction and simulation on the basis of a Markov-chain prior model, *Geophysics* **71 issue 5**: R69–R78.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*, Springer, Berlin.

Liu, J. S., Wong, W. H. and Kong, A. (1994). Covariance structures and convergence rate of Gibbs sampler with various scans, *Journal of the Royal Statistical Society Series B* **57 issue 1**: 157–169.

Marinari, E. and Parisi, G. (1993). Simulated tempering: a new Monte Carlo scheme, *Europhysics Letters* **19 issue 6**: 451–458.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953). Equations of state calculations by fast computing machines, *Journal of Chemical Physics* **21 issue 6**: 1087–1092.

Mira, A., Tenconi, P. and Bressanini, D. (2003). Variance reduction in MCMC, *Technical Report 29/2003*, Department of Economics, University of Insubria, Italy.

Neal, R. (2003). Slice sampling (with discussion), *Annals of Statistics* **31**: 705–767.

Pinto, R. L. and Neal, R. M. (2001). Improving Markov chain Monte Carlo estimators by coupling to an approximating chain, *Technical Report No. 0101*, Department of Statistics, University of Toronto.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions, *Journal of the Royal Statistical Society. Series B* **60**: 255–268.

Scott, A. L. (2002). Bayesian methods for hidden Markov models: Recursive compution in the 21st century, *Journal of the American Statistical Association* **97**: 337–351.

Swendsen, R. H. and Wang, J. S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations, *Physical Review Letters* **58 issue 2**: 86–88.

Waagepetersen, R. and Sørensen, D. (2001). A tutorial on reversible jump MCMC with a view toward application in QTL-mapping, *International Statistical Review* **69**: 49–61.

Zoeter, O. and Heskes, T. (2006). Deterministic approximate inference techniques for conditionally Gaussian state space models, *Statistics and Computing* **16**: 279–292.

Paper I

Control variates for the Metropolis–Hastings algorithm

Hugo Hammer and Håkon Tjelmeland
Department of Mathematical Sciences
Norwegian University of Science and Technology
Trondheim, Norway

Abstract

We propose new control variates for variance reduction in estimation of mean values using the Metropolis–Hastings algorithm. Traditionally, states that are rejected in the Metropolis–Hastings algorithm are simply ignored, which intuitively seems to be a waste of information. We present a setting for construction of zero mean control variates for general target and proposal distributions and develop the ideas for the standard Metropolis–Hastings and the reversible jump algorithms. We give results for three simulation examples. We get best results for variates that are functions of the current state $\mathbf{x}$ and the proposal $\mathbf{y}$, but we also consider variates that in addition are functions of the Metropolis–Hastings acceptance/rejection decision. The variance reduction achieved varies depending on the target distribution and proposal mechanisms used. In simulation experiments we typically achieve relative variance reductions between 15% and 35%.

Key words: Control variate, Markov chain Monte Carlo, rejected states, variance reduction.

1 Introduction

Suppose we want to estimate the mean, μ, of a function $f(\mathbf{x})$ when $\mathbf{x} \in \mathbb{R}^n$ is distributed according to a target distribution $\pi(\cdot)$. If $\mathbf{x}$ is of high dimension and $\pi(\cdot)$ is complex, Markov chain Monte Carlo (MCMC) techniques are typically the only viable alternatives for evaluating μ. In the Metropolis–Hastings algorithm (Hastings, 1970), each iteration consists of two steps. Letting $\mathbf{x}$ denote the current state of the Markov chain, one first proposes a new state $\mathbf{y}$. Second, one accepts $\mathbf{y}$ with a probability, otherwise retaining $\mathbf{x}$. After a burn in period the current state is essentially from the target distribution $\pi(\cdot)$ and can be used to estimate μ. The most natural and traditional estimator of μ is the empirical mean of $f(\mathbf{x})$.

It is always possible to do better than the empirical mean when it comes to estimation variance (Liu, 2001). In the *antithetic variates* method (Hammersley and Morton, 1956) one uses samples with negative correlation. The variance of the empirical mean is then less than with independent samples. The *Rao-Blackwellization* method uses that $\mathrm{E}[\mathrm{E}(f(\mathbf{x})|z)] = \mathrm{E}(f(\mathbf{x}))$ and $\mathrm{Var}[\mathrm{E}(f(\mathbf{x})|z)] \leq \mathrm{Var}(f(\mathbf{x}))$ for any random variable z, implying that if $\mathrm{E}(f(\mathbf{x})|z)$ is analytically available, the empirical mean of this has less variance than the empirical mean of $f(\mathbf{x})$. The *control variate* method constructs estimators that are linear combinations of the original unbiased estimator and a random variable with known mean, called control variate.

Variance reduction techniques can also be used in a Metropolis–Hastings setting. Casella and Robert (1996) develop a Rao-Blackwellization technique for this situation, but the cost for evaluating the resulting estimator is quadratic in the number of iterations. Pinto and Neal (2001) use the same sequence of random numbers in two Markov chains, one sampling from $\pi(\cdot)$ and one from a Gaussian approximation to $\pi(\cdot)$. The latter chain is used to construct a control variate. Large variance reductions results if $\pi(\cdot)$ is close to the Gaussian distribution. Mira et al. (2003) use a zero-variance principle from physics literature to construct an estimator that is the sum of the original estimator $f(\mathbf{x})$ and a zero mean term. The additional term can be considered a control variate. The method is only explored for simple low dimensional cases, so the potential for more complicated situations is unclear. Atchadé and Perron (2005) construct control variates that are functions of all proposed states, but limit the attention to the independent proposal case.

In the present paper we use the control variate technique and consider variates that are functions of both the current state $\mathbf{x}$ and the proposal $\mathbf{y}$. We introduce a scheme for construction of zero mean variates for general target and proposal distributions and develop the idea for both the standard Metropolis–Hastings and the reversible jump algorithms. In simulation examples we try both control variates that are functions of only $\mathbf{x}$ and $\mathbf{y}$ and variates that also are functions of the acceptance/rejection decision. We get the highest variance reductions for the simpler control variates that are functions only of $\mathbf{x}$ and $\mathbf{y}$.

The paper is organised as follows. In Section 2 we give a brief introduction to the Metropolis–Hastings algorithm, and in Section 3 we describe the control variate technique. The new control variates are developed in Section 4 and simulation examples are presented in Section 5. Finally, in Section 6 we provide closing remarks.

2 MCMC simulation

The aim is to generate samples from a target distribution $\pi(\cdot)$. The basic idea in MCMC is to construct a Markov chain with $\pi(\cdot)$ as limiting distribution (Liu, 2001). An algorithm typically has two steps in each iteration. Letting $\mathbf{x}$ denote the current state, one first proposes a new state $\mathbf{y}$ and, second, accepts $\mathbf{y}$ with a probability, otherwise keeping $\mathbf{x}$ as the current state. Let $\mathbf{x}^1, \ldots, \mathbf{x}^N$ denote N states of the Markov chain after having discarded a "burn-in" period, so that $\mathbf{x}^1, \ldots, \mathbf{x}^N$ are samples from $\pi(\cdot)$. Thus, with

$$\mu = E(f(\mathbf{x})) = \int f(\mathbf{x})\pi(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{1}$$

the most frequently used estimator for μ is the sample mean,

$$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}^i). \tag{2}$$

In the following we briefly describe three MCMC algorithms, starting with the standard Metropolis–Hastings scheme.

2.1 Standard Metropolis–Hastings algorithm

The Metropolis–Hastings setup is a recipe for construction of time reversible Markov chains with a given limiting distribution, see Smith and Roberts (1993) and Dellaportas and Roberts (2003). The standard Metropolis–Hastings algorithm follows the framework of a MCMC algorithm described above. Still letting $\mathbf{x}$ be the current state, we generate a proposal $\mathbf{y}$ from a distribution $q(\mathbf{y}|\mathbf{x})$ and accept $\mathbf{y}$ with a probability $\alpha(\mathbf{y}|\mathbf{x})$. Different choices are possible for $\alpha(\mathbf{y}|\mathbf{x})$. Peskun (1973) shows that it is optimal in terms of asymptotic variance to use

$$\alpha(\mathbf{y}|\mathbf{x}) = \min\{1, R(\mathbf{x},\mathbf{y})\} \quad \text{where} \quad R(\mathbf{x},\mathbf{y}) = \frac{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})}. \tag{3}$$

Another choice of $\alpha(\mathbf{y}|\mathbf{x})$ is what is known as the Barker (1965) acceptance probability,

$$\alpha(\mathbf{y}|\mathbf{x}) = \frac{R(\mathbf{x},\mathbf{y})}{1+R(\mathbf{x},\mathbf{y})}. \tag{4}$$

As the latter choice is suboptimal it is today essentially never used, but it has a connection to a control variate we introduce in this article. We discuss this further in Section 4.

2.2 Mode jumping algorithm

A variant of the Metropolis–Hastings setup is the mode jumping algorithm introduced in Tjelmeland and Hegstad (2001). It uses two proposal densities $q_0(\cdot|\mathbf{x})$ and $q_1(\cdot|\mathbf{x})$ and works as follows. First, $k = 0$ or 1 is randomly selected and potential new state $\mathbf{y}$ is generated from $q_k(\cdot|\mathbf{x})$. Second, we accept the proposed state $\mathbf{y}$ with probability

$$\alpha(\mathbf{y}|\mathbf{x}) = \min\{1, R(\mathbf{x},\mathbf{y})\} \quad \text{where} \quad R(\mathbf{x},\mathbf{y}) = \frac{\pi(\mathbf{y})q_{1-k}(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q_k(\mathbf{y}|\mathbf{x})}. \tag{5}$$

Otherwise we keep $\mathbf{x}$ as the current state.

2.3 Reversible jump algorithm

The reversible jump algorithm (Green, 1995; Waagepetersen and Sørensen, 2001) generalises the standard Metropolis–Hastings algorithm. It is most often used when the target distribution have a sample space of varying dimension.

The target distribution $\pi(\cdot)$ is the joint probability distribution of $\mathbf{x} = (m, \mathbf{z})$, where $m \in \{1, 2, \ldots, M\}$ is a model indicator and $\mathbf{z}$ a stochastic vector of possibly varying dimension. The $\mathbf{z}$ takes values in a set $C = \cup_{m=1}^{M} C_m$, where $C_m = \mathbb{R}^{n_m}$, $n_m \geq 1$. Given a value for m, $\mathbf{z}$ can only take values in C_m. Letting $\mathbf{x} = (m, \mathbf{z})$ with $\mathbf{z} \in C_m$ be the current state, one iteration works as follows. Propose a new state $\mathbf{y} = (m', \mathbf{z}')$ with $\mathbf{z}' \in C_{m'}$ by first proposing m' and a $\mathbf{u} \in \mathbb{R}^{n_{mm'}}$ from a proposal distribution $q(m', \mathbf{u}|\mathbf{x}) = p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})$, where $p_{mm'}$ is the distribution for m' and $q_{mm'}(\mathbf{u}|\mathbf{z})$ the distribution for $\mathbf{u}$. Next, a one-to-one deterministic relation is defined between $(\mathbf{z}, \mathbf{u})$ and $(\mathbf{z}', \mathbf{u}')$ for some $\mathbf{u}'$. The proposed new value $\mathbf{y} = (m', \mathbf{z}')$ is accepted with probability

$$\alpha(\mathbf{y}|\mathbf{x}) = \min\{1, R(\mathbf{x},\mathbf{y})\} \quad \text{where} \quad R(\mathbf{x},\mathbf{y}) = \frac{\pi(\mathbf{y})q(m, \mathbf{u}'|\mathbf{y})}{\pi(\mathbf{x})q(m', \mathbf{u}|\mathbf{x})}|J| \tag{6}$$

and J is the Jacobi determinant for the transformation from $(\mathbf{z}, \mathbf{u})$ to $(\mathbf{z}', \mathbf{u}')$. More details for this algorithm is given in the Appendix.

3 Control variates

Suppose we have samples $\mathbf{x}^1, \ldots, \mathbf{x}^N$ from the target distribution $\pi(\cdot)$ and these are used to estimate μ defined in (1) with an unbiased estimator μ^*. Typically μ^* is the sample mean given by (2). Suppose we have another random variable v, the control variate, with zero mean and which is correlated with μ^*. Then for any value c we can also use the unbiased estimator

$$\tilde{\mu} = \mu^* + c \cdot v. \tag{7}$$

This gives

$$\mathrm{Var}(\tilde{\mu}) = \mathrm{Var}(\mu^*) + c^2\mathrm{Var}(v) + 2c\mathrm{Cov}(\mu^*, v),$$

which is minimised for

$$c = -\frac{\mathrm{Cov}(\mu^*, v)}{\mathrm{Var}(v)}. \tag{8}$$

For this optimal value of c, the relative variance reduction by using $\tilde{\mu}$ in stead of μ^* is

$$\frac{\mathrm{Var}(\mu^*) - \mathrm{Var}(\tilde{\mu})}{\mathrm{Var}(\mu^*)} = [\mathrm{Corr}(\mu^*, v)]^2. \tag{9}$$

Thus, the aim is to construct a control variate which has a high squared correlation with μ^*. Clearly, it is also possible to use more than one control variate. In the next section we discuss ways to construct control variates for the Metropolis–Hastings algorithm.

4 Control variates for the Metropolis–Hastings algorithm

In this section we construct control variates for the Metropolis–Hastings algorithm. Let $\mathbf{x}^1, \ldots, \mathbf{x}^N$ and $\mathbf{y}^1, \ldots, \mathbf{y}^N$ denote the states of the Markov chain and the corresponding proposals in an MCMC algorithm, respectively.

4.1 A first control variate

Consider a control variate of the form

$$v = \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}^i, \mathbf{y}^i), \tag{10}$$

where $g(\cdot,\cdot)$ is a function to be specified. Then $\mathrm{E}(v) = 0$ if $\mathrm{E}(g(\mathbf{x},\mathbf{y})) = 0$, where the expectation is with respect to $(\mathbf{x},\mathbf{y}) \sim \pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})$. Define the function

$$g_0(\mathbf{x},\mathbf{y}) = w_1(\mathbf{x},\mathbf{y})f(\mathbf{x}) + w_2(\mathbf{x},\mathbf{y})f(\mathbf{y}), \tag{11}$$

where $w_1(\cdot,\cdot)$ and $w_2(\cdot,\cdot)$ are functions to be specified. We can then use $g_0(\cdot,\cdot)$ as $g(\cdot,\cdot)$ in (10) if

$$E(g_0(\mathbf{x},\mathbf{y})) = \iint [w_1(\mathbf{x},\mathbf{y})f(\mathbf{x}) + w_2(\mathbf{x},\mathbf{y})f(\mathbf{y})]\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} = 0. \tag{12}$$

To find $w_1(\cdot,\cdot)$ and $w_2(\cdot,\cdot)$ that satisfy (12), we write the above expression as a sum of two integrals and change the order of integration in the last integral. This gives the requirement

$$\iint w_1(\mathbf{x},\mathbf{y})f(\mathbf{x})\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} + \iint w_2(\mathbf{y},\mathbf{x})f(\mathbf{x})\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y} = 0. \tag{13}$$

Thus, a sufficient condition for $E(g_0(\mathbf{x},\mathbf{y})) = 0$ is

$$w_1(\mathbf{x},\mathbf{y})\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x}) = -w_2(\mathbf{y},\mathbf{x})\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y}), \tag{14}$$

which, for any function $\psi(\mathbf{x},\mathbf{y})$, is satisfied for

$$w_1(\mathbf{x},\mathbf{y}) = -\psi(\mathbf{x},\mathbf{y})\frac{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x}) + \pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})} \tag{15}$$

and

$$w_2(\mathbf{x},\mathbf{y}) = \psi(\mathbf{y},\mathbf{x}) \frac{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x}) + \pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}. \tag{16}$$

When choosing $\psi(\cdot,\cdot)$ it is natural to focus on $w_2(\mathbf{x},\mathbf{y})$, the "weight" assigned to the potential new state. Two choices are of particular interest. With $\psi(\mathbf{x},\mathbf{y}) = 1$ the weight assigned to $f(\mathbf{y})$ equals the Barker (1965) acceptance probability (4), the relative probability for the pair $(\mathbf{y},\mathbf{x})$, an intuitively reasonable choice. The control variate is then

$$g_0(\mathbf{x},\mathbf{y}) = \frac{R(\mathbf{x},\mathbf{y})}{1 + R(\mathbf{x},\mathbf{y})}(f(\mathbf{y}) - f(\mathbf{x})). \tag{17}$$

Another natural choice for $\psi(\cdot,\cdot)$ is

$$\psi(\mathbf{x},\mathbf{y}) = \min\left\{1 + \frac{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})}{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}, 1 + \frac{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})}\right\}, \tag{18}$$

which gives $w_2(\cdot,\cdot)$ equal to the Peskun (1973) optimal acceptance probability,

$$w_2(\mathbf{x},\mathbf{y}) = \alpha(\mathbf{y}|\mathbf{x}) = \min\left\{1, \frac{\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})}{\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})}\right\}. \tag{19}$$

Our experience in simulation examples is that (17) gives slightly larger variance reductions than the variate based on (19). In the examples in Section 5 we therefore only report results for (17). We end this section with some remarks.

Remark 1. With the same kind of reasoning as above we get the same control variates for the mode jumping and the reversible jump algorithms. In the Appendix we give a proof of (17) in the reversible jump setting.

Remark 2. The estimator corresponding to (18) can also be derived using Rao-Blackwellization. More specifically, the sample mean estimator can be improved by $(1/N)\sum_{i=1}^{N} E(f(\mathbf{x}^i)|\mathbf{x}^{i-1},\mathbf{y}^i)$ $= (1/N)\sum_{i=1}^{N}[(1-\alpha(\mathbf{y}^i|\mathbf{x}^{i-1})f(\mathbf{x}^{i-1}) + \alpha(\mathbf{y}^i|\mathbf{x}^{i-1})f(\mathbf{y}^i)$, where the expectation is over the acceptance/rejection decision. Note however that the above discussion shows that a $w_2(\mathbf{x},\mathbf{y})$ corresponding to the Barker acceptance probability can be used even if the Peskun optimal acceptance probability is used in the acceptance/rejection decision.

Remark 3. All we need to calculate $\tilde{\mu}$ is the current state $\mathbf{x}$, the proposal $\mathbf{y}$ and the acceptance ratio $R(\mathbf{x},\mathbf{y})$. These are already calculated when running the simulation algorithm. Thus, the extra computational cost by using $\tilde{\mu}$ in stead of $\widehat{\mu}$ is negligible.

4.2 Control variates including the acceptance indicator

The control variate (17) is based on the current state $\mathbf{x}$ and the potential new state $\mathbf{y}$. A natural next step is to consider control variates that depend also on the acceptance/rejection decision, or variates that are also functions of the proposal in the next iteration of the Markov chain. The above scheme can easily be generalised to these situations and in the following we consider the inclusion of the acceptance/rejection decision.

Let $\gamma \in \{0,1\}$ be the Metropolis–Hastings acceptance indicator, i.e.

$$\mathrm{P}(\gamma|\mathbf{x},\mathbf{y}) = [\alpha(\mathbf{y}|\mathbf{x})]^{\gamma}[1-\alpha(\mathbf{y}|\mathbf{x})]^{(1-\gamma)}, \quad \gamma = 0,1. \tag{20}$$

A variate that depends on both $\mathbf{x}$, $\mathbf{y}$ and γ would then be

$$v = \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}^i, \mathbf{y}^i, \gamma^i), \tag{21}$$

where

$$g(\mathbf{x},\mathbf{y},\gamma) = w_1(\mathbf{x},\mathbf{y},\gamma)f(\mathbf{x}) + w_2(\mathbf{x},\mathbf{y},\gamma)f(\mathbf{y}), \tag{22}$$

$\gamma^1,\ldots,\gamma^N$ are the acceptance indicators for each iteration, and $w_1(\cdot,\cdot,\cdot)$ and $w_2(\cdot,\cdot,\cdot)$ are functions to be specified. Proceeding similar to (12) and (13), but now also summing over the two values of γ, we get a sum of four integrals that has to equal zero. A sufficient condition for this is that the sum of the integrands is identical to zero. This gives

$$w_1(\mathbf{x},\mathbf{y},0)\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})(1-\alpha(\mathbf{y}|\mathbf{x})) + w_2(\mathbf{y},\mathbf{x},0)\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})(1-\alpha(\mathbf{x}|\mathbf{y})) \tag{23}$$

$$+w_1(\mathbf{x},\mathbf{y},1)\pi(\mathbf{x})q(\mathbf{y}|\mathbf{x})\alpha(\mathbf{y}|\mathbf{x}) + w_2(\mathbf{y},\mathbf{x},1)\pi(\mathbf{y})q(\mathbf{x}|\mathbf{y})\alpha(\mathbf{x}|\mathbf{y}) = 0.$$

Many solutions of this exists. In particular we may explore variates that result from setting two of $w_1(\mathbf{x},\mathbf{y},0)$, $w_2(\mathbf{x},\mathbf{y},0)$, $w_1(\mathbf{x},\mathbf{y},1)$ and $w_2(\mathbf{x},\mathbf{y},1)$ to zero and solve for the other two similarly to what we did to get (17). Assuming we use the Peskun optimal acceptance probability (3), two of the resulting six cases produce degenerate control variates. For arbitrary functions $\psi_i(\cdot,\cdot), i = 1,\ldots,4$, the remaining four cases generate the following control variates

$$\begin{aligned}
g_1(\mathbf{x},\mathbf{y},\gamma) &= \psi_1(\mathbf{x},\mathbf{y})(\alpha(\mathbf{y}|\mathbf{x})-\gamma)f(\mathbf{x}), && (24)\\
g_2(\mathbf{x},\mathbf{y},\gamma) &= (1-\gamma)\psi_2(\mathbf{x},\mathbf{y})\alpha(\mathbf{y}|\mathbf{x})f(\mathbf{x}) - \gamma\psi_2(\mathbf{y},\mathbf{x})(1-\alpha(\mathbf{x}|\mathbf{y}))f(\mathbf{y}), && (25)\\
g_3(\mathbf{x},\mathbf{y},\gamma) &= (1-\gamma)\psi_3(\mathbf{x},\mathbf{y})\alpha(\mathbf{y}|\mathbf{x})f(\mathbf{y}) - \gamma\psi_3(\mathbf{y},\mathbf{x})(1-\alpha(\mathbf{x}|\mathbf{y}))f(\mathbf{x}), && (26)\\
g_4(\mathbf{x},\mathbf{y},\gamma) &= \psi_4(\mathbf{x},\mathbf{y})(\alpha(\mathbf{y}|\mathbf{x})-\gamma)f(\mathbf{y}). && (27)
\end{aligned}$$

The $g_1(\cdot,\cdot,\cdot)$ is not a function of $f(\mathbf{y})$ and is thereby not achieving our goal of using all proposed states to estimate μ. The $g_2(\cdot,\cdot,\cdot)$ gives only weight to $f(\mathbf{y})$ when $\mathbf{y}$ is accepted and only to $f(\mathbf{x})$ when $\mathbf{x}$ is retained. Thus, the weight is given to the state that becomes $\mathbf{x}$ in the next iteration. Intuitively it seems better to give weight to the state that is left, and this is what $g_3(\cdot,\cdot,\cdot)$ is doing. The $g_4(\cdot,\cdot,\cdot)$ is only a function of $f(\mathbf{y})$ and is thereby achieving the goal of using all proposed states to estimate μ. Moreover, as $g_4(\cdot,\cdot,\cdot)$ is not a function of $f(\mathbf{x})$ one would expect the corresponding control variate to be less correlated with the variate defined in Section 4.1.

Remark 4. The choice of $\psi_i(\cdot,\cdot)$ will clearly influence the effectiveness of the approach. A simple and natural choice is to set $\psi_i(\cdot,\cdot) \equiv 1$. For the examples discussed in the next section we have tried both this and other choices. For particular $\pi(\cdot)$ and $f(\cdot)$ we have found choices of $\psi_i(\cdot,\cdot)$ that give slightly better results than $\psi_i(\cdot,\cdot) \equiv 1$, but this seems to be very problem specific. Thus, $\psi_i(\cdot,\cdot) \equiv 1$ seems reasonable and this is what we use in the simulation examples.

Remark 5. Clearly, other variates than (24) to (27) that fulfil (23) exist. For example, one may set $w_2(\mathbf{x},\mathbf{y},0) = w_2(\mathbf{x},\mathbf{y},1)$ and $w_1(\mathbf{x},\mathbf{y},0) = 0$ and proceed as before. We have tried also this, and variants of this, without getting better results.

Remark 6. Suppose we achieve a relative variance reduction a by using a control variate compared with just the sample mean. Alternatively we can obtain the same variance reduction by running the chain for more iterations and estimate μ by the sample mean in the longer

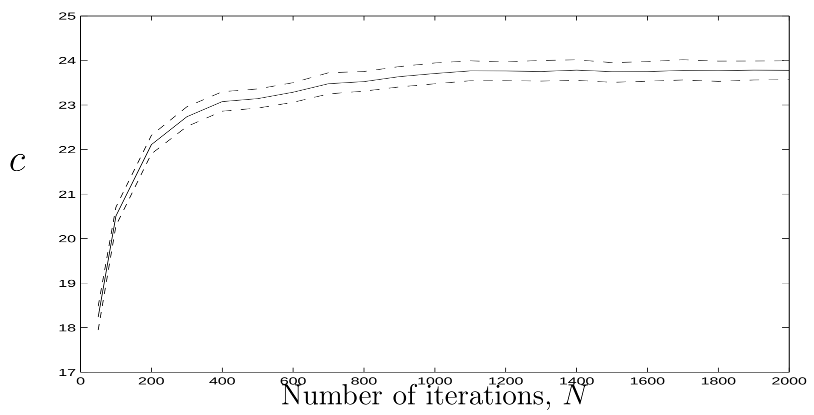


Figure 1: Values of c as a function of the number of realizations N. The dotted lines represent 95% percentile bootstrap intervals.

chain. Suppose the original chain was run for N_1 iterations and the longer chain for N_2 iterations (both after convergence). Then the longer chain gives a relative variance reduction of $a = (1/N_1 - 1/N_2)/(1/N_1) = (N_2 - N_1)/N_1$. Thus, to get a relative variance reduction of a we need to increase the number of iterations by a factor $r_a = N_2/N_1 = 1/(1-a)$.

4.3 Calculation of the control variate estimate

When calculating $\tilde{\mu}$ in (7) for one of the control variates in Section 4 we want to use the optimal value for c given in (8). However, this value cannot be directly used as both $\text{Cov}(\mu^*, v)$ and $\text{Var}(v)$ are unknown. In the following discussion we consider the control variate in (10), but a similar discussion is also valid for the variates defined by (21). Substituting (2) and (10) into (8), we get

$$c_{\text{opt}}(N) = -\frac{\frac{1}{N^2}\sum_{i=1}^N \sum_{j=1}^N \text{Cov}(f(\mathbf{x}^i), g(\mathbf{x}^j, \mathbf{y}^j))}{\frac{1}{N^2}\sum_{i=1}^N \sum_{j=1}^N \text{Cov}(g(\mathbf{x}^i, \mathbf{y}^i), g(\mathbf{x}^j, \mathbf{y}^j))}, \tag{28}$$

where we note that c_{opt} depends both on the number of iterations run, N, and the initial distribution. To remove the effect of the initial distribution we consider the situation after the burn-in period. Thus, we have a stationary Markov chain and get

$$c_{\text{opt}}(N) = -\frac{\sum_{h=-(N-1)}^{N-1} \left(1 - \frac{|h|}{N}\right) \gamma_{fg}(h)}{\sum_{h=-(N-1)}^{N-1} \left(1 - \frac{|h|}{N}\right) \gamma_{gg}(h)}, \tag{29}$$

where $\gamma_{fg}(h) = \text{Cov}(f(\mathbf{x}^i), g(\mathbf{x}^{i+h}, \mathbf{y}^{i+h}))$ and $\gamma_{gg}(h) = \text{Cov}(g(\mathbf{x}^i, \mathbf{y}^i), g(\mathbf{x}^{i+h}, \mathbf{y}^{i+h}))$. The asymptotic value when $N \to \infty$ becomes

$$c_{\text{opt}}(\infty) = \lim_{N\to\infty} c_{\text{opt}}(N) = -\frac{\sum_{h=-\infty}^{\infty} \gamma_{fg}(h)}{\sum_{h=-\infty}^{\infty} \gamma_{gg}(h)}. \tag{30}$$

Figure 1 We focus on estimating the asymptotic value $c_{\text{opt}}(\infty)$. This is clearly a good approximation to $c_{\text{opt}}(N)$ when N is large enough so that $|h|/N$ is small compared to 1 for all values of h where $\gamma_{fg}(h)$ or $\gamma_{gg}(h)$ are significantly different from zero. Thus, $c_{\text{opt}}(N) \approx c_{\text{opt}}(\infty)$ when

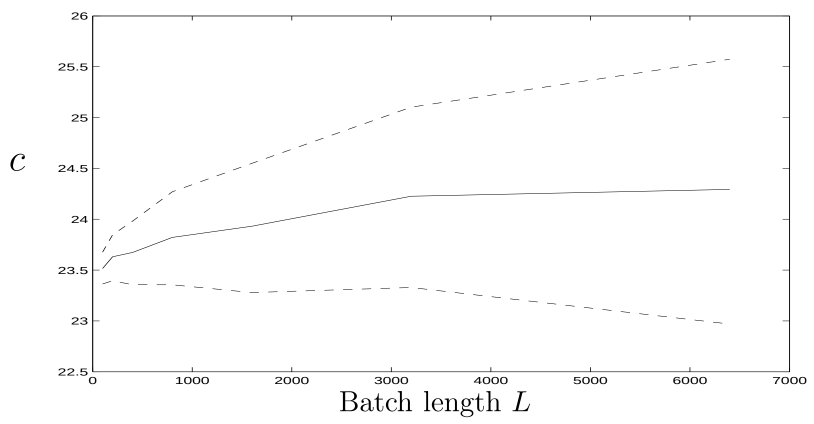


Figure 2: Values of the estimator $\widehat{c}_{\text{opt}}(\infty)$ based on a chain with N realizations, as a function of batch length L. The dotted lines represent 95% bootstrap percentile intervals.

N is larger than some multiple of the effective correlation length, where the effective correlation length may be defined as the smallest value r so that $\gamma_{fg}(h) \leq 0.05$ and $\gamma_{gg}(h) \leq 0.05$ for all $|h| > r$. In all our simulation exercises $\gamma_{gg}(h)$ drops to zero much faster than $\gamma_{fg}(h)$, so in practise it is the effective correlation length of $\gamma_{fg}(h)$ that matter. For the toy Gaussian example used to produce Figure 1 the effective correlation length of $\gamma_{fg}(\cdot)$ is about 60 iterations (found by visual inspection of the empirical correlation function). In Figure 1 we see that $c_{\text{opt}}(\infty)$ is obtained at $N \approx 1000$, so an N equal to about 17 times the effective correlation length is sufficient to obtain $c_{\text{opt}}(N) \approx c_{\text{opt}}(\infty)$. We have done similar studies for the three simulation examples in Section 5 with about the same general conclusion, an N equal to 15 to 20 effective correlation lengths is sufficient.

To estimate asymptotic values for $\text{Cov}(\mu^\star, v)$ and $\text{Var}(v)$ and thereby $c_{\text{opt}}(\infty)$ one may estimate the numerator and denominator in (30) separately via time series analysis methods (Priestly, 1981; Geier, 1992; Han and Green, 1992). We adopt the computationally simpler method of dividing the simulated chain into batches. More precisely, letting N be the number of simulated iterations after having discarded the burn-in period, divide the N iterations into M batches of $L = N/M$ iterations each. Assuming L to be at least 15 to 20 times the effective correlation length we get $c_{\text{opt}}(L) \approx c_{\text{opt}}(\infty) \approx c_{\text{opt}}(N)$. Moreover, we get that the correlation between the batch means are small, so ignoring these correlations we get the estimator

$$\widehat{c}_{\text{opt}}(\infty) = -\frac{\widehat{\text{Cov}(\widehat{\mu}^j, v^j)}}{\widehat{\text{Var}(v^j)}} = -\frac{\frac{1}{M}\sum_{j=1}^{M}(\widehat{\mu}^j - \widehat{\mu})(v^j - v)}{\frac{1}{M}\sum_{j=1}^{M}(v^j - v)^2}, \tag{31}$$

where $\widehat{\mu}^j$ and v^j are the sample mean and the control variate for batch number j, $\widehat{\mu} = (1/M)\sum_{j=1}^{M}\widehat{\mu}^j$ and $v = (1/M)\sum_{j=1}^{M} v^j$.

From the above we have that the batch length, L, should be at least 15 to 20 effective correlation lengths. For a given simulation length N we can then either choose to make the number of batches as large as possible under this restriction or to make the length of the batches, L, as large as possible under the restriction $M \geq 20$ (say). For the toy Gaussian example mentioned above, Figure 2 shows (estimated) values of $\widehat{c}_{\text{opt}}(\infty)$, based on a a chain with N iterations, as a function of the batch length L. The dotted lines are 95% bootstrap percentile intervals. We see that the uncertainty in the estimate increases as a function of

L. Similar studies for the simulation examples in Section 5 gave qualitatively similar results. Thus, the best is to maximise the number of batches, M, under the given restriction on L.

5 Numerical examples

In this section we evaluate the control variates defined above in three simulation examples. All three examples are based on previously analysed and published data sets and we adopt the exact same models and simulation algorithms previously used.

5.1 Gaussian Markov random field example

Here we consider a simple but much analysed binomial time series from Kitagawa (1987). Each day in 1983 and 1984 it was recorded whether there was more than one mm rainfall in Tokyo. We use the likelihood function (Kitagawa, 1987)

$$f(\mathbf{d}|\chi) = \prod_{t=0}^{n-1} \binom{n_t}{d_t} p(\chi_t)^{d_t} (1 - p(\chi_t))^{(n_t - d_t)} \tag{32}$$

where $p(\cdot)$ is the logit link and $\mathbf{d} = (d_1, \ldots, d_n)^T$ is number of days it rained more than one mm on each calendar day. Clearly, $n_t = 2$ for $t \neq 60$ and $n_{60} = 1$ corresponding to February 29. We use two prior models for $\chi = (\chi_1, \ldots, \chi_n)$, circular Gaussian Markov random fields with first and second order neighbourhood with precision κ which is apriori gamma distributed with parameters a and b. Rue and Held (2004) refer to the priors as RW1 and RW2, respectively. With $\mathbf{x} = (\chi, \kappa)$ the interest is in $\pi(\mathbf{x}|\mathbf{d})$ and in particular the posterior expectation for rain $p_t = E(p(\chi_t)|\mathbf{d})$, $t = 1, \ldots, 366$. Thus, with notation from Section 2 we have $f(\mathbf{x}) = p(\chi_t)$.

To sample from the posterior we use a Metropolis–Hastings algorithm from Rue and Held (2004). We first propose a new precision value $\kappa' = \phi\kappa$ where ϕ has density $\pi(\phi) \propto 1 + 1/\phi$ on the interval $[1/\Phi, \Phi]$. This conveniently makes $\pi(\kappa'|\kappa)/\pi(\kappa|\kappa') = 1$. Next we propose a new value χ' conditioned on κ' for the Markov random field from a second order Taylor approximation to the posterior distribution and finally jointly accept or reject (χ', κ'). We simulate using the library `GMRFlib` (Rue and Follestad, 2002). To estimate $p_t, t = 1, \ldots, 366$ we try all five control variates in Section 4. We denote the variates derived from (17) and (24) to (27) with v_0, v_1, v_2, v_3 and v_4, respectively and compute results for the corresponding five estimators,

$$\tilde{\mu}_{(l)} = \hat{\mu} + c \cdot v_l \quad \text{for } l = 0, \ldots, 4. \tag{33}$$

In the simulations we adopt hyper-parameter values $a = 1.0$ and $b = 0.000289$ from Rue and Held (2004). Further, we use $\Phi = 7$ which give acceptance rate around 30%. For each of the RW1 and RW2 priors we run the algorithm for 300000 iterations (after convergence). Our focus is on the variance reduction obtained by using $\tilde{\mu}_{(l)}$ for $l = 0, \ldots, 4$ in stead of $\hat{\mu}$. As discussed in Section 4.3, we estimate the variance reductions by dividing each run into M batches of length 20 effective correlation lengths. To get uncertainty estimates for the variance reductions we bootstrap the batches.

Figure 3 shows bootstrap percentile intervals for the variance reductions in the estimation of $p_t = E(p(\chi_t)|\mathbf{d})$ for each calendar day $t = 1, \ldots, 366$. The top and bottom rows show the reductions when using $\tilde{\mu}_{(0)}$ and $\tilde{\mu}_{(1)}$, respectively, in stead of $\hat{\mu}$, for each of the RW1 (left column) and RW2 (right column) priors. The $\tilde{\mu}_{(0)}$ results in variance reductions between 20 and

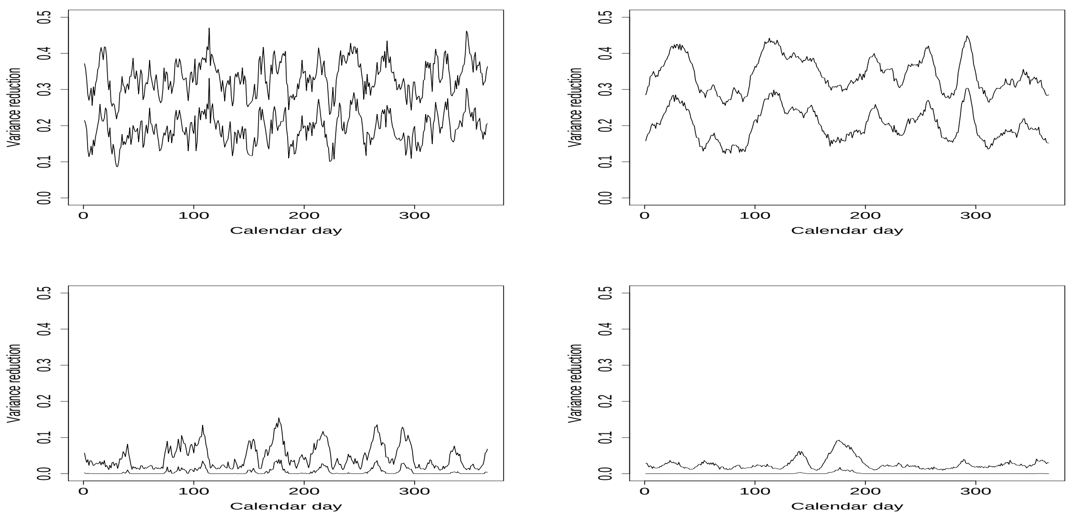


Figure 3: Gaussian Markov random field example: As function of calendar day t, 95% bootstrap percentile intervals for the relative variance reductions by estimating the posterior mean p_t by $\tilde{\mu}_{(0)}$ (upper row) and $\tilde{\mu}_{(1)}$ (lower row) in stead of the sample mean $\hat{\mu}$. Left and right columns contain results when using the RW1 and RW2 priors, respectively.

30%, while $\tilde{\mu}_{(1)}$ give minimal variance reductions. None of the other estimators give promising results. We also tried combinations of the control variates without getting significantly better results than by v_0 alone. Using v_0 alone typically resulted in r_a values between 1.2 and 1.55 which means that one has to increase the number of iterations by 20–55% to achieve the same variance reduction by the sample mean estimator.

5.2 Mode jumping example

In this section we reconsider the example in Section 4.2 of Tjelmeland and Hegstad (2001), where a mixture model is used for a data set originally presented in Brooks et al. (1997) concerning fetal deaths in litters of mice. The model is a mixture of beta-binomial and binomial distributions

$$p(\lambda|\eta) = \gamma \left[\binom{\eta}{\lambda} \prod_{r=0}^{\lambda-1} \frac{\mu + r\theta}{1 + r\theta} \prod_{r=0}^{\eta-\lambda-1} \frac{1 - \mu - r\theta}{1 + r\theta} \right] + (1 - \gamma) \left[\binom{\eta}{\lambda} \nu^{\lambda} (1 - \nu)^{\eta - \lambda} \right], \tag{34}$$

where λ is the number of deaths and η the number of implants or fetuses. The model parameters are $\gamma \in [0, 1]$, $\mu \in [0, 1]$, $\theta \geq 0$ and $\nu \in [0, 1]$, to which independent vague prior distributions are assigned. The resulting posterior turns out to have two distinct modes. Most of the posterior mass is in a mode with small values for γ and large values for ν, whereas a smaller mode has larger values for γ and small values for ν. Our focus here is to estimate the posterior mean for the probability mass contained in the smaller mode.

Given a current state $\mathbf{x} = (\gamma, \mu, \theta, \nu)$, we simulate using the mode jumping algorithm presented in Section 2.2. A potential new state $\mathbf{y}$ is generated by first drawing a vector φ from a wide Gaussian distribution and setting $k = 0$ or 1 at random. Second, a local optimisation algorithm for the posterior density is run, starting at $\mathbf{x} + \varphi$ if $k = 0$ and at $\mathbf{x} - \varphi$

Table 1: Mode jumping example: For the empirical mean, $\widehat{\mu}$, and each of $\tilde{\mu}_{(l)}, l = 0, \ldots, 4$, 95% confidence intervals for the posterior mean of the amount of probability mass contained in the smaller posterior mode, corresponding standard deviations, estimated r_a-values and 95% bootstrap percentile confidence intervals for the relative variance reductions by using $\hat{\mu}_{(l)}, l = 0, \ldots, 4$ in stead of $\hat{\mu}$.

Estimator	Confidence interval	Standard.dev.	r_a	Var. red.
$\hat{\mu}$	(0.01477, 0.01891)	$1.06 \cdot 10^{-03}$		
$\tilde{\mu}_{(0)}$	(0.01489, 0.01795)	$7.81 \cdot 10^{-04}$	1.83	(0.407, 0.496)
$\tilde{\mu}_{(1)}$	(0.01467, 0.01877)	$1.05 \cdot 10^{-03}$	1.02	(0.001, 0.053)
$\tilde{\mu}_{(2)}$	(0.01467, 0.01874)	$1.04 \cdot 10^{-03}$	1.03	(0.007, 0.072)
$\tilde{\mu}_{(3)}$	(0.01504, 0.01834)	$8.43 \cdot 10^{-04}$	1.57	(0.316, 0.411)
$\tilde{\mu}_{(4)}$	(0.01470, 0.01809)	$8.39 \cdot 10^{-04}$	1.58	(0.325, 0.416)

if $k = 1$. Third, a Gaussian or t-distribution is fitted to the local optimum found and, finally, the potential new state y is generated from the fitted distribution. More details can be found in the first reference given above. We ran the algorithm for 100000 iterations.

We consider the same control variates and estimators and adopt the same computational procedures to obtain confidence intervals as in Section 5.1. Table 1 summarises the results. For $\hat{\mu}$ and each of $\tilde{\mu}_{(l)}, l = 0, \ldots, 4$ we give 95% confidence intervals for the posterior mean of the amount of probability mass in the smaller mode, corresponding standard deviations, r_a-values and 95% bootstrap percentile intervals for the relative variance reductions by using $\hat{\mu}_{(l)}, l = 0, \ldots, 4$ in stead of $\hat{\mu}$. In contrast to the previous example we obtain satisfying results for three estimators, namely $\tilde{\mu}_{(0)}$, $\tilde{\mu}_{(3)}$ and $\tilde{\mu}_{(4)}$. However, $\tilde{\mu}_{(0)}$ is still best, and v_0, v_3 and v_4 are highly correlated so little extra is gained by using more than one variate.

5.3 Reversible jump example

Here we consider the model and data sets presented in Richardson and Green (1997). For three different data sets Richardson and Green (1997) use a mixture of Gaussian densities with a stochastic number of mixture components. Thus, the resulting posterior is defined on a space of varying dimension and a reversible jump algorithm is adopted for sampling. We use the same data sets as in Richardson and Green (1997) and also adopt the same model and simulation algorithm. Three groups of proposals are used. First, Gibbs updates is used for allocation of observations into mixture components and for model parameters. Second, reversible jump moves are used to split one mixture component into two or merging two into one. The last is to propose to remove an existing or to add a new empty mixture component. For details we refer to the reference given above.

Our focus is on number of mixture components, denoted m, and in particular its posterior distribution. Thus, for each value of m we let $f(\mathbf{x})$ be the indicator function that is one when the number of mixture components equals the given value. For estimation we consider only updates of the last two types. The first does not change m and is therefore of no relevance here. We consider the same five control variates as in Sections 5.1 and 5.2. However, we now define two control variates of each type, one for each of the two update types involving a change in m. For $l = 0, \ldots, 4$ let $v_{l,2}$ be the control variate defined from equations (17) and

(24) to (27), respectively, for the second update type, and similarly $v_{l,3}$ for the third update type. We considered the five estimators

$$\tilde{\mu}_{(l)} = \widehat{\mu} + c_2 v_{l,2} + c_3 v_{l,3} \quad \text{for } l = 0, 1, \ldots, 4 \tag{35}$$

and ran the algorithm for 32 million sweeps (as defined in Richardson and Green (1997)) after convergence. For each value of m we estimate the same type of quantities as in the previous examples, adopting the same computational procedures. The results are summarised in Figure 4. The left and right columns show results the "enzyme" and "acidity" data sets, respectively. The results for the third data set were almost equally promising. For $m = 2, \ldots, 10$, bootstrap percentile intervals is shown for the relative variance reduction by using $\tilde{\mu}_{(l)}$ in stead of $\hat{\mu}$. Row l give results for $\tilde{\mu}_{(l-1)}$. The first variate, v_0, is again most effective, but as in Section 5.2 also $\tilde{\mu}_{(3)}$ and $\tilde{\mu}_{(4)}$ give noticeable variance reductions.

6 Closing remarks

In this paper we introduce five new control variates for the Metropolis–Hastings algorithm. We consider functions of both the current and the proposed new state, and this enables us to construct control variates with zero mean values for general target and proposal distributions. We work out the ideas for the standard Metropolis–Hastings setting and for the more general reversible jump situation.

To apply the new variates require little extra effort, both in terms of implementation and computation. It is best implemented as a program post-processing the simulation output. The only change necessary in the simulation code is to store the proposed new value $\mathbf{y}$ and the acceptance ratio $R(\mathbf{x}, \mathbf{y})$ in addition to the current state $\mathbf{x}$. The post-processing is best implemented as a general program, the only part that need recoding for each problem is the function of interest, $f(\mathbf{x})$. The extra computation time is proportional to the number of iterations and is usually negligible. In contrast, the extra time used by the Rao-Blackwellization scheme introduced in Casella and Robert (1996) is quadratic in the number of iterations,

In three simulation examples we have explored what variance reductions are obtained with the new control variates. Two of the variates, v_1 and v_2 seem to be of little practical importance. The best results are consistently obtained for the simplest control variate, v_0, but also v_3 and v_4 give promising results in two of the examples. However, as v_3 and v_4 typically seem to be highly correlated with v_0, we expect it to be most reasonable to use only v_0.

The relative variance reductions obtained vary significantly depending on the target and proposal distributions used and the expectation of interest. The largest reduction in our examples is 45%, which corresponds to $r_a = 1.83$. Thus, to obtain a similar variance reduction by increasing the number of iterations, the run length must be increased by 83%. This number is from our mode jumping example where the simulation run took days of computation time, so here such a variance reduction is definitely of practical use.

The largest reductions seem to be obtained with proposal distributions that proposes large changes. This is certainly true for the mode jumping example. The reductions are smaller for the reversible jump example where, loosely speaking, the proposed changes are rather small. This conclusion is intuitively reasonable, when proposing very small changes the correlation between $\mathbf{x}$ and $\mathbf{y}$ becomes high and not much can be gained by using also the rejected proposals.

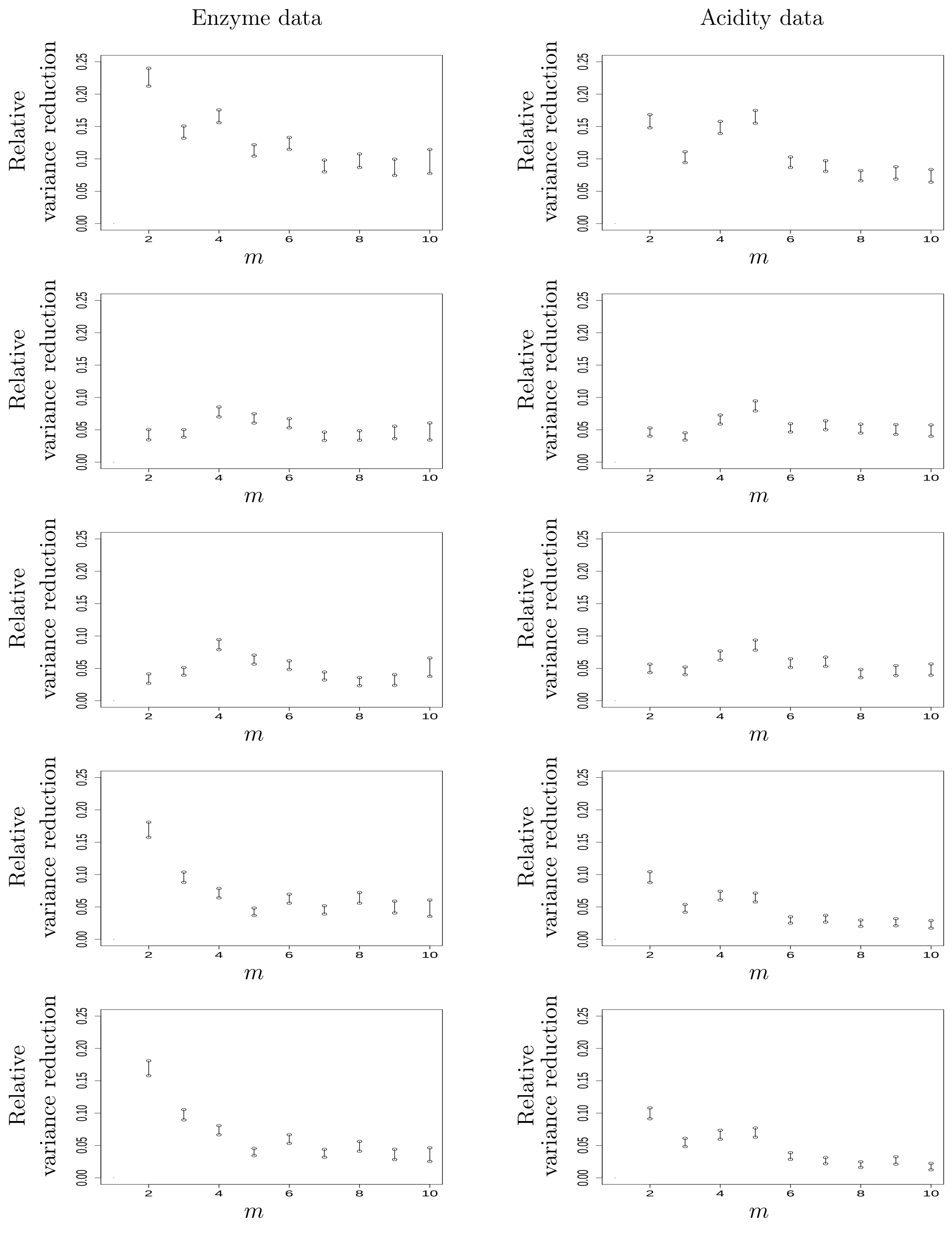


Figure 4: Reversible jump example: 95% Bootstrap percentile intervals for relative variance reduction in estimation of $\mathrm{P}(m = i),\ i = 2, \ldots, 10$ by using $\tilde{\mu}_{(l)}, l = 0, \ldots, 4$ in stead of the sample mean $\hat{\mu}$. The left and right columns contain results for the Enzyme and the Acidity data, respectively. Row l shows results for $\tilde{\mu}_{(l-1)}$.

Acknowledgements

We thank Håvard Rue for letting us use his code for the Gaussian Markov random field example.

References

Atchadé, Y. F. and Perron, F. (2005). Improving on the independent Metropolis-Hastings algorithm, *Statistica Sinica* **15**: 3–18.

Barker, A. A. (1965). Monte Carlo calculations of the radial distribution functions for a proton-electron plasma, *Australian Journal of Physics* **18**: 119–133.

Brooks, S. P., Morgan, B. J., Rideout, M. S. and Pack, S. E. (1997). Finite mixture models for proportions, *Biometrics* **53**: 1097–1115.

Casella, G. and Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes, *Biometrica* **83**: 81–94.

Dellaportas, P. and Roberts, G. O. (2003). *An introduction to MCMC*, J. Møller (ed.), Spatial Statistics and Computational Methods, number 173, Lecture Notes in Statistics, Springer, Berlin, pp. 1–41.

Geier, C. (1992). Practical Markov chain Monte Carlo (with discussion), *Statistical Science* **7**: 473–511.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrica* **82**: 711–732.

Hammersley, J. M. and Morton, K. W. (1956). A new Monte Carlo technique: Antithetic variates, *Proceedings of the Cambridge Philosophical Society* **52**: 449–475.

Han, X. L. and Green, P. J. (1992). Metropolis methods, Gaussian proposals, and antithetic variables *in* P. Barone, A. Frigessi and M. Piccioni (eds), *Stochastic Models, Statistical Methods and Algorithms in Image Analysis, number 74 in Lecture Notes in Statistics, Springer, Berlin,* pp. 142–164.

Hastings, W. (1970). Monte Carlo sampling using Markov chains and their applications, *Biometrica* **57**: 97–109.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time series (with discussion), *Journal of the American Statistical Association* **82**: 1032–1063.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*, Springer, Berlin.

Mira, A., Tenconi, P. and Bressanini, D. (2003). Variance reduction in MCMC, *Technical Report 29/2003*, Department of Economics, University of Insubria, Italy.

Peskun, P. (1973). Optimal Monte-Carlo sampling using Markov chains, *Biometrica* **60**: 607–612.

Pinto, R. L. and Neal, R. M. (2001). Improving Markov chain Monte Carlo estimators by coupling to an approximating chain, *Technical Report No. 0101*, Department of Statistics, University of Toronto.

Priestly, M. B. (1981). *Spectral analysis and time series*, Academic, London.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components, *Journal of the Royal Statistical Society, Series B* **59**: 731–792.

Rue, H. and Follestad, T. (2002). *GMRFlib: A C-library for fast and exact simulation of Gaussian Markov random fields*, Statistics Report No. 1, Department of Mathematical Sciences, Norwegian University of Science and Technology, Norway.

Rue, H. and Held, L. (2004). *Gaussian Markov Random Fields, Theory and Applications*, Chapman and Hall/CRC.

Smith, A. and Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion), *Journal of the Royal Statistical Society, Series B* **55**: 2–24.

Tjelmeland, H. and Hegstad, B. K. (2001). Mode jumping proposals in MCMC, *Scandinavian Journal of Statistics* **28**: 205–223.

Waagepetersen, R. and Sørensen, D. (2001). A tutorial on reversible jump MCMC with a view toward application in QTL-mapping, *International Statistical Review* **69**: 49–61.

Hugo Hammer, Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim, Norway
E-mail: hammer@math.ntnu.no

A Proof of the first control variate for the Reversible jump algorithm

Recall the algorithm described in Section 2.3. Here comes some more details related to the algorithm which is needed to prove the first control variate. The functional relation between $(\mathbf{z},\mathbf{u})$ and $(\mathbf{z}',\mathbf{u}')$ can be written as

$$(\mathbf{z}',\mathbf{u}') = \phi_{mm'}(\mathbf{z},\mathbf{u}) = (\phi_{1mm'}(\mathbf{z},\mathbf{u}), \phi_{2mm'}(\mathbf{z},\mathbf{u}))$$

and

$$(\mathbf{z},\mathbf{u}) = \phi_{mm'}^{-1}(\mathbf{z}',\mathbf{u}') = (\phi_{1m'm}(\mathbf{z}',\mathbf{u}'), \phi_{2m'm}(\mathbf{z}',\mathbf{u}')).$$

The deterministic mappings must be of dimension $\phi_{1mm'} : \mathbb{R}^{n_m+n_{mm'}} \to \mathbb{R}^{n_{m'}}$ and $\phi_{2mm'} : \mathbb{R}^{n_m+n_{mm'}} \to \mathbb{R}^{n_{m'm}}$. When considering a move from state $(m,\mathbf{z})$ to $(m',\mathbf{z}') = (m', \phi_{1mm'}(\mathbf{z},\mathbf{u}))$ and the reversed move $(m',\mathbf{z}')$ to $(m,\mathbf{z}) = (m, \phi_{1m'm}(\mathbf{z}',\mathbf{u}'))$ the crucial so called dimensional matching condition $n_m + n_{mm'} = n_{m'} + n_{m'm}$ must be satisfied to obtain reversibility. It means that the vectors $(\mathbf{z},\mathbf{u})$ and $(\mathbf{z}',\mathbf{u}')$ must be of the same dimension. We write the Jacobi determinant in (6) as

$$J = \left| \frac{\phi_{mm'}(\mathbf{z},\mathbf{u})}{\partial \mathbf{z} \partial \mathbf{u}} \right|.$$

Remember that we have $\mathbf{x} = (m,\mathbf{z})$, $\mathbf{y} = (m',\mathbf{z}')$ and $\mathbf{z}' = \phi_{1mm'}(\mathbf{z},\mathbf{u})$. Thus, $g_0(\mathbf{x},\mathbf{y})$ defined in (11) can be written on the form

$$g_0(\mathbf{x},\mathbf{y}) = w_1(m,\mathbf{z},m',\mathbf{u})f(m,\mathbf{z}) + w_2(m,\mathbf{z},m',\mathbf{u})f(m',\phi_{1mm'}(\mathbf{z},\mathbf{u})). \tag{36}$$

This gives

$$\begin{aligned}
E(g_0(\mathbf{x},\mathbf{y})) &= \sum_m \sum_{m'} \iint [w_1(m,\mathbf{z},m',\mathbf{u})f(m,\mathbf{z}) + w_2(m,\mathbf{z},m',\mathbf{u})f(m',\phi_{1mm'}(\mathbf{z},\mathbf{u}))] \\
&\qquad \pi(m,\mathbf{z})p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})\mathrm{d}\mathbf{z}\mathrm{d}\mathbf{u} \\
&= \sum_m \sum_{m'} \iint w_1(m,\mathbf{z},m',\mathbf{u})f(m,\mathbf{z})\pi(m,\mathbf{z})p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})\mathrm{d}\mathbf{z}\mathrm{d}\mathbf{u} \\
&+ \sum_m \sum_{m'} \iint w_2(m,\mathbf{z},m',\mathbf{u})f(m',\phi_{1mm'}(\mathbf{z},\mathbf{u}))\pi(m,\mathbf{z})p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})\mathrm{d}\mathbf{z}\mathrm{d}\mathbf{u}
\end{aligned}$$

Performing the substitution $(\mathbf{z}',\mathbf{u}') = \phi_{mm'}(\mathbf{z},\mathbf{u})$ in the last term above, and thereafter interchanging the variable symbols $(m,\mathbf{z},\mathbf{u})$ and $(m',\mathbf{z}',\mathbf{u}')$ in the same term we get

$$\begin{aligned}
E(g_0(\mathbf{x},\mathbf{y})) &= \sum_m \sum_{m'} \iint w_1(m,\mathbf{z},m',\mathbf{u})f(m,\mathbf{z})\pi(m,\mathbf{z})p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})\mathrm{d}\mathbf{z}\mathrm{d}\mathbf{u} \\
&+ \sum_m \sum_{m'} \iint w_2(m',\phi_{1mm'}(\mathbf{z},\mathbf{u}),m,\phi_{2mm'}(\mathbf{z},\mathbf{u}))f(m,\mathbf{z})\pi(m',\phi_{1mm'}(\mathbf{z},\mathbf{u})) \\
&\qquad p_{m'm}q_{m'm}(\phi_{mm'}(\mathbf{z},\mathbf{u})) \left| \frac{\phi_{mm'}(\mathbf{z},\mathbf{u})}{\partial \mathbf{z} \partial \mathbf{u}} \right| \mathrm{d}\mathbf{z}\mathrm{d}\mathbf{u}.
\end{aligned} \tag{37}$$

Thus, we get a sufficient condition for $E(g_0(\mathbf{x},\mathbf{y}))=0$ by setting the sum of the two integrands identical to zero. This gives

$$\begin{aligned}
&w_1(m,\mathbf{z},m',\mathbf{u}) \\
&= \frac{\pi(m',\phi_{1mm'}(\mathbf{z},\mathbf{u}))p_{m'm}q_{m'm}(\phi_{mm'}(\mathbf{z},\mathbf{u}))\left|\frac{\phi_{mm'}(\mathbf{z},\mathbf{u})}{\partial\mathbf{z}\partial\mathbf{u}}\right|}{\pi(m',\phi_{1mm'}(\mathbf{z},\mathbf{u}))p_{m'm}q_{m'm}(\phi_{mm'}(\mathbf{z},\mathbf{u}))\left|\frac{\phi_{mm'}(\mathbf{z},\mathbf{u})}{\partial\mathbf{z}\partial\mathbf{u}}\right|+\pi(m,\mathbf{z})p_{mm'}q_{mm'}(\mathbf{u}|\mathbf{z})} \qquad (38)\\
&w_2(m,\mathbf{z},m',\mathbf{u}) = -w_1(m,\mathbf{z},m',\mathbf{u}) \qquad (39)
\end{aligned}$$

and we get the first control variate for the reversible jump setting by substituting (38) and (39) into (36). Finally, we get (17) using the expression for the acceptance ratio $R(\mathbf{x},\mathbf{y})$.

Paper II

Directional Metropolis–Hastings algorithms on hyperplanes

Hugo Hammer and Håkon Tjelmeland
Department of Mathematical Sciences
Norwegian University of Science and Technology
Trondheim, Norway

Abstract

In this paper we define and study new directional Metropolis–Hastings algorithms that propose states in hyperplanes. Each iteration in directional Metropolis–Hastings algorithms consist of three steps. First a direction is sampled by an auxiliary variable. Then a potential new state is proposed in the subspace defined by this direction and the current state. Lastly, the potential new state is accepted or rejected according to the Metropolis–Hastings acceptance probability. Traditional directional Metropolis–Hastings algorithms define the direction by one vector and so the corresponding subspace in which the potential new state is sampled is a line. In this paper we let the direction be defined by two or more vectors and so the corresponding subspace becomes a hyperplane.

We compare the performance of directional Metropolis–Hastings algorithms defined on hyperplanes with other frequently used Metropolis–Hastings schemes. The experience is that hyperplane algorithms on average produce larger jumps in the sample space and thereby have better mixing properties per iteration. However, with our implementations the hyperplane algorithms is more computation intensive per iteration and so the simpler algorithms in most cases are better when run for the same amount of computer time. An interesting area for future research is therefore to find variants of our directional Metropolis–Hastings algorithms that require less computation time per iteration.

Keywords: Angular Gaussian distribution, Directional Metropolis–Hastings, Markov chain Monte Carlo, Hyperplane algorithms, Subspace.

1 Introduction

We are often interested in calculating the mean of some function $f(x)$ for x distributed according to a target probability distribution $\pi(\cdot)$. If x is of high dimension and $\pi(\cdot)$ is complex, stochastic simulation is the only viable alternative. There exists a wide range of simulation algorithms for this. If the target distribution $\pi(\cdot)$ is sufficiently complex, the Metropolis–Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) is the standard choice. In MH algorithms we run a Markov chain which has $\pi(\cdot)$ as its limiting probability distribution. Each new state of the chain is generated in two steps. Letting x denote the current state of the Markov chain, first a potential new state, y, is proposed and second the proposal y is accepted with a certain probability, otherwise retaining x as the current state.

In directional MH algorithms the proposal step is done in two parts. First, a direction is sampled via an auxiliary variable. Next, the potential new state is sampled in the subspace defined by the sampled direction and the current state of the Markov chain. Several directional MH algorithms are discussed in the literature: Chen and Schmeiser (1993) generate the direction from an uniform distribution, Roberts and Rosenthal (1998) present an algorithm that moves along a direction centered at the gradient direction evaluated in the current point. Also Eidsvik and Tjelmeland (2006) let the direction depend on the current state of the Markov chain. Gilks et al. (1994), Roberts and Gilks (1994) and Liu and Sabatti (2000) use information from parallel chains to generate proposals along favourable directions.

Goodman and Sokal (1989) and Liu and Sabatti (2000) find what proposal distribution to use in the chosen subspace to obtain unit MH acceptance rate. Eidsvik and Tjelmeland (2006) also study this, but for direction distributions that depend on the current state of the Markov chain.

In all the articles discussed above, the directional MH algorithms use directions that are defined by one vector, and so the corresponding subspace is a line. The focus of this paper is when the direction is defined by two or more vectors, so that the resulting subspace is a (hyper) plane. By allowing ourselves to generate the potential new states in a hyperplane instead of just a line, the setup becomes more flexible and thereby there should be a potential for obtaining algorithms with better mixing properties. We implement this idea by generalising the setup in Eidsvik and Tjelmeland (2006). Similar to what is done in that article our focus is on proposal distributions that give unit MH acceptance rates. We compare the performance of our new directional MH schemes with other frequently used MH algorithms. Our experience is that the hyperplane algorithms on average produce larger jumps in each iteration and thereby have better mixing properties per iteration. In some cases our new algorithms even give negative one iteration correlations, which we were not able to obtain by either the directional MH algorithms using only one vector to define the direction, or the simple random walk and Langevin proposal algorithms. However, with our implementations the hyperplane algorithms require more computation time per iteration and in most cases therefore is outperformed by the simpler algorithms when run for the same amount of computer time. An interesting area for future research is therefore to define variants of the hyperplane algorithms that require less computation time per iteration.

The paper is organised as follows. In Section 2 we give a short introduction to directional MH algorithms and in Section 3 we present details for directional MH algorithms generating proposals in hyperplanes. In Section 4 we study how to define the proposal distributions to obtain unit acceptance rate. In section 5 we present a simulation example comparing the performance of our new directional MH algorithm with other frequently used MH schemes.

2 Metropolis–Hastings algorithms

A MH algorithm simulate from a specified target distribution, $\pi(\cdot)$, by simulating a Markov chain for a large number of iterations, where the Markov chain is constructed to have the target distribution $\pi(\cdot)$ as its limiting distribution. In this paper we restrict the attention to continuous target distribution on $\mathbb{R}^n$ and let $\pi(\cdot)$ denote its density with respect to the Lebesgue measure on $\mathbb{R}^n$. We let $x \in \mathbb{R}^n$ denote the current state of the Markov chain.

2.1 Parametric Metropolis–Hastings

Parametric MH is a subclass of the MH algorithm where a parametric form is used to generate potential new states (Green, 1995; Waagepetersen and Sørensen, 2001), also known as reversible jump algorithms. The algorithms are originally developed for target distributions defined on sample spaces of varying dimension, but they are equally valid when the dimension of the state vector x is fixed. In the following we describe the procedure for this latter case.

To generate a potential new state y one first samples a $t \in \mathbb{R}^m$ from a distribution $q(t|x)$

and secondly generates the proposal y using the one-to-one transformation

$$\left.\begin{array}{l} y = w_1(x,t) \\ s = w_2(x,t) \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} x = w_1(y,s) \\ t = w_2(y,s), \end{array}\right. \tag{1}$$

where $w_1 : \mathbb{R}^{n+m} \to \mathbb{R}^n$ and $w_2 : \mathbb{R}^{n+m} \to \mathbb{R}^m$. The proposal y should thereafter be accepted with probability

$$\alpha(y|x) = \min\left\{1, \frac{\pi(y)q(s|y)}{\pi(x)q(t|x)}|J|\right\}, \tag{2}$$

where

$$J = \begin{vmatrix} \frac{\partial w_1(x,t)}{\partial x} & \frac{\partial w_1(x,t)}{\partial t} \\ \frac{\partial w_2(x,t)}{\partial x} & \frac{\partial w_2(x,t)}{\partial t} \end{vmatrix} \tag{3}$$

is the Jacobian determinant for the one-to-one relation (1).

In the following we also use an auxiliary stochastic variable in the parametric MH setup. Letting $\varphi \in \Phi$ denote the auxiliary variable, the procedure is then as follows. First we generate φ from some distribution $h(\varphi|x)$. As indicated in the notation the distribution for φ may depend on the current state vector x. Next we generate t from a distribution $q(t|x,\varphi)$ that also may depend on the auxiliary variable. From t and φ we compute the proposal y using the one-to-one relation (1), where now w_1 and w_2 may be functions also of φ. Note, however, that the one-to-one relation is still between (x,t) and (y,s), the auxiliary variable φ should here be considered as a fixed parameter. Lastly, we accept the proposal y with probability

$$\alpha(y|\varphi,x) = \min\left\{1, \frac{\pi(y)h(\varphi|y)q(s|\varphi,y)}{\pi(x)h(\varphi|x)q(t|\varphi,y)}|J_\varphi|\right\}, \tag{4}$$

where, as indicated in the notation, the Jacobian determinant may be a function of φ.

2.2 Directional Metropolis–Hastings

The class of directional MH algorithms is an important subclass of parametric MH algorithms. In directional MH algorithms potential new states are traditionally generated along a line in the sample space defined by the current state x and an auxiliary variable φ, where φ is either a point or a directional vector in $\mathbb{R}^n$. This situation is considered in Eidsvik and Tjelmeland (2006) and in the following we introduce this setup and discuss the most important experiences for this situation. In the next section we generalise the scheme to allow proposals in a hyperplane.

Following Eidsvik and Tjelmeland (2006), we use z to denote the auxiliary variable whenever it represents a point in $\mathbb{R}^n$, and use u when it represents a direction vector. First consider the case that the line is defined by x and an auxiliary point z. An iteration of the MH algorithm then starts by generating z from a distribution $p(z|x)$. Next, we generate a scalar t from some distribution $q(t|z,x)$ and compute the proposal y through the one-to-one relation

$$\left.\begin{array}{l} y = x + t(z-x) \\ s = -\frac{t}{1-t} \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} x = y + s(z-y) \\ t = -\frac{s}{1-s}. \end{array}\right. \tag{5}$$

Using (4), the acceptance probability becomes

$$\alpha(y|z,x) = \min\left\{1, \frac{\pi(y)p(z|y)q(s|z,y)}{\pi(x)p(z|x)q(t|z,y)}|J_z|\right\}, \tag{6}$$

where the Jacobian determinant is

$$J_z = \begin{vmatrix} \frac{\partial y(x,t)}{\partial x} & \frac{\partial y(x,t)}{\partial t} \\ \frac{\partial s(x,t)}{\partial x} & \frac{\partial s(x,t)}{\partial t} \end{vmatrix} = \begin{vmatrix} (1-t)\cdot \mathbf{I}_n & z-x \\ \mathbf{0} & -(1-t)^{-2} \end{vmatrix} = -(1-t)^{n-2}, \tag{7}$$

where I_n is the n-dimensional identity matrix and $\mathbf{0}$ is a matrix with only zero entries.

Consider next the situation when we use a direction vector u to define the line. To obtain a unique vector u for a given line, Eidsvik and Tjelmeland (2006) require

$$u \in S^n = \{u \in \mathbb{R}^n \setminus \{0\} : \|u\| = 1 \text{ and } u_l > 0 \text{ for } l = \min\{j; u_j \neq 0\}, \}. \tag{8}$$

Each iteration of the algorithm then starts by generating a direction u from a distribution $g(u|x)$. Next, a scalar t is proposed from a distribution $q(t|u,x)$ and the proposal y is found by the one-to-one relation

$$\left.\begin{array}{l} y = x + tu \\ s = -t \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} x = y + su \\ t = -s. \end{array}\right. \tag{9}$$

The acceptance probability now becomes

$$\alpha(y|u,x) = \min\left\{1, \frac{\pi(y)g(u|y)q(s|u,y)}{\pi(x)g(u|x)q(t|u,y)}|J_u|\right\}, \tag{10}$$

where

$$J_u = \begin{vmatrix} \frac{\partial y(x,t)}{\partial x} & \frac{\partial y(x,t)}{\partial t} \\ \frac{\partial s(x,t)}{\partial x} & \frac{\partial s(x,t)}{\partial t} \end{vmatrix} = \begin{vmatrix} \mathbf{I}_n & u \\ \mathbf{0} & -1 \end{vmatrix} = -1. \tag{11}$$

A critical choice in the two MH algorithms defined above is the distribution for the auxiliary variable, $p(z|x)$ or $g(u|x)$. For $p(z|x)$ Eidsvik and Tjelmeland (2006) propose to use a Gaussian approximation to the target distribution. For $g(u|x)$ a uniform distribution in S^n is clearly a possible choice. However, a better choice would be one that is adapted to the target distribution in question and the current state x. This can be achieved by first sampling a point z from a $p(z|x)$ and let u be the direction vector that define the same line, i.e.

$$u = \begin{cases} \frac{z-x}{||z-x||} & \text{if } \frac{z-x}{||z-x||} \in S^n, \\ -\frac{z-x}{||z-x||} & \text{otherwise.} \end{cases} \tag{12}$$

The resulting $g(u|x)$ is then given as

$$g(u|x) = \int_{-\infty}^{\infty} |r|^{n-1} p(x + ru|x)\mathrm{d}r. \tag{13}$$

To be able to compute the corresponding acceptance probability $g(u|x)$ must be analytically available. In general this is not be the case, but when $p(z|x)$ is Gaussian the corresponding $g(u|x)$ is called the angular Gaussian distribution and can easily be evaluated, see e.g. Watson (1983) and Pukkila and Rao (1988).

One should note that the acceptance probabilities in the two above directional MH algorithms differ. They become different even if the proposal mechanisms are chosen identical by defining $g(u|x)$ indirectly via $p(z|x)$ as discussed above and the proposal distributions along the line are chosen to be the same. The experience reported in Eidsvik and Tjelmeland (2006) is that the algorithm based on u results in longer jumps and better mixing. Intuitively that can be understood as the acceptance probabilities are conditional on the sampled values of the auxiliary variable and a direction u contains less information than a point z. When generalising the above scheme to a situation where the auxiliary variable defines a hyperplane instead of a line our focus is therefore on direction vectors.

2.3 Hyperplane directional MH algorithms

We now consider a directional MH algorithm where the potential new state is proposed in a hyperplane of a dimension $k < n$. Thus, $k = 1$ corresponds to the algorithm discussed above. Each iteration of the algorithm then starts by generating k vectors $u_1, \ldots, u_k$ according to a distribution $g(u|x)$, where $u = (u_1, \ldots, u_k)$. Next we propose a k-dimensional vector $t = (t_1, \ldots, t_k)^T \in \mathbb{R}^k$ from a distribution $q(t|x, u)$ and compute the proposal y through the one-to-one relation

$$\left.\begin{array}{l} y = x + \sum_{i=1}^k t_i u_i \\ s = -t \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} x = y + \sum_{i=1}^k s_i u_i \\ t \ = -s, \end{array}\right. \tag{14}$$

where $s = (s_1, \ldots, s_k)^T \in \mathbb{R}^k$. The corresponding acceptance probability becomes

$$\alpha(y|u, x) = \min\left\{1, \frac{\pi(y)g(u|y)q(s|u, y)}{\pi(x)g(u|x)q(t|u, y)}|J_u|\right\}, \tag{15}$$

where the Jacobian determinant is given by

$$J_u = \begin{vmatrix} \frac{\partial y(x,t)}{\partial x} & \frac{\partial y(x,t)}{\partial t} \\ \frac{\partial s(x,t)}{\partial x} & \frac{\partial s(x,t)}{\partial t} \end{vmatrix} = \begin{vmatrix} \mathbf{I}_n & u_1 \ \cdots \ u_k \\ \mathbf{0} & -\mathbf{I}_k \end{vmatrix} = (-1)^k. \tag{16}$$

To fully define the simulation algorithm it remains to specify the distribution of the auxiliary variables, $g(u|x)$, and the proposal distribution $q(t|u, x)$. In the following we first discuss how $q(t|u, x)$ should be chosen to give unit acceptance probability.

2.3.1 Proposal q with unit acceptance rate

Following the strategy used in Eidsvik and Tjelmeland (2006) one can easily identify a proposal distribution $q(t|x, u)$ that gives unit acceptance probability, namely

$$q(t|u, x) = c(x, u)\ \pi\left(x + \sum_{i=1}^k t_i u_i\right)\ g\left(u \middle| x + \sum_{i=1}^k t_i u_i\right), \tag{17}$$

where $c(x, u)$ is a normalising constant given by

$$c(x, u) = \left[\int_{\mathbb{R}^k} \pi\left(x + \sum_{i=1}^k t_i u_i\right)\ g\left(u \middle| x + \sum_{i=1}^k t_i u_i\right)\,\mathrm{d}t\right]^{-1}. \tag{18}$$

To see that the resulting acceptance probability is equal to one, insert the above expression for $q(t|u, x)$ in (15),

$$\alpha(y|u, x) = \min\left\{\frac{\pi(y)\ g(u|y)\ c(y, u)\ \pi\left(y + \sum_{i+1}^k s_i u_i\right) g\left(u \middle| y + \sum_{i=1}^k s_i u_i\right)}{\pi(x)\ g(u|x)\ c(x, u)\ \pi\left(x + \sum_{i=1}^k t_i u_i\right) g\left(u | x + \sum_{i=1}^k t_i u_i\right)}\right\}. \tag{19}$$

Using the one-to-one relation between (x,t) and (y,s) in (14) and noting that $c(x,u)=c(y,u)$ we see that everything in the fraction cancels and we get $\alpha(y|u,x)=1$.

It should be noted that in most cases we will in practice not be able to use the $q(t|u,x)$ defined above because the integral in (17) is not analytically available. However, the above expressions are still of interest as we may construct analytically available approximations to (17) and thereby get acceptance rates close to one. In practice how to do this depend on the properties of the $q(t|u,x)$ in (17), which in turn depends on the choice of $g(u|x)$. Next we therefore discuss the choice of $g(u|x)$.

3 Choice of $g(u|x)$

The goal is to choose $g(u|x)$ so that the resulting hyperplanes with high probability goes through high density areas of the target distribution. To obtain this it seems reasonable to draw auxiliary points, $z_1,\ldots,z_k$, from a distribution that resembles the target density and use direction vectors that span the same hyperplane. In the following we discuss two ways to define direction vectors $u_1,\ldots,u_k$ from $z_1,\ldots,z_k$. The first set of direction vectors we denote by $u_1^{(1)},\ldots,u_k^{(1)}$ and the resulting distribution by $g_1(u^{(1)}|x)$ and use $u_1^{(2)},\ldots,u_k^{(2)}$ and $g_2(u^{(2)}|x)$ for the corresponding quantities for the second set of direction vectors.

Assume we have available a Gaussian approximation to the target distribution $\pi(\cdot)$, and denote this by $\tilde{\pi}(\cdot)$. Thus, letting $z_1,\ldots,z_k$ be independent realisations from $\tilde{\pi}(\cdot)$, we may define corresponding direction vectors $u_1^{(1)},\ldots,u_k^{(1)}$ by applying (12) to each z_i separately. Referring to our discussion in Section 2.2, the resulting $g_1(u^{(1)}|x)$ clearly becomes a product of angular Gaussian densities.

Notice that even though each $u_i^{(1)}$ is uniquely defined by the corresponding z_i, the set $u_1^{(1)},\ldots,u_k^{(1)}$ is not uniquely defined from the set $z_1,\ldots,z_k$. Other direction vectors spanning the same hyperplane may be defined by taking linear combinations of $u_1^{(1)},\ldots,u_k^{(1)}$. Thus, as we discuss in the last paragraph of Section 2.2, we should expect to get an algorithm with better mixing properties by in stead using a set of direction vectors, $u_1^{(2)},\ldots,u_k^{(2)}$, that is uniquely defined from $z_1,\ldots,z_k$. We obtain this by setting

$$\begin{aligned}
u_1^{(2)} &= [1,0,\ldots,0,u_{1,k+1},\ldots,u_{1,n}]^T \\
u_2^{(2)} &= [0,1,\ldots,0,u_{2,k+1},\ldots,u_{2,n}]^T \\
&\ \vdots \\
u_k^{(2)} &= [0,0,\ldots,1,u_{k,k+1},\ldots,u_{k,n}]^T.
\end{aligned} \tag{20}$$

Thus, $u_1^{(2)},\ldots,u_k^{(2)}$ are defined by $k(n-k)$ variables and these can by found from $z_1,\ldots,z_k$ by solving

$$z_i = x + \sum_{j=1}^{k} c_{ij} u_j^{(2)} \quad \text{for } i=1,\ldots,k, \tag{21}$$

with respect to $u^{(2)}=\{u_{ij}, i=1,\ldots,k, j=k+1,\ldots,n\}$ and $c=\{c_{ij}, i,j=1,\ldots,k\}$. The resulting distribution for $u^{(2)}$, $g_2(u^{(2)}|x)$, is found by first finding the joint distribution for $u^{(2)}$ and c, $g_2(u^{(2)},c|x)$, and thereafter marginalise this over c,

$$g_2\left(u^{(2)}|x\right) = \int g_2\left(u^{(2)},c\middle| x\right)dc. \tag{22}$$

For $n = 2$ and $k = 1$ this integral can be solved analytically and $u_{1,2}$, which is then the only variable in $u^{(2)}$, has a Cauchy distribution. For $n > 2$ we have not been able to solve the integral analytically, but simulations show that $g_2(u^{(2)}|x)$ has heavy tails also for $n > 2$. As $g_2(u^{(2)}|x)$ is not analytically available for $n > 2$ it can not be used, but an approximation to it may be used. To construct such an approximation we note that both the conditional marginal distribution for c given x, and the conditional distribution for $u^{(2)}$ given c and x are Gaussians. Thus, it is natural to factorize the integrand in (22) as a product of these two distributions,

$$g_2(u^{(2)}, c|x) = g_2(u^{(2)}|c, x)g_2(c|x). \tag{23}$$

Note, however, that $u^{(2)}$ and c given x are not jointly Gaussian. Inserting this in (22) and performing a linear substitution for c so that the new variable, denoted by $\tilde{c}$, gets a k^2-variate standard Gaussian distribution, we have

$$g_2(u^{(2)}|x) = \int g_2(u^{(2)}|\tilde{c}, x)\mathrm{N}(\tilde{c})\mathrm{d}\tilde{c}, \tag{24}$$

where $\mathrm{N}(\cdot)$ is the density of a standard Gaussian distribution. A natural approximation is therefore to sample $\tilde{c}_1, \ldots, \tilde{c}_N$ independently from the standard Gaussian distribution and use

$$\widehat{g}_2(u^{(2)}|x) = \frac{1}{N}\sum_{i=1}^{N} g_2(u^{(2)}|\tilde{c}_i, x). \tag{25}$$

Clearly, $\widehat{g}_2(u^{(2)}|x)$ is easy to sample from and it is straight forward to evaluate the corresponding density. We will therefore use this as our second distribution for u. We note in passing that taking $N = 1$ corresponds to using $z_1, \ldots, z_k$ to define the hyperplane. Clearly, in practice one should use a value for N much larger than this.

4 Appearance of the unit acceptance rate proposal distribution

For $k = 1$ Eidsvik and Tjelmeland (2006) show that the proposal distribution that gives unit acceptance rate, equation (17), typically becomes a bi-modal distribution with one mode close to x and one far away from x. Thus, using this proposal distribution enables large jumps in the sample space by proposing values to the mode away from the current state x. In this section we study how this generalises for $k > 1$. As discussed in Section 2.3.1 we can not actually use (17) as our proposal distribution, but this is still of interest for two reasons. It shows what the potential is for using a proposal distribution that is an approximation to (17) and it may generate ideas for how to construct good approximations to (17).

To study the appearance of (17) for $k > 1$ we consider a simplified variant of a Bayesian model that in Rabben et al. (2008) is used for non-linear inversion of seismic data. The variable of interest, $x \in \mathbb{R}^n$, is assumed to have a Gaussian prior distribution with mean m and covariance matrix S. Given x, the data d is assumed Gaussian with mean value $ax \odot x + bx$ and covariance matrix Σ, where a and b are scalar (known) parameters and $\odot$ denotes elementwise multiplication. Thus, the posterior distribution for x given d becomes non-Gaussian for $a \neq 0$. Our simplification relative to the model used in Rabben et al. (2008) is that we here consider $n = 10$, and in Section 5 $n = 30$, whereas in the seismic inversion application much larger values of n are of interest.

Our target distribution of interest is the posterior distribution $\pi(x|d) \propto \pi(d|x)\pi(x)$. Our simulation algorithm requires that we have available a Gaussian approximation to $\pi(x|d)$ and this is easily obtained by approximating the likelihood mean in each dimension, $ax_i^2 + bx_i$ by the corresponding linear Taylor approximation around the prior mean, m_i. We let $\tilde{\pi}(x|d)$ denote the resulting Gaussian posterior approximation. In the following we use $b = 1$, $m = d = (1, \ldots, 1)^T$, $S_{ij} = \exp\{-(i-j)^2\}$ and $\Sigma_{ij} = \exp\{-|i-j|\}$, and consider three different values for a, 0, 0.1 and 0.3. Clearly, for $a = 0$ the two posterior distributions π and $\tilde{\pi}$ are identical.

The appearance of the proposal distribution (17) that gives unit acceptance probability depends on our choice of distribution for the auxiliary variable u. We consider the two alternatives defined in Section 3, $g_1(u^{(1)}|x)$ and $g_2(u^{(2)}|x)$. As discussed in Section 3, the $g_2(u^{(2)}|x)$ is not analytically available, so here we use the approximation in (25) with a very large value for N so that the result is indistinguishable from $g_2(u^{(2)}|x)$. To map the appearance of (17) we first run until convergence a directional MH algorithm with $g_1(u^{(1)}|x)$ as distribution for the auxiliary variable. At an arbitrary iteration after convergence we stop the simulation, generate a $u^{(1)}$ and the corresponding $u^{(2)}$ and map the resulting two proposal distributions,

$$q_j\left(t|u^{(j)}, x\right) = c\left(x, u^{(j)}\right)\pi\left(x + \sum_{i=1}^{k} t_i u_i^{(j)}\right) g_j\left(u^{(j)} \middle| x + \sum_{i=1}^{k} t_i u_i^{(j)}\right) \tag{26}$$

for $j = 1, 2$. The two $q_1(t|u^{(1)}, x)$ and $q_2(t|u^{(2)}, x)$ are not directly comparable as the same vector t give different proposed values y. To fix this we in stead choose an orthonormal basis for the hyperplane spanned by x and $u^{(1)}$ (or $u^{(2)}$), $v_1, \ldots, v_k$ and consider the two distributions relative to this basis, i.e.

$$r_j\left(t \middle| u^{(j)}, v_1, \ldots, v_k, x\right) \propto \pi\left(x + \sum_{i=1}^{k} t_i v_i\right) g_1\left(u^{(j)} \middle| x + \sum_{i=1}^{k} t_i v_i\right) \tag{27}$$

for $j = 1, 2$. It should be noted that by transforming to an orthonormal basis gives that the Euclidean norm $\|t\|$ is equal to the Euclidean distance between x and the proposed value y in $\mathbb{R}^n$. In Figure 1 we show results for $k = 2$ when $a = 0$ (left column) and $a = 0.3$ (right column). Thus, the target distribution is Gaussian in the left column and non-Gaussian in the right column. The upper row shows the target distribution $\pi(x + t_1 v_1 + t_2 v_2)$ for $t = (t_1, t_2)^T$, $t_1, t_2 \in (-7, 5, 7.5)$. For the same values of t the middle row shows $r_1(t|u^{(1)}, v_1, v_2, x)$ and the lower row $r_2(t|u^{(2)}, v_1, v_2, x)$. The current state x is located in the center in all six plots. We see that in the lower row we get an **O**-shaped distribution. Comparing the upper and lower rows for $a = 0$ we observe that $r_2(t|u^{(2)}, v_1, v_2, x)$ seem to follow the contours of the target distribution. Intuitively it seems reasonable that proposals to other parts of the target distribution with almost the same density as x should often be accepted with high probability. We observe this property also for $a = 0.3$, but here $r_2(t|u^{(2)}, v_1, v_2, x)$ is not symmetrical. Note that the results in the lower row is in accordance with the observation made in Eidsvik and Tjelmeland (2006), namely that the proposal distributions for $k = 1$ typically have two modes, one on each side of a high density area of the target distribution. Here we have a high density area of the target distribution in the center of the **O**-shaped distribution. For $a = 0$, middle row, we see that the two largest modes are placed on each side of a high density area of the target distribution, one close to x and one on the other side of the high density area of the target distribution. Again this is accordance with Eidsvik and Tjelmeland (2006).

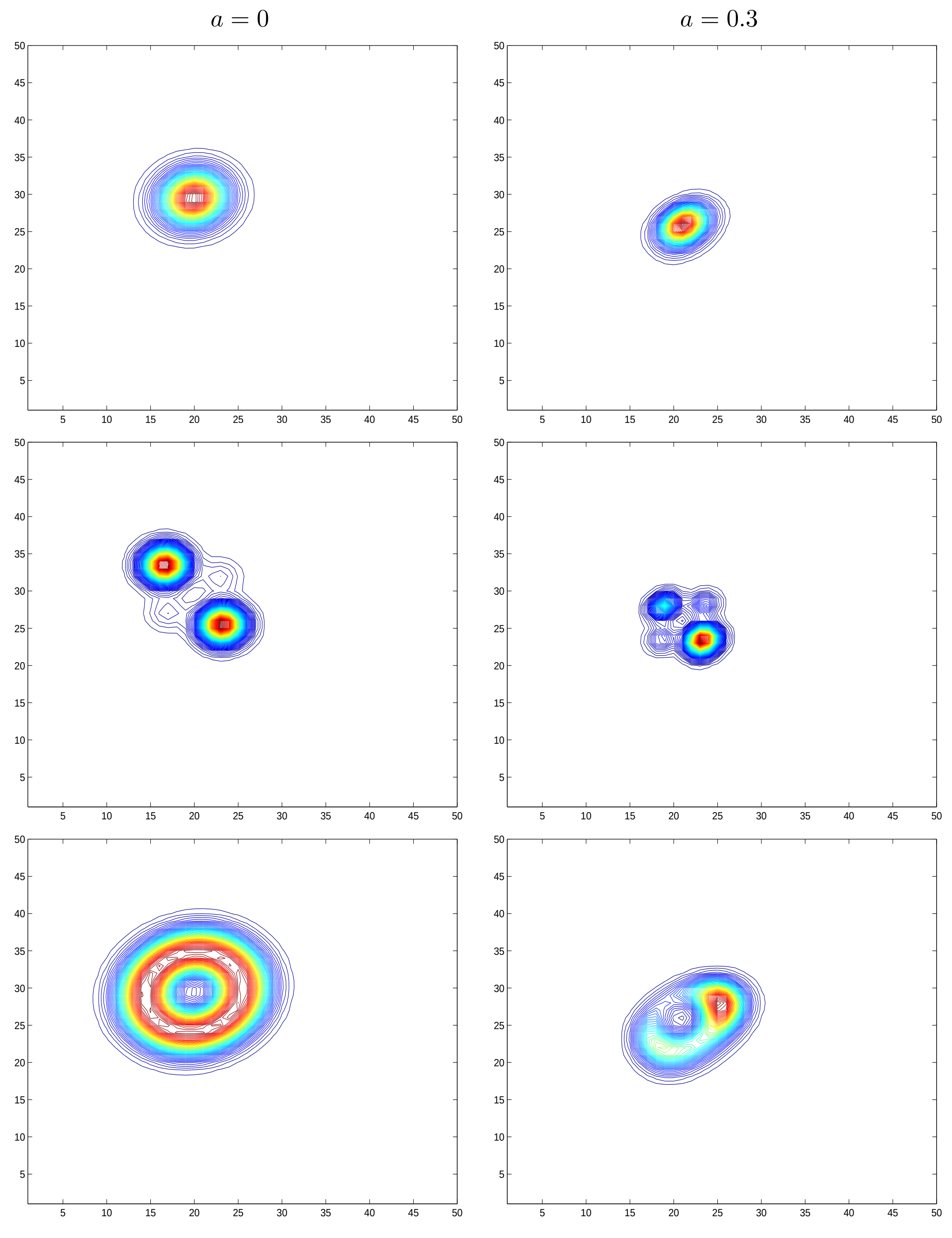


Figure 1: Results for the Bayesian simulation example with $a = 0$ (left column) and $a = 0.3$ (right column). The upper row shows the target distribution $\pi(x+t_1v_1+t_2v_2)$ for $t = (t_1, t_2)^T$, $t_1, t_2 \in (-7, 5, 7.5)$. For the same values of t the middle row shows $r_1(t|u^{(1)}, v_1, v_2, x)$ and the lower row $r_2(t|u^{(2)}, v_1, v_2, x)$. The current state x is in the center, $(25, 25)$, in all of the panels.

Comparing the middle and lower rows for $a = 0$, we observe that the two largest modes in the middle row are parts of the **O**-shaped distribution. This seem to be a general behaviour. We observe the same for $a = 0.3$. Thus, long jumps are possible with both $g_1(u|x)$ and $g_2(u|x)$, but we are loosing something in *where* we are able to propose potential new states by using $u^{(1)}$ based on a non-unique basis relative to using $u^{(2)}$ that is based on a unique basis.

5 Bayesian simulation example

In this Section we analyse the behaviour of the algorithm based on $g_1(u^{(1)}|x)$ introduced above and compare the results with simpler MH algorithms. In turns out that the simulations necessary to calculate a satisfying approximation to (22) by using (25) is too expensive to be of any practical use. We again consider the Bayesian model for non-linear seismic inversion defined in Section 5, but now let $n = 30$.

5.1 Approximating the unit acceptance rate proposal distribution

As the proposal distribution (17) is not analytically available we need to construct an approximation to it and use this as our proposal distribution. From the discussion in Section 4 we know that the proposal distribution (17) typically contains two large modes, one close to $t = 0$ and one further away from $t = 0$. Our strategy is to approximate (17) by a mixture of two Gaussian distributions.

To locate the large mode close to $t = 0$ we start a deterministic minimisation algorithm at $t = 0$, searching for a local minimum for $V(t) = -\ln\{q_1(t|u^{(1)}, x)\}$. Let μ_0 denote the location of local minimum found and let $\Sigma_0 = (\nabla^2 V(\mu_0))^{-1}$ denote the inverse of the Hessian matrix evaluated at μ_0. We use μ_0 and Σ_0 as the mean vector and covariance matrix, respectively, in the first mixture component. We want to use a similar procedure to approximate the second large mode in (17), but then need to use another starting value for t in the minimisation algorithm. From the results discussed in Section 4 we know that the maximal value for $\pi(x + \sum_{i=1}^k t_i u_i)$ (as function of t) is located close to the line going through the two larger local maxima of $q_1(t|u^{(1)}, x)$. Moreover, the maximal value for $\pi(x + \sum_{i=1}^k t_i u_i)$ is located between the two large local maxima of $q_1(t|u^{(1)}, x)$. A natural initial value for the second minimisation run is therefore $t = \mu_0 + 2(\mu_\pi - \mu_0)$, where μ_π is the location of the local maximum of $\pi(x + \sum_{i=1}^k t_i u_i)$. Letting μ_1 denote the location of the resulting second local minimum found for $V(t)$, and letting $\Sigma_1 = (\nabla^2 V(\mu_1))^{-1}$ denote the corresponding inverse Hessian matrix, our approximation to $q_1(t|u^{(1)}, x)$ is

$$\widehat{q}_1(t|u^{(1)}, x) = \frac{q_1(\mu_0|u^{(1)}, x)\mathrm{N}(t; \mu_0, \Sigma_0) + R \cdot q_1(\mu_1|u^{(1)}, x)\mathrm{N}(t; \mu_1, \Sigma_1)}{q_1(\mu_0|u^{(1)}, x) + R \cdot q_1(\mu_1|u^{(1)}, x)} \tag{28}$$

where $\mathrm{N}(t; \mu, \Sigma)$ is the density of the Gaussian distribution with mean vector μ and covariance matrix Σ evaluated at t, and R is a scalar constant that can make proposals to the mode far away from the current state more or less likely. The results in Roberts et al. (1997) and Roberts and Rosenthal (1998) indicate that a value of $R > 1$ should be preferable. For $a = 0$ simulations show that the acceptance rate is close to one for all choices of R. In this case it is therefore best always to propose the mode away from the current state, i.e. use a large value for R. For $a \neq 0$ the situation is more complicated. In all the simulations of the directional MH algorithm we have tuned the parameter R to the optimal value in terms of mean jump length. It turns out that high acceptance rates, over 50%, works best. Before presenting further simulation results for the proposal distribution just defined we give a few remarks.

Remark 1. Simulation show that for $R = 1$ the algorithm results in acceptance rates over 95% for $k = 1, 2, 3, 4$, so the approximation in (28) is good.

Remark 2. If we use a quasi-Newton optimisation algorithm to find μ_0 and μ_1, the algorithm in addition to finding the minimum also build up an approximation to the Hessian matrix that can be used as Σ_0 and Σ_1.

Remark 3. We motivate our approximation $\widehat{q}_1(t|u^{(1)}, x)$ from the results in Section 4, that is for $k = 2$. However, we have done similar studies for $k = 3$, and also there $q_1(t|u^{(1)}, x)$ typically seems to contain two large modes.

5.2 Performance of the directional MH algorithm for $k \geq 1$

To evaluate the performance of the directional MH algorithm with proposal distribution $\widehat{q}_1(t|u^{(1)}, x)$ we run the algorithm for $k = 1, 2, 3, 4$ and compare the results with the results from the simpler random walk (RW) and Langevin (L) proposal algorithms. We tuned the parameters to get close to the optimal acceptance rates of 0.23 and 0.57, respectively, calculated in Roberts et al. (1997) and Roberts and Rosenthal (1998). The results are summarised in Figure 2 and Table 1. In Figure 2 we give estimated autocorrelation functions for the various algorithms considered. The left column shows the autocorrelation as a function of iterations, whereas in the right column the autocorrelation is shown as function of number of target density evaluations. The three rows show results for $a = 0$, $a = 0.1$ and $a = 0.3$, respectively. Table 1 gives some key attributes for the runs. For $a = 0$ we see that the directional MH works better then both RW and L, both when measured in number of iterations and in number of target density evaluations. For $k > 1$ we even get negative lag one correlations. For $a = 0.1$ the directional MH algorithms are better when measured in number of iterations, but when evaluated in terms of the number of target density evaluations the Langevin algorithm is clearly preferable. For $a = 0.1$ the directional MH algorithm with $k = 1$ and RW performs equally good in terms of the number of target density evaluations. The directional MH with $k > 1$ are less favourable in this case. Note in Table 1 that using $k > 1$ does not generate much longer jumps than $k = 1$. As $k > 1$ requires more target density evaluations than $k = 1$, the $k > 1$ is less attractive in this case. For $a = 0.3$ the qualitative performances of the various algorithms are similar to with $a = 0.1$, but the differences are larger.

6 Closing remarks

Directional MH algorithms are generalised to propose new states in a hyperplane in this article. The hyperplane is specified by the current state and auxiliary variables. We consider two types of auxiliary variables, one is uniquely given by the hyperplane and the other is not. Consistent with the findings in Eidsvik and Tjelmeland (2006), our analysis indicates that it is preferable to use the auxiliary variables that is uniquely given by the hyperplane. However, we are not able to do the analytical analysis necessary to implement the MH algorithm in this case. Therefore we instead consider the MH algorithm corresponding to the auxiliary variables that are not uniquely defined by the hyperplane and compare its performance with frequently used simpler alternatives in a simulation example. The experience is that hyperplane algorithms on average produce larger jumps in the sample space and thereby better mixing properties per

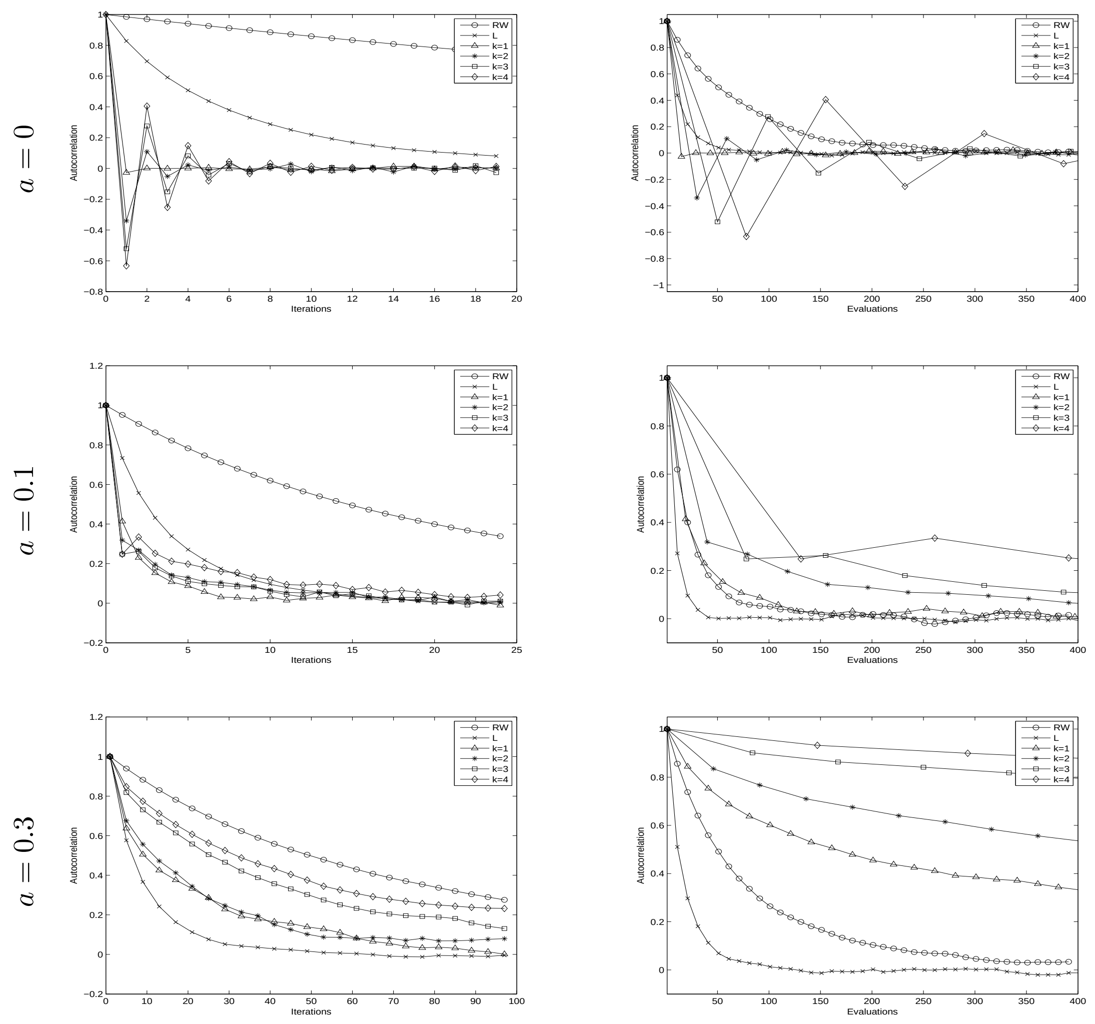


Figure 2: Estimated autocorrelation functions for random walk and Langevin proposal MH algorithms, and for our directional MH algorithm with $k = 1, 2, 3, 4$. Left column: Autocorrelation as a function of iteration. Right column: Autocorrelation as a function of number of target evaluations. The upper, middle and lower rows shows results for $a = 0$, $a = 0.1$ and $a = 0.3$, respectively.

Table 1: Attributes for the random walk and Langevin proposal MH algorithms, and for the directional MH algorithm for $k = 1, 2, 3, 4$

	Mean number of target evaluations per iteration	Mean acceptance rates	Mean jump length	Evaluations per independent sample
$a = 0$:				
RW	1	0.23	0.33	300
L	2	0.54	1.61	75
$k = 1$	14	0.99	5.5	30
$k = 2$	29	0.99	6.4	150
$k = 3$	49	0.99	6.8	300
$k = 4$	77	0.99	7.1	750
$a = 0.1$:				
RW	1	0.25	0.31	200
L	2	0.58	1.1	50
$k = 1$	18	0.61	3.0	150
$k = 2$	39	0.58	3.2	600
$k = 3$	77	0.56	3.3	1200
$k = 4$	130	0.56	3.3	2500
$a = 0.3$:				
RW	1	0.25	0.28	400
L	2	0.58	0.96	100
$k = 1$	20	0.51	1.0	2000
$k = 2$	45	0.64	1.0	2700
$k = 3$	83	0.73	0.94	$15 \cdot 10^3$
$k = 4$	146	0.80	0.90	$600 \cdot 10^3$

iteration. However, in our implementation the hyperplane algorithms is more computation intensive per iteration and so the simpler algorithms in most cases are preferable when run for the same amount of computation time. An interesting area for future research is therefore to find variants of our directional Metropolis–Hastings algorithm that require less computation time per iteration.

References

Chen, M. and Schmeiser, B. (1993). Performance of the Gibbs, hit-and-run and Metropolis samplers, *Journal of computational and graphical statistics* **2**: 251–272.

Eidsvik, J. and Tjelmeland, H. (2006). On directional Metropolis–Hastings algorithms, *Statistics and Computing* **16**: 93–106.

Gilks, W. R., Roberts, G. O. and George, E. I. (1994). Adaptive directional sampling, *The Statistician* **43**: 179–189.

Goodman, J. and Sokal, A. D. (1989). Multigrid Monte Carlo method. Conceptual foundations, *Physical Review D* **40**: 2035–2072.

Green, P. (1995). Reversible jump Markov chain Monte Carlo computations and Bayesian model determination, *Biometrica* **57**: 97–109.

Hastings, W. (1970). Monte Carlo sampling using Markov chains and their applications, *Biometrica* **57**: 97–109.

Liu, J. S. and Sabatti, C. (2000). Generalized Gibbs sampler and multigrid Monte Carlo for Bayesian computation, *Biometrika* **87**: 353–369.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines, *Journal of Chemical Physics* **21**: 1087–1092.

Pukkila, T. M. and Rao, C. R. (1988). Pattern recognition based on scale invariant discriminant functions, *Information sciences* **45**: 379–389.

Rabben, T. E., Tjelmeland, H. and Ursin, B. (2008). Nonlinear Bayesian joint inversion of seismic reflection coefficients, *Geophysical Journal International* p. To appear.

Roberts, G. O., Gelman, A. and Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms, *Annals of Applied Probability* **7**: 110–120.

Roberts, G. O. and Gilks, W. R. (1994). Convergence of adaptive directional sampling, *Journal of multivariate analysis* **49**: 287–298.

Roberts, G. O. and Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions, *Journal of the Royal Statistical Society. Series B* **60**: 255–268.

Waagepetersen, R. and Sørensen, D. (2001). A tutorial on reversible jump MCMC with a view toward application in QTL-mapping, *International Statistical Review* **69**: 49–61.

Watson, G. S. (1983). *Statistics on spheres*, Wiley-Interscience.

Paper III

Approximate forward-backward algorithm for a switching linear Gaussian model – with application to seismic inversion

Hugo Hammer and Håkon Tjelmeland
Department of Mathematical Sciences
Norwegian University of Science and Technology
Trondheim, Norway

Abstract

Motivated by the application of seismic inversion in the petroleum industry we consider a hidden Markov model with two hidden layers. The bottom layer is a Markov chain and given this the variables in the second hidden layer are assumed conditionally independent and Gaussian distributed. The observation process is assumed Gaussian with mean values that are linear functions of the second hidden layer. This model class, which we call switching linear Gaussian models, has clear similarities with what is known as switching linear dynamical systems and conditionally Gaussian state space models. The important difference is that we allow the observations to depend on both past and future values of the hidden Gaussian process. The forward-backward algorithms is not directly feasible for switching linear Gaussian models as the recursions then involve a mixture of Gaussian densities where the number of terms grows exponentially with the length of the Markov chain. We propose an approximate forward-backward algorithm by dropping the less important terms and thereby obtain a computationally feasible algorithm that generates samples from an approximation to the conditional distribution of the unobserved layers given the data. We also use this approximation as a proposal distribution in a Metropolis–Hastings setting and obtain high acceptance rates and good mixing properties. We demonstrate the effectiveness and quality of the approximation in simulation examples.

1 Introduction

Hidden Markov models combined with recursive algorithms have successfully been used in many areas, see the discussion and references in MacDonald and Zucchini (1997), Künsch (2000), Scott (2002) and Cappé et al. (2005). In a hidden Markov model the observations are assumed to be an incomplete and noisy function of an underlying unobserved process, where the latent unobserved process has a discrete state space and is assumed to be Markov. The goal is typically to restore the underlying Markov chain from the noisy observations and perhaps also to estimate unknown parameters in both the latent and observation processes. With reasonable assumptions for the observation process these goals are achieved via efficient recursive computations known as the forward-backward algorithms, corresponding to the famous Kalman filter equations when the latent Markov process is Gaussian.

Hidden Markov models with two hidden layers have been considered by several authors. Switching linear dynamical systems (Zoeter and Heskes, 2006; Bar-Shalom and Li, 1998) and switching state space models (Barber, 2006) are hidden Markov models with two latent layers. The bottom hidden layer is a discrete state space Markov chain and conditioned on this bottom layer the second hidden layer is a Gaussian Markov process. Finally, given the two hidden layers the observations are assumed independent and Gaussian. The mean vector and covariance matrix for the observed value at any time index are functions of the two unobserved states at the same time index. Larsen et al. (2006) consider a similar model, but allow the observations to be functions of both past and future values of the hidden Gaussian process. The goal is again to restore the unobserved Markov chain. The forward-backward recursions can be formulated also for these models, but they are not computationally feasible as the forward recursion involves a mixture of Gaussian distributions where the number of terms grows exponentially with the length of the Markov chain. In the two first references given above, approximate forward recursions are defined by substituting the mixture of Gaussian distributions by a single Gaussian term. Larsen et al. (2006) define approximate recursions by approximating the marginal distribution for the unobserved continuous process by a product of Gaussian densities.

Motivated by the application of seismic inversion in the petroleum industry, we consider the same model as discussed in Larsen et al. (2006). Here the Markov chain represents lithology-fluid classes along a vertical trace through the underground, the intermediate Gaussian layer represents elastic parameters of the rock along the same trace, and the observations are the seismic data. The focus is again to restore the unobserved layers and particularly the Markov chain. Parameter estimation is clearly also of interest, but not considered here. To get a computationally feasible algorithm we must again consider approximate forward-backward algorithms. In the forward part of the forward-backward algorithm we propose to drop terms associated with small weights in the Gaussian mixture. Thus, our approximation is less dramatic than what is used before, but to a somewhat higher computational cost. The quality of the approximation depends on the number and importance of the terms that are dropped. Using the output of the resulting approximate forward-integration-backward-simulation algorithm as proposals in a Metropolis–Hastings setting (Hastings, 1970) we are able to correct for the induced approximation and the associated Metropolis–Hastings acceptance rate is a natural measure of the quality of the approximation.

The paper is organised as follows. Section 2 defines and introduces necessary notation for what we call the switching linear Gaussian model. In Section 3 we give a brief introduction to the seismic inversion application and explain how the switching linear Gaussian model constitutes the core part of this model. In Section 4 we first develop the exact forward recursions for the switching linear Gaussian model, then discuss the dropping Gaussian terms approximation and develop the backward simulation algorithm. In Section 5 we evaluate the approximate forward–backward algorithm in simulation examples. Finally, in Section 6 we provide conclusions.

2 The switching linear Gaussian model

In this paper we represent (multivariate) Gaussian distributions in its canonical form, as this simplify the forward-backward recursions. A Gaussian distribution with mean vector μ and covariance matrix Σ is in its canonical form parametrised by the precision matrix $Q = \Sigma^{-1}$ and the vector $q = Q\mu$, and we use $\mathrm{N}(q, Q)$ to denote this distribution. The corresponding density we denote by $\mathrm{N}(u|q, Q)$ and this reads

$$\mathrm{N}(u|q,Q) = \frac{\sqrt{|Q|}}{(2\pi)^{\frac{r}{2}}} \exp\left\{-\frac{1}{2}q^T Q^{-1} q\right\} \exp\left\{-\frac{1}{2}\left[u^T Q u - 2q^T u\right]\right\}. \tag{1}$$

Consider a three layer hidden Markov model $\{(x_i, y_i, z_i)\}_{i=1}^n$ as visualised in Figure 1, where $x_i \in \{1, \ldots, L\}$, $y_i = (y_{i1}, \ldots, y_{ir})^T \in \mathbb{R}^r$ and $z_i = (z_{i1}, \ldots, z_{is})^T \in \mathbb{R}^s$ for $i = 1, \ldots, n$. We require the number of possible values for the x_i's, L, to be small. In the seismic data

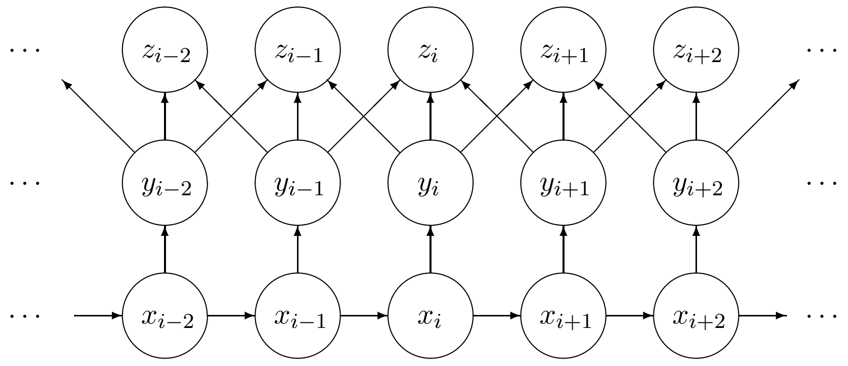


Figure 1: Directed acyclic graph representation of the hidden Markov model discussed in Sections 2 and 4.

example in Section 5 we have $L = 4$. We assume $x = (x_1, \dots, x_n)$ to be a stationary, aperiodic and ergodic Markov chain with transition matrix

$$\mathrm{P} = [p(x_i|x_{i-1})]_{x_{i-1},x_i=1}^{L}. \tag{2}$$

Thus, the marginal distribution of x_1 equals the limiting distribution induced by P, which we denote by $p(x_1)$. Conditioned on x we assume the elements of $y = (y_1, \dots, y_n)$ to be independent and Gaussian distributed, where the (conditional) mean vector and precision matrix of y_i are (known) functions of x_i, which we denote by $\mu(x_i)$ and $Q(x_i)$, respectively. Thus,

$$y_i|x \sim \mathrm{N}(q(x_i), Q(x_i)), \tag{3}$$

where $q(x_i) = Q(x_i)\mu(x_i)$. Given y we assume the elements of the third layer, $z = (z_1, \dots, z_n)$, to be independent of x and independent of each other. The conditional distribution for z_i is Gaussian with mean vector $a_i^T y_{i-1} + b_i^T y_i + c_i^T y_{i+1}$ and precision matrix R_i, i.e.

$$z_i|y \sim \mathrm{N}\left(A_i^T y_{i-1} + B_i^T y_i + C_i^T y_{i+1}, R_i\right), \tag{4}$$

where $A_i = a_i R_i$, $B_i = b_i R_i$ and $C_i = c_i R_i$. Note that we allow the coefficient matrices a_i, b_i and c_i to vary with i, and in particular we of course require $a_1 = c_n = 0$.

3 The seismic inversion application

Seismic inversion is the discipline of predicting lithology-fluid characteristics in a reservoir from seismic data. Numerous introductory books to seismic terminology and inversion exist, see for example Sheriff and Geldart (1995).

Seismic data is created by an explosion which sends sound waves into the ground. Parts of the waves are reflected, returned upwards and observed by microphones (geo- or hydrophones). These observations are the basis for the seismic data. A forward model, describing what we observe for given lithology–fluid characteristics, is known from physics theory. In seismic inversion we are interested in the corresponding inverse problem, predicting the lithology–fluid characteristics given observed seismic data.

Seismic inversion is typically done based on either pre-stack or post-stack data. The pre-stack data are also referred to as amplitude versus offset (AVO) data or common midpoint gather (CMP). In a CMP gather, seismic data with different offsets (reflection angles) related to the same vertical profile are gathered together. Before the data is used for inversion, noise effects like moveout, multiples and the effect of geometrical spreading and absorption are removed. The data is also pre-stack migrated such that any dip related effects are removed. Post-stack data are generated from pre-stack data using the method of "stacking" (Sheriff and Geldart, 1995). Here we consider how to do seismic inversion from one vertical profile of pre-stack seismic data.

Our forward model is similar to the ones in Buland et al. (2003) and Larsen et al. (2006). When dealing with seismic data the depth is typically not referenced by distance from the surface, but by the time used by the sound wave from the surface to a location in the underground and back, called travel time and denoted by t. A problem not considered here is how to convert travel times to depths. Let the discrete variable $x(t)$ be the lithology–fluid class in the reservoir at travel time t. Assuming an isotropic and elastic medium, the material properties are uniquely defined by the continuous variables P-wave velocity, S-wave velocity and density. At time t we let

$$\begin{aligned} y(t,1) &= \ln(\alpha(t)) \\ y(t,2) &= \ln(\beta(t)) \\ y(t,3) &= \ln(\rho(t)) \end{aligned} \tag{5}$$

denote the logarithms of each of these three quantities, respectively. The forward relation from x to y is based on the so-called Gassman equations (Berryman, 1995), we return to this below. For each depth t and offset (reflection angle) θ a reflection coefficient, $r(t, \theta)$, results from y. For this we use what is known as a weak contrast approximation (Aki and Richards, 1980; Buland and Omre, 2003), which reads

$$r(t,\theta) = \gamma_\alpha(\theta)\frac{\partial}{\partial t}y(t,1) + \gamma_\beta(\theta)\frac{\partial}{\partial t}y(t,2) + \gamma_\rho(\theta)\frac{\partial}{\partial t}y(t,3), \tag{6}$$

where

$$\gamma_\alpha(\theta) = \frac{1}{2}(1+\tan^2(\theta)) \tag{7}$$

$$\gamma_\beta(\theta) = -4\overline{\beta/\alpha}^2\sin^2(\theta) \tag{8}$$

$$\gamma_\rho(\theta) = -4\left(\overline{\beta/\alpha}^2\sin^2(\theta)\right), \tag{9}$$

and one has assumed the ratio $\beta(t)/\alpha(t)$ to have an approximately constant value, $\overline{\beta/\alpha}$, in the reservoir. Seismic data depends on the reflection coefficients and is observed for each travel time t and offset θ. Still following Buland et al. (2003) and Larsen et al. (2006), we use a convolution model with additive noise for this,

$$d(t,\theta) = \int w(u,\theta)\cdot r(t-u,\theta)\mathrm{d}u + \varepsilon(t,\theta), \tag{10}$$

where $w(u,\theta)$ is a wavelet function and $\varepsilon(t,\theta)$ is Gaussian noise. Similar to Buland and Omre (2003) we assume that the main part of the Gaussian noise has a correlation structure corresponding to the wavelet. The argument for this is that both the signal and noise parts are the results of sound waves going through the (same) underground.

For doing the inversion we consider a discrete version of the forward model discussed above. We use $i = 1, \dots, n$ to denote n travel times along the vertical profile and let x_i and $y_i \in \mathbb{R}^3$ denote corresponding values for $x(t)$ and $y(t)$. We consider s offset values $\theta_1, \dots, \theta_s$ and let $r_i = (r_{i1}, \dots, r_{is})^T \in \mathbb{R}^s$ and $d_i = (d_{i1}, \dots, d_{is})^T \in \mathbb{R}^s$ denote reflection coefficients and seismic data at travel time i, respectively. As prior for $x = (x_1, \dots, x_n)^T$ we use a Markov chain, as specified in (2). The distribution for $y = (y_1^T, \dots, y_n^T)^T$ given x is, as mention above, based on the Gassman equations and we assume Gaussian distributions as specified in (3). Using a central difference approximation for the partial derivatives in (6) we get an expression for $r = (r_1^T, \dots, r_n^T)^T$ as a function of y,

$$r_i = \Gamma^T \cdot \frac{y_{i+1} - y_{i-1}}{2} \text{ for } i = 1, \dots, n, \tag{11}$$

where

$$\Gamma = \begin{bmatrix} \gamma_\alpha(\theta_1) & \gamma_\alpha(\theta_2) & \cdots & \gamma_\alpha(\theta_s) \\ \gamma_\beta(\theta_1) & \gamma_\beta(\theta_2) & \cdots & \gamma_\beta(\theta_s) \\ \gamma_\rho(\theta_1) & \gamma_\rho(\theta_2) & \cdots & \gamma_\rho(\theta_s) \end{bmatrix}. \tag{12}$$

To avoid boundary problems for $i = 1$ and $i = n$, we use forward and backward difference approximations, respectively, for the derivatives there.

Finally we discretise (10) to get the distribution for $d = (d_1^T, \dots, d_n^T)^T$ given r,

$$d_{ij} = \sum_{u=-k}^{k} w(u, \theta_j) \cdot r_{i-u,j} + \varepsilon_{ij}, \tag{13}$$

where the zero mean Gaussian noise ε_{ij} is given as

$$\varepsilon_{ij} = \sum_{u=-k}^{k} w(u, \theta_j) \varepsilon^1_{i-u,j} + \varepsilon^2_{i,j} \tag{14}$$

and $\varepsilon^1_{i,j}$ and $\varepsilon^2_{i,j}$ are independent Gaussian white noise and we assume the variance of ε^1_{ij} to be much larger than the variance in ε^2_{ij} so that most of the noise are on the same form as the wavelet. Clearly, we also assume the wavelet to be nonzero only in a finite interval so that the sum in (13) has a finite number of terms.

3.1 Simulating from the seismic model

We are now interested in simulating the variables x and y conditioned on the data d from the model above. To define an effective MCMC algorithm for this, we introduce $z = (z_1, \dots, z_n)^T$, where $z_i = (z_{i1}, \dots, z_{is})^T \in \mathbb{R}^s$ for $i = 1, \dots, n$,

$$z_i = r_i + \varepsilon^1_i \tag{15}$$

and $\varepsilon^1_i = (\varepsilon^1_{i1}, \dots, \varepsilon^1_{is})^T$. The distribution for x, y and z is then on the form specified in Section 2. In (4) we have $A_i = -\Gamma\, R_i/2$, $B_i = \mathbf{0}$ and $C_i = \Gamma\, R_i/2$ for $i = 2, \dots, n-1$, and using forward and backward difference at the boundaries $B_1 = -\Gamma\, R_1$, $C_1 = \Gamma\, R_1$, $A_n = -\Gamma\, R_n$ and

$B_n = \Gamma\, R_n$. Finally we use $R_i = \sigma_1^{-2} I$ for $i = 1, \ldots, n$, where I denote the identity matrix. Combining (13), (14) and (15) we get the relation between z and d,

$$d_{ij}|z \sim \mathrm{N}\left(\sigma_2^{-2} \sum_{u=-k}^{k} w(u, \theta_j) \cdot z_{i-u,j}, \sigma_2^{-2} I\right). \tag{16}$$

To simulate effectively from the specified model we use a Metropolis–Hastings algorithm (Han and Green, 1992; Hastings, 1970) consisting of two steps in each iteration. In the first step we use a joint Gibbs update for y and z. The joint full conditional for y and z is Gaussian and thereby easy to sample from. The full conditional for x and y is a model of the type specified in Section 2. In the next section we discuss how to generate samples from an approximation to this distribution and we adopt this as proposal distribution.

Remark 1. A perhaps more direct Metropolis–Hastings algorithm would be a three block Gibbs algorithm, updating each of x, y and z separately. The full conditionals for y and z are Gaussian and to sample the full conditional for x the (standard and exact) forward-backward algorithm can be used. However, this algorithm has an extremely slow mixing because x and y are highly correlated. This is the reason we in stead propose the two step algorithm specified above.

Remark 2. A variant of the two block Metropolis–Hastings algorithm defined above is to propose new values only for parts of x and y in each iteration. In simulations we have found that one has to adopt this alternative if n is large, as otherwise either the approximate proposal distribution is not computational feasible or the resulting Metropolis–Hastings acceptance rate becomes very low.

4 Simulation algorithm

In this section we define the approximate forward-backward algorithm for the model defined in Section 2 and discuss how this can be used as a proposal distribution in a Metropolis–Hastings algorithm. We start by deriving the exact forward-integration recursion.

4.1 Forward integration

The conditional distribution of interest is

$$\pi(x, y|z) \propto \pi(x)\pi(y|x)\pi(z|y) = p(x_1) \cdot \prod_{i=2}^{n} p(x_i|x_{i-1}) \cdot \prod_{i=1}^{n} \mathrm{N}\left(y_i|q(x_i), Q(x_i)\right) \tag{17}$$

$$\cdot \prod_{i=1}^{n} \mathrm{N}\left(z_i|A_i^T y_{i-1} + B_i^T y_i + C_i^T y_{i+1}, R_i\right).$$

To make the forward integration easier we need to introduce more notation. Let $T_1(x_1, y_1, x_2, y_2, y_3)$ be a product of all the factors in $\pi(x, y|z)$ containing x_1 or y_1, let $T_2(x_2, y_2, x_3, y_3, y_4)$ be a product of all factors in $\pi(x, y|z)$ containing x_2 or y_2 and that is not already included in $T_1(x_1, y_1, x_2, y_2, y_3)$, and so on. Thus, by adopting the conventions $p(x_{n+1}|x_n) = 1$ and $B_{n+1} = C_{n+1} = 0$ we have

$$\pi(x, y|z) \propto \prod_{i=1}^{n} T_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}), \tag{18}$$

with

$$\begin{aligned}
&T_1(x_1, y_1, x_2, y_2, y_3) = p(x_1)p(x_2|x_1) \\
&\qquad \mathrm{N}(y_1|q(x_1), Q(x_1))\mathrm{N}(z_1|B_1y_1 + C_1y_2, R_1)\mathrm{N}(z_2|A_2^Ty_1 + B_2^Ty_2 + C_2^Ty_3, R_2), \qquad (19) \\
&T_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) = \\
&\qquad p(x_{i+1}|x_i)\mathrm{N}(y_i|q(x_i), Q(x_i))\mathrm{N}(z_{i+1}|A_{i+1}^Ty_i + B_{i+1}^Ty_{i+1} + C_{i+1}^Ty_{i+2}, R_{i+1}) \qquad (20)
\end{aligned}$$

for $i = 2, \dots, n$. Recalling $C_n = 0$ and the notational conventions just made we see that none of the above functions really depends on x_{n+1}, y_{n+1} or y_{n+2}. We include these in the argument lists just to simplify the notation. Now define $U_i(x_i, y_i,\ x_{i+1}, y_{i+1}, y_{i+2})$ and $V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ for $i = 1, \dots, n-1$ so that

$$\begin{aligned}
\pi(x_i, \dots, x_n, y_i, \dots, y_n|z) &\propto U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) \\
&\cdot \prod_{j=i+1}^{n} T_j(x_j, y_j, x_{j+1}, y_{j+1}, y_{j+2}), \qquad (21) \\
\pi(x_i, x_{i+1}, \dots, x_n, y_{i+1}, \dots, y_n|z) &\propto V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2}) \\
&\cdot \prod_{j=i+1}^{n} T_j(x_j, y_j, x_{j+1}, y_{j+1}, y_{j+2}). \qquad (22)
\end{aligned}$$

This gives $U_1(x_1, y_1, x_2, y_2, y_3) = T_1(x_1, y_1, x_2, y_2, y_3)$,

$$U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) = T_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) \sum_{x_{i-1}=1}^{L} V_{i-1}(x_{i-1}, x_i, y_i, y_{i+1}) \qquad (23)$$

for $i = 2, \dots, n-1$, and

$$V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2}) = \int U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2})\mathrm{d}y_i \qquad (24)$$

for $i = 1, \dots, n$. Note that none of the U_i or V_i functions are really functions of x_{n+1}, y_{n+1} or y_{n+2}, but again we include them as arguments to simplify the notation. In the following we use $\mathbf{0}$ and $\mathbf{I}$ to denote an $r \times r$ matrix with all elements equal to zero and the r-dimensional identity matrix, respectively. Moreover, let D_i and F_i be $3r \times 3r$ and $2r \times 2r$ identity matrices, respectively, for $i = 1, \dots, n-2$, and define

$$D_{n-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix},\ D_n = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \text{ and } F_{n-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}. \qquad (25)$$

The following Theorem presents the forward recursions for $i = 1, \dots, n$.

Theorem 1. *Consider the hidden Markov model defined in Section 2 and the notation intro-*

duced above. We then have

$$\begin{aligned} U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) &\propto p(x_{i+1}|x_i) \\ &\quad \cdot \sum_{j=1}^{N_i} f_{ij}(x_i) N\left(D_i \begin{bmatrix} y_i \\ y_{i+1} \\ y_{i+2} \end{bmatrix} \middle| g_{ij}(x_i), G_{ij}(x_i) \right) \quad \textit{for } i = 1, \dots, n \end{aligned} \tag{26}$$

$$\begin{aligned} V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2}) &\propto p(x_{i+1}|x_i) \\ &\quad \cdot \sum_{j=1}^{N_i} f_{ij}(x_i) N\left(F_i \begin{bmatrix} y_{i+1} \\ y_{i+2} \end{bmatrix} \middle| k_{ij}(x_i), K_{ij}(x_i) \right) \quad \textit{for } i = 1, \dots, n-1 \end{aligned} \tag{27}$$

$$V_n(x_n, x_{n+1}, y_{n+1}, y_{n+2}) \propto \sum_{j=1}^{N_n} f_{nj}(x_n), \tag{28}$$

where $N_i = L^{i-1}$ and $f_{ij}(x_i)$, $g_{ij}(x_i)$, $G_{ij}(x_i)$, $k_{ij}(x_i)$, and $K_{ij}(x_i)$ can be computed recursively. Initial values for $g_{11}(x_1)$, $G_{11}(x_1)$ and $f_{11}(x_1)$ are given by

$$G_{11}(x_1) = \begin{bmatrix} Q(x_1)+B_1R_1^{-1}B_1^T+A_2R_2^{-1}A_2^T & B_1R_1^{-1}C_1^T+A_2R_2^{-1}B_2^T & A_2R_2^{-1}C_2^T \\ C_1R_1^{-1}B_1^T+B_2R_2^{-1}A_2^T & C_1R_1^{-1}C_1^T+B_2R_2^{-1}B_2^T & B_2R_2^{-1}C_2^T \\ C_2R_2^{-1}A_2^T & C_2R_2^{-1}B_2^T & C_2R_2^{-1}C_2^T \end{bmatrix} \tag{29}$$

$$g_{11}(x_1) = \begin{bmatrix} q(x_1)+B_1z_1+A_2z_2 \\ C_1z_1+B_2z_2 \\ C_2z_2 \end{bmatrix} \tag{30}$$

$$f_{11}(x_1) = p(x_1)\sqrt{|Q(x_1)|}\exp\left\{-\frac{1}{2}\left(q(x_1)^TQ(x_1)^{-1}q(x_1)\right)\right\}. \tag{31}$$

For $i = 2, \dots, n-2$, $j = 1, \dots, N_i$ and $l = 1, \dots, L$, we have the following recursions

$$\begin{aligned} G_{i,(j-1)L+l}(x_i) &= \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} Q(x_i) \left[\boldsymbol{I}\ \boldsymbol{0}\ \boldsymbol{0}\right] + \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} K_{i-1,j}(l) \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \\ &\quad + \begin{bmatrix} A_{i+1} \\ B_{i+1} \\ C_{i+1} \end{bmatrix} R_{i+1}^{-1} \left[A_{i+1}^T\ B_{i+1}^T\ C_{i+1}^T\right], \end{aligned} \tag{32}$$

$$g_{i,(j-1)L+l}(x_i) = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} q(x_i) + \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} k_{i-1,j}(l) + \begin{bmatrix} A_{i+1} \\ B_{i+1} \\ C_{i+1} \end{bmatrix} z_{i+1}, \tag{33}$$

$$K_{i,(j-1)L+l}(x_i) = \left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} G_{i,(j-1)L+l}(x_i)^{-1} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \right)^{-1}, \tag{34}$$

$$k_{i,(j-1)L+l}(x_i) = K_{i,(j-1)L+l}(x_i) \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} G_{i,(j-1)L+l}(x_i)^{-1} g_{i,(j-1)L+l}(x_i). \tag{35}$$

For the final two iterations, $i = n-1$ and n, the recursions are slightly different. For $j =$

$1, \ldots, N_{n-1}$ and $l = 1, \ldots, L$ we have

$$G_{n-1,(j-1)L+l}(x_{n-1}) = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} Q(x_{n-1}) \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} + K_{n-2,j}(l) + \begin{bmatrix} A_n \\ B_n \end{bmatrix} R_n^{-1} \begin{bmatrix} A_n^T & B_n^T \end{bmatrix} \quad (36)$$

$$g_{n-1,(j-1)L+l}(x_{n-1}) = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} q(x_{n-1}) + k_{n-2,j}(l) + \begin{bmatrix} A_n \\ B_n \end{bmatrix} z_n \quad (37)$$

$$K_{n-1,(j-1)L+l}(x_{n-1}) = \left(\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} G_{n-1,(j-1)L+l}(x_{n-1}) \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \right)^{-1} \quad (38)$$

$$k_{n-1,(j-1)L+l} = K_{n-1,(j-1)L+l}(x_{n-1}) \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} G_{n-1,(j-1)L+l}(x_{n-1})^{-1} g_{n-1,(j-1)L+l}(x_{n-1}), \quad (39)$$

and for $j = 1, \ldots, N_n$ and $l = 1, \ldots, L$

$$G_{n,(j-1)L+l}(x_n) = Q(x_n) + K_{n-1,j}(l) \quad (40)$$

$$g_{n,(j-1)L+l}(x_n) = q(x_n) + k_{n-1,j}(l). \quad (41)$$

Finally, the recursion for $f_{ij}(x)$ is, for $i = 2, \ldots, n$, $j = 1, \ldots, N_{n-1}$ and $l = 1, \ldots, L$

$$f_{i,(j-1)L+l}(x_i) = f_{i-1,j}(l) p(x_i|l) \sqrt{\frac{|Q(x_i)K_{i-1,j}(l)|}{|G_{i,(j-1)L+l}(x_i)|}} \exp \left\{ -\frac{1}{2} \left[q(x_i)^T Q(x_i)^{-1} q(x_i) + \right. \right. \quad (42)$$
$$\left. \left. k_{i-1,j}(l)^T K_{i-1,j}(l) k_{i-1,j}(l) - g_{i,(j-1)L+l}(x_i)^T G_{i,(j-1)L+l}(x_i)^{-1} g_{i,(j-1)L+l}(x_i) \right] \right\}.$$

The theorem is proved by induction. Reordering terms in $T_1(x_1, y_1, x_2, y_2, y_3)$ straightforwardly gives (26) for $i = 1$ and initial values (29), (30) and (31). Integrating out y_i in (26) and using well known properties of the multivariate Gaussian distribution gives (27) and (28), and the recursion formulas (34), (35), (38) and (39). We represent Gaussian distributions in the canonical form and not by the mean vector and covariance matrix and this is the reason for the somewhat unfamiliar expressions in these four equations. Summing over x_i in (27) and renumbering terms give after straightforward but tedious calculations (26) and the recursions (32), (33), (36), (37), (40), (41) and (42).

The number of Gaussian terms in (26) and (27) grows exponentially with i and the recursive algorithm defined by Theorem 1 is thereby computationally feasible only for very small values of n. In the next section we discuss how to approximate $U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2})$ and $V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ by ignoring less important terms in (26) and (27).

4.2 Approximate forward integration algorithm

In this section we propose an approximate and computationally feasible version of the recursions defined in Theorem 1. We first compute the (exact) representation of $U_1(x_1, y_1, x_2, y_2, y_3)$ and $V_1(x_1, x_2, y_2, y_3)$ as given in Theorem 1. When starting iteration $i - 1 \leq n - 2$ of the sequential algorithm we have available an approximation of $V_{i-1}(x_{i-1}, x_i, y_i, y_{i+1})$ on the form

$$\widetilde{V}_{i-1}(x_{i-1}, x_i, y_i, y_{i+1}) \propto p(x_i|x_{i-1}) \cdot \sum_{j=1}^{\widetilde{N}_{i-1}(x_{i-1})} \widetilde{f}_{i-1,j}(x_{i-1}) \mathrm{N}\left(\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} \middle| \widetilde{k}_{i-1,j}(x_{i-1}), \widetilde{K}_{i-1,j}(x_{i-1}) \right), \quad (43)$$

where we use tilde to distinguish quantities in the approximation from the corresponding exact ones. Note that the approximate representation is on the same form as in (27), except that in the approximation the number of Gaussian terms may depend on the value of x_{i-1}. Of course, for $i = 2$ we use $\widetilde{V}_{i-1}(x_{i-1}, x_i, y_i, y_{i+1}) = V_{i-1}(x_{i-1}, x_i, y_i, y_{i+1})$. Starting with $\widetilde{V}_{i-1}(x_{i-1}, x_i, y_i, y_{i+1})$ we perform iteration $i-1$ in two steps. First we compute approximate representations of $U_i(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2})$ and $V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ using the same type of recursion formulas as in Theorem 1. More precisely, letting $U_i^\star(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2})$ and $V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ denote the approximations, we have

$$U_i^\star(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}) \propto p(x_{i+1}|x_i) \cdot \sum_{j=1}^{N_i^\star(x_i)} f_{ij}^\star(x_i) \mathrm{N}\left(\left[\begin{array}{c} y_i \\ y_{i+1} \\ y_{i+2} \end{array} \right] \middle| \, g_{ij}^\star(x_i) G_{ij}^\star(x_i) \right), \tag{44}$$

$$V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2}) \propto p(x_{i+1}|x_i) \cdot \sum_{j=1}^{N_i^\star(x_i)} f_{ij}^\star(x_i) \mathrm{N}\left(\left[\begin{array}{c} y_{i+1} \\ y_{i+2} \end{array} \right] \middle| \, k_{ij}^\star(x_i), K_{ij}^\star(x_i) \right), \tag{45}$$

where $N_i^\star(x_i) = L\widetilde{N}_{i-1}(x_i)$, and $G_{ij}^\star(x_i)$, $g_{ij}^\star(x_i)$, $f_{ij}^\star(x_i)$, $K_{ij}^\star(x_i)$ and $k_{ij}^\star(x_i)$ are given from expressions (32) to (35) and (42) after substituting $G_{i,(j-1)L+l}(x_i)$, $g_{i,(j-1)L+l}(x_i)$, $f_{i,(j-1)L+l}(x_i)$, $K_{i,(j-1)L+l}(x_i)$, $k_{i,(j-1)L+l}(x_i)$, $K_{i-1,j}(l)$, $k_{i-1,j}(l)$ and $f_{i-1,j}(l)$ by $G^\star_{i,(j-1)L+l}(x_i)$, $g^\star_{i,(j-1)L+l}(x_i)$, $f^\star_{i,(j-1)L+l}(x_i)$, $K^\star_{i,(j-1)L+l}(x_i)$, $k^\star_{i,(j-1)L+l}(x_i)$, $\widetilde{K}_{i-1,j}(l)$, $\widetilde{k}_{i-1,j}(l)$ and $\widetilde{f}_{i-1,j}(l)$, respectively. In the second step of the iteration we start from $V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ and generate a second (and final) approximation for $V_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$, denoted $\widetilde{V}_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$, by dropping the less important Gaussian terms in the first approximation (45) and thereafter renumbering the remaining terms. We use the same approximation procedure also for the final two iterations, $i = n-1$ and n. The expressions are again slightly modified for this iterations.

What terms in (45) that are of less importance is not obvious as they are functions of y_{i+1} and y_{i+2}, which are still unknown when this decision has to be taken. Natural strategies for handling this are either to maximise over or to integrate out y_{i+1} and y_{i+2} before comparing the size of the different terms. To maximise over y_{i+1} and y_{i+2} is obtained by evaluating the Gaussian density in (45) at its mean value. Thus, for a threshold value ε this strategy gives that we should drop terms in (45) that have

$$\frac{f_{ij}^\star(x_i) \mathrm{N}\left(\mu^\star(x_i) | \, k_{ij}^\star(x_i), K_{ij}^\star(x_i) \right)}{\max_{k=1,\ldots,N_i^\star(x_i)} \left\{ f_{ik}^\star(x_i) \mathrm{N}\left(\mu^\star(x_i) | \, k_{ik}^\star(x_i), K_{ik}^\star(x_i) \right) \right\}} < \varepsilon, \tag{46}$$

where $\mu^\star(x_i) = \left(K_{ij}^\star \right)^{-1} k_{ij}^\star(x_i)$. With the second strategy, to integrate out y_{i+1} and y_{i+2}, only $f_{ij}^\star(x_i)$ remains to compare. Thus, again for a given threshold ε, we drop all terms that corresponds to a $f_{ij}^\star(x_i)$ that have

$$\frac{f_{ij}^\star(x_i)}{\max_{k=1,\ldots,N_i^\star(x_i)} \left\{ f_{ik}^\star(x_i) \right\}} < \varepsilon. \tag{47}$$

In the simulation examples in Section 5 we adopt the first strategy, but we do not expect the second strategy to behave much differently. One should note that we decide what terms to drop separately for each possible value of x_i, and as a result the number of remaining terms, $\widetilde{N}_i(x_i)$, may differ for the different values of x_i.

Clearly, alternative term dropping strategies may be defined. First, one may use the term dropping step for $U_i^\star(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2})$ instead of for $V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2})$, but we do not expect this to make much difference. Second, instead of choosing a specific threshold value ε, one may fix the number of terms we want to keep in $\widetilde{V}_i(x_i, x_{i+1}, y_{i+1}, y_{i+2})$ and drop the necessary number of small terms in $V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2})$. Thereby the memory requirements for running the algorithm will be known in advance, but the quality of the approximation may be more variable than with the strategy we have chosen.

4.3 Backward simulation

When the (exact or approximate) forward integration has been done and the necessary quantities stored in memory, backward simulation is straight forward. Here we give the necessary equations for the approximate, computational feasible algorithm.

Sequentially for $i = n, \ldots, 1$ we first simulate x_i from

$$\pi^\star(x_i | x_{i+1}, \ldots, x_n, y_{i+1}, \ldots, y_n, z) \propto V_i^\star(x_i, x_{i+1}, y_{i+1}, y_{i+2}) \tag{48}$$

and then draw y_i from

$$\pi^\star(y_i | x_i, \ldots, x_n, y_{i+1}, \ldots, y_n) \propto U_i^\star(x_i, y_i, x_{i+1}, y_{i+1}, y_{i+2}). \tag{49}$$

The first is a discrete distribution and the second a mixture of r-variate Gaussian densities, so both are simple to generate realizations from. The resulting sample is thereby simulated from an approximation to the conditional distribution $\pi(x, y|z)$,

$$\pi^\star(x, y|z) = \prod_{i=1}^{n} \left[\pi^\star(x_i | x_{i+1}, \ldots, x_n, y_{i+1}, \ldots, y_n, z)\pi^\star(y_i | x_i, \ldots, x_n, y_{i+1}, \ldots, y_n)\right]. \tag{50}$$

One should also note that it is straight forward to evaluate $\pi^\star(x, y|z)$ for the generated sample (x, y), but to do this correctly one must of course remember to include the normalizing constants in the two conditional distributions $\pi^\star(x_i | x_{i+1}, \ldots, x_n, y_{i+1}, \ldots, y_n, z)$ and $\pi^\star(y_i | x_i, \ldots, x_n, y_{i+1}, \ldots, y_n)$.

4.4 Simulation from the hidden Markov model

To correct for the error introduced by the approximation discussed above one may use $\pi^\star(x, y|z)$ as a proposal distribution in a independent proposal Metropolis–Hastings algorithm. The resulting acceptance rate is also a natural measure for the quality of the approximation.

5 Simulation examples

We study the approximate forward-backward algorithm introduced above in a number of simulation exercises. We consider five sets of parameter values, which we denote by P1 to P5. Parameter set P1, which is our base case, are chosen to reflect realistic values for the seismic inversion application. P2 and P3 are identical to P1 except that we decrease and increase, respectively, the noise variance. The parameter values in P4 is the same as in P1 except for the rock physics model, the variances in y given x are larger in P4 than in P1. In all these first simulation studies we use a seismic profile of length $n = 100$, which is sufficiently small to allow us to use the approximate forward-backward algorithm for all of x and y. In a last simulation example we increase the resolution of the lattice and consider $n = 400$. For this we use parameters corresponding to P1, but where some parameter values are changed to compensate for the increased resolution. We denote the resulting parameter values by P5. With $n = 400$ we are no longer able to update the whole profile, but must adopt the block Metropolis–Hastings algorithm discussed in Remark 2 in Section 3.1 to obtain satisfactory convergence and mixing results.

In the following we first, in Section 5.1, define the precise parameter values used for P1 to P5. Thereafter, in Section 5.2, we quantify the noise level in the five models by defining and reporting values for the signal-to-noise-ratios. In Section 5.3 we present simulation results and use these to discuss the quality of the approximate algorithms and convergence and mixing properties of the corresponding Metropolis–Hastings algorithms. We consider both simulation of x and y given z and the seismic inversion problem of sampling x, y and z given the seismic data d. In Section 5.4 we present results for the seismic inversion problem for parameter sets P1 to P4. Finally, in Section 5.5 we give results for a small sensitivity study, using a different parameter set for inversion than what is used to generate the data.

5.1 Parameter values

Our base case parameter values, P1, are chosen to be realistic for the seismic inversion application and is based on the values used in Larsen et al. (2006). We consider a model with $L = 4$ possible classes for x_i, where $x_i = 1, 2$ and 3 represent gas-, oil- and brine (water) saturated sandstone, respectively, and $x_i = 4$ represents shale. Sandstone is porous and allows flow of gas, oil, and water, whereas the porosity in shale is negligible and therefore acts as a barrier to fluid flow. Our choice of transition matrix for x, P, is based on the values used in Larsen et al. (2006), but we consider a coarser seismic resolution than done there. Numbering the nodes from the bottom of the trace, we use

$$\mathrm{P} = \begin{bmatrix} 0.9441 & 0 & 0 & 0.0559 \\ 0.0431 & 0.9146 & 0 & 0.0424 \\ 0.0063 & 0.0230 & 0.9422 & 0.0284 \\ 0.0201 & 0.0202 & 0.1006 & 0.8591 \end{bmatrix}. \tag{51}$$

This is obtained by first cubing the P used in Larsen et al. (2006) and thereafter moving the resulting values at the three positions with zeros above to the diagonal. The three zero elements are important in the seismic application as these represent known physical properties. As water has a higher density than oil, which again has a higher density than gas, water can not be above gas or oil and oil can not be above gas, unless separated by a non-porous shale layer. The resulting marginal probabilities for x_i are $[0.24, 0.16, 0.38, 0.22]$.

As discussed in Section 3 we consider a situation where $y_i \in \mathbb{R}^3$, where the three elements represent the logarithms of P-wave and S-wave velocities and density, respectively. In Larsen et al. (2006) the distribution of $y_i|x_i$ is represented as empirical distributions given by a set of corresponding x_i and y_i values. We use the same set of (x_i, y_i) values to estimate mean vectors and covariance matrices for the four assumed Gaussian distributions. The resulting mean vectors are

$$\mu(1) = \begin{bmatrix} 8.052 \\ 7.492 \\ 7.688 \end{bmatrix}, \quad \mu(2) = \begin{bmatrix} 8.071 \\ 7.472 \\ 7.730 \end{bmatrix}, \quad \mu(3) = \begin{bmatrix} 8.121 \\ 7.467 \\ 7.746 \end{bmatrix}, \quad \mu(4) = \begin{bmatrix} 8.166 \\ 7.546 \\ 7.846 \end{bmatrix} \tag{52}$$

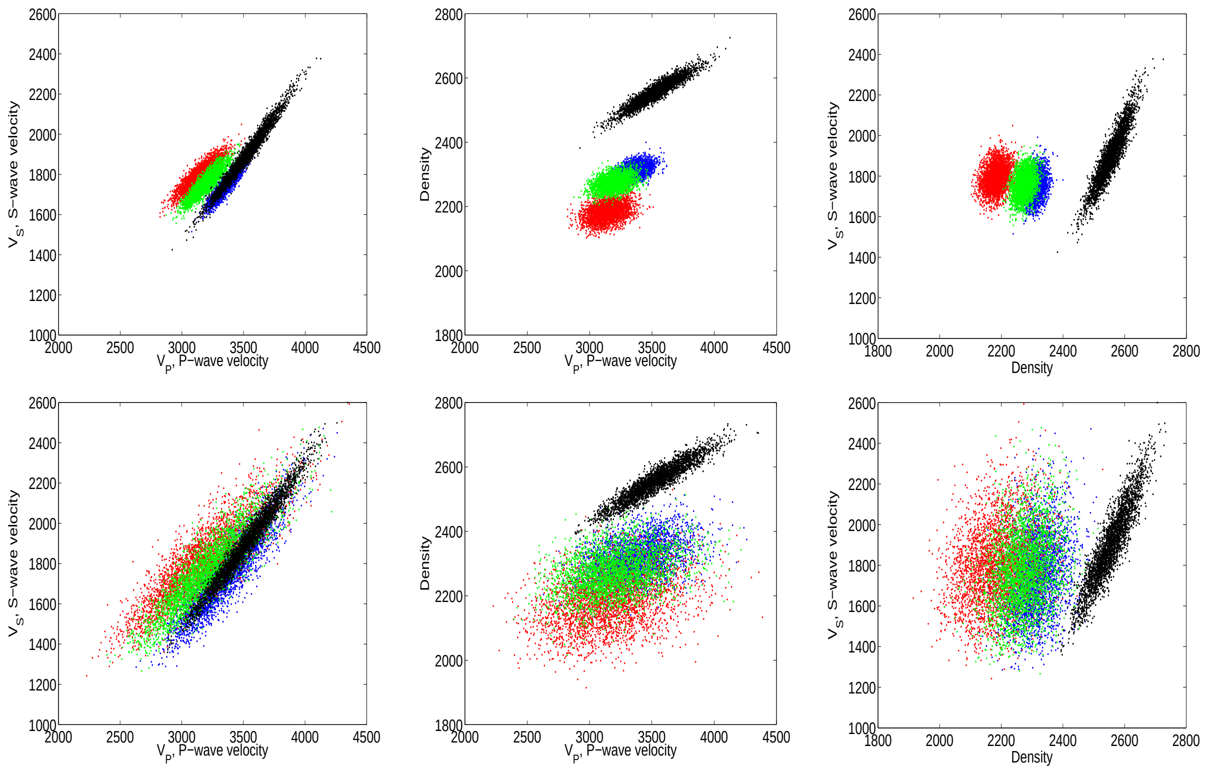


Figure 2: Scatter plots of P- and S-wave velocities and density of samples from the distribution adopted for $y_i|x_i$ in P1, P2, P3 and P5 (upper row) and P4 (lower row). Red, green, blue and black is used for gas-, oil- and brine-saturated sandstone and shale, respectively.

and the covariance matrices are given by the following figures, where the diagonal and off-diagonal entries give standard deviations and correlations, respectively, for each of the four possible values of x_i,

$$\begin{bmatrix} 0.031 & 0.876 & 0.322 \\ 0.876 & 0.033 & 0.271 \\ 0.322 & 0.271 & 0.012 \end{bmatrix}, \quad \begin{bmatrix} 0.027 & 0.891 & 0.384 \\ 0.891 & 0.032 & 0.295 \\ 0.384 & 0.295 & 0.009 \end{bmatrix} \tag{53}$$

$$\begin{bmatrix} 0.022 & 0.912 & 0.453 \\ 0.912 & 0.032 & 0.317 \\ 0.453 & 0.315 & 0.008 \end{bmatrix}, \quad \begin{bmatrix} 0.044 & 0.982 & 0.935 \\ 0.982 & 0.068 & 0.917 \\ 0.935 & 0.917 & 0.015 \end{bmatrix}. \tag{54}$$

The upper row in Figure 2 shows scatterplots of simulated P- and S-wave velocities and density according to the specified distributions. Here and in all the following figures we use the colours red, green, blue and black for gas-, oil- and brine-saturated sandstone and shale, respectively. We observe that shale is well separated from the other three classes and that gas-saturated sandstone is reasonably well separated from oil- and brine-saturated sandstone, whereas there is large overlap between the densities of oil- and brine-saturated sandstone.

To specify the model of d given y it remains to specify the number of and what angles that are used, the wavelet for each angle, and variances for the two error terms ε^1_{ij} and ε^2_{ij}. Analogously to Larsen et al. (2006) we use the $s = 5$ angles $\theta = 0^\circ, 10^\circ, 20^\circ, 30^\circ$ and 40°. We also use the same angle independent Ricker wavelet, except that we scale it to compensate for coarser resolution. We use

$$w(u,\theta) = (1 - 2(\pi\,\phi\,u)^2)\exp\left(-(\pi\,\phi\,u)^2\right),\; u = -k,\ldots,k, \tag{55}$$

where $\phi = 0.11$ and $k = 10$. Also for the error variances we use corresponding values as in Larsen et al. (2006), except for an appropriate scaling to compensate for the difference in seismic resolution. We use $\mathrm{Var}(\varepsilon^1_{ij}) = 0.015^2$ and $\mathrm{Var}(\varepsilon^2_{ij}) = \mathrm{Var}(\varepsilon^1_{ij})/10^4$.

The above completely defines the parameter values in P1. P2 to P4 are small modifications of P1. In P2 and P3 the variances for the noise terms ε^1_{ij} and ε^2_{ij} are scaled to give higher and lower signal-to-noise ratios, respectively. We use $\mathrm{Var}(\varepsilon^1_{ij})$ equal to 0.0085^2 and 0.026^2 in P2 and P3, respectively, still with $\mathrm{Var}(\varepsilon^2_{ij}) = \mathrm{Var}(\varepsilon^1_{ij})/10^4$. The parameter values in P4 are identical to P1 except for a scaling of the covariance matrices in the Gaussian densities for $y_i|x_i$ so that the four classes are less separated. For classes 1 to 3, gas-, oil- and brine-saturated sandstone, the covariance matrices used in P1 are multiplied with 10, whereas for class four, shale, the covariance matrix is multiplied with 2. These scaling values are also used in Larsen et al. (2006). Scatter plots of simulated P-wave and S-wave velocities and density are shown in the lower row of Figure 2.

Parameter set P5 corresponds to P1, but some of the parameter values are changed to compensate for an increased lattice resolution. Thus, P5 corresponds to the base case P1 on a finer lattice. The transition matrix P for P5 is the same as in Larsen et al. (2006)

$$\mathrm{P} = \begin{bmatrix} 0.980 & 0 & 0 & 0.020 \\ 0.015 & 0.970 & 0 & 0.015 \\ 0.002 & 0.008 & 0.980 & 0.010 \\ 0.007 & 0.007 & 0.036 & 0.950 \end{bmatrix} \tag{56}$$

with resulting marginal probabilities for x_i equal to $[0.23, 0.16, 0.39, 0.22]$. The distribution for y_i given x_i is the same as for P1. The wavelet has the same parametric form as for P1, but the parameter values are changed to compensate for the difference in resolution. We use (55) with $\phi = 0.03$ and $k = 30$, which is identical to the choice in Larsen et al. (2006). To compensate for the change in resolution we also need to scale the error variances to get approximately the same noise level as in P1. We set $\mathrm{Var}(\varepsilon^1_{ij}) = 0.015^2/3$ and again $\mathrm{Var}(\varepsilon^2_{ij}) = \mathrm{Var}(\varepsilon^1_{ij})/10^4$.

5.2 Signal-to-noise-ratio

It is common to quantify the amount of noise in an inversion problem by a signal-to-noise (SN) ratio. In the setting we consider here it is not obvious how to split d into signal and noise parts. The x and y represent physical quantities, and so it may be reasonable to classify variability in d that has its origin in x or y as the signal part. However, our main interest is to regain x from d, and this makes it natural to include in the signal only the variability in d that originates from x. We define one signal-to-noise ratio for each of these two views, starting with the first.

Let $\widetilde{d}$ denote the data in the absence of observation noise so that (13) becomes

$$\widetilde{d}_{ij} = \sum_{u=-k}^{k} w(u,\theta_j)\cdot r_{i-u,j}. \tag{57}$$

Parameter set	P1	P2	P3	P4	P5
SN	2.82	8.35	0.92	6.15	3.10
SN*	1.33	2.47	0.64	0.45	1.10

Table 1: Signal-to-noise ratios for the various parameter sets.

It is natural to define the strength of the signal by $\mathrm{Var}(\widetilde{d}_{ij})$. However, this quantity is a function of angle j and, due to border effects, node number i. To get one number we average over i and j,

$$\mathrm{S} = \frac{1}{ns}\sum_{i=1}^{n}\sum_{j=1}^{s}\mathrm{Var}(\widetilde{d}_{ij}). \tag{58}$$

This can not be calculated analytically, but is easily estimated via simulation. The strength of the noise is correspondingly defined by

$$\mathrm{N} = \frac{1}{ns}\sum_{i=1}^{n}\sum_{j=1}^{s}\mathrm{Var}(\varepsilon_{ij}). \tag{59}$$

The resulting signal-to-noise ratio we denote by $\mathrm{SN} = \mathrm{S}/\mathrm{N}$.

Let $\mathrm{S}^\star$ and $\mathrm{N}^\star$ denote the strength of the signal and noise, respectively, when only the variation that results from x is considered as signal. $\mathrm{S}^\star$ is then again given by (58) if $\widetilde{d}_{ij}$ is replaced by

$$d^\star_{ij} = \sum_{u=-k}^{k} w(u,\theta_j)\cdot r^\star_{i-u,j}, \tag{60}$$

where $r^\star_{ij}$ is given from (11) by substituting y_{i-1} and y_{i+1} by their mean values,

$$r^\star_i = (r^\star_{i1},\ldots,r^\star_{is})^T = \Gamma^T\cdot\frac{\mu(x_{i+1})-\mu(x_{i-1})}{2}. \tag{61}$$

$\mathrm{N}^\star$ can then be expressed as

$$\mathrm{N}^\star = \frac{1}{ns}\sum_{i=1}^{n}\sum_{j=1}^{s}\mathrm{Var}(d_{ij}-d^\star_{ij}) \tag{62}$$

and the corresponding signal-to-noise ratio is $\mathrm{SN}^\star = \mathrm{S}^\star/\mathrm{N}^\star$. Again the quantities that are not analytically available can easily be found via simulation.

The resulting signal-to-noise ratios for parameter sets P1 to P5 are given in Table 1. We see that P4 has higher SN than P1, even if retrieving x from d is clearly harder in P4 than in P1. We find $\mathrm{SN}^\star$ to be a better measure for the difficulty of the inverse problem, but signal-to-noise ratio definitions similar to SN are most common in the seismic inversion literature, probably because the objective often is inversion to y only.

5.3 Evaluation of approximate forward-backward algorithm

In this section we evaluate the performance of the proposed approximate forward-backward algorithm in a simulation exercise for each of the parameter sets P1 to P5. The length of the

Parameter set	P1	P2	P3	P4	P5
ε	10^{-4}	10^{-4}	10^{-3}	10^{-3}	10^{-4}
acceptance rate	0.83	0.91	0.55	0.51	0.63

Table 2: Threshold values ε used in (46) and resulting Metropolis–Hastings acceptance rates for each of the parameter sets P1 to P5.

lattice is $n = 100$ for P1 to P4 and $n = 400$ for P5. In each case we first simulate x, y, z and d according to the model specified in Sections 2 and 3 and thereafter use the proposed algorithms to simulate from the resulting posterior distributions. We consider both simulating x and y given z by the algorithm discussed in Section 4.4 and the seismic inversion problem of simulating x, y and z given d by the algorithm defined in Section 3.1. We evaluate the quality of the approximate algorithm by the acceptance probability and the convergence and mixing properties of the Metropolis–Hastings algorithms.

The approximate forward-backward algorithm, and thereby the corresponding Metropolis–Hastings algorithms, contains one tuning parameter, ε in (46). The quality of the approximation and effectiveness of the algorithm clearly depends on this value. If ε is too small the number of terms that needs to be computed and stored becomes large and the algorithm becomes infeasible. If ε is too large the quality of the approximation deteriorates with very low Metropolis–Hastings acceptance probabilities as a result. For each of the parameter sets we found a reasonable value for ε by trial and error, the values are reported in Table 2. As expected ε must be higher when the signal-to-noise-ratios $\text{SN}^\star$ is low, and for the same value of ε the acceptance rate decreases when $\text{SN}^\star$ decreases. The acceptance rates for P3 and P4 can easily be increased by using a smaller ε, but then the number of Gaussian terms to store quickly becomes too large for the memory of typical computers. Our experience is that it is mainly the available memory that limits the possible value of ε and thereby the quality of the approximation, not the computation time. The acceptance rates reported in Table 2 is when simulating x, y and z given d, but simulating x and y given z produced essentially the same rates.

Figures 3 to 6 present the simulation results for parameter sets P1 to P4, respectively. To upper rows show the simulated "true" values. Note that we use the same realisation of x in all four cases to make comparison easier. The middle and lower rows contain posterior simulation results when conditioning on z and d, respectively. The two lower rows consist of three parts. To the left the "true" x is replotted for easier comparison, in the middle each state of the Metropolis–Hastings algorithm is plotted side by side, and the plots to the right show resulting marginal probabilities for each x_i. The Markov runs shown in Figures 3 to 6 are all initiated by setting all $x_i = 1$ and drawing y (and z when conditioning on d) values from the corresponding full conditional. In all the runs the initial state is left within very few iterations, so the burn-in phases of the Markov chains are not visible in the figures. We have also tried starting with all $x_i = 4$, and other initial values, without experiencing burn-in problems. The results clearly show that the approximate forward-backward algorithm gives a very good approximation to the distribution of interest and produces thereby very good mixing properties when used as a proposal distribution in a Metropolis–Hastings setting.

Figure 7 presents simulation results for parameter set P5. As mentioned above, this is for $n = 400$ and then the memory requirements for the approximate algorithm made it impossible to update all of x and y jointly. In stead we adopted the block Metropolis–Hastings algorithm

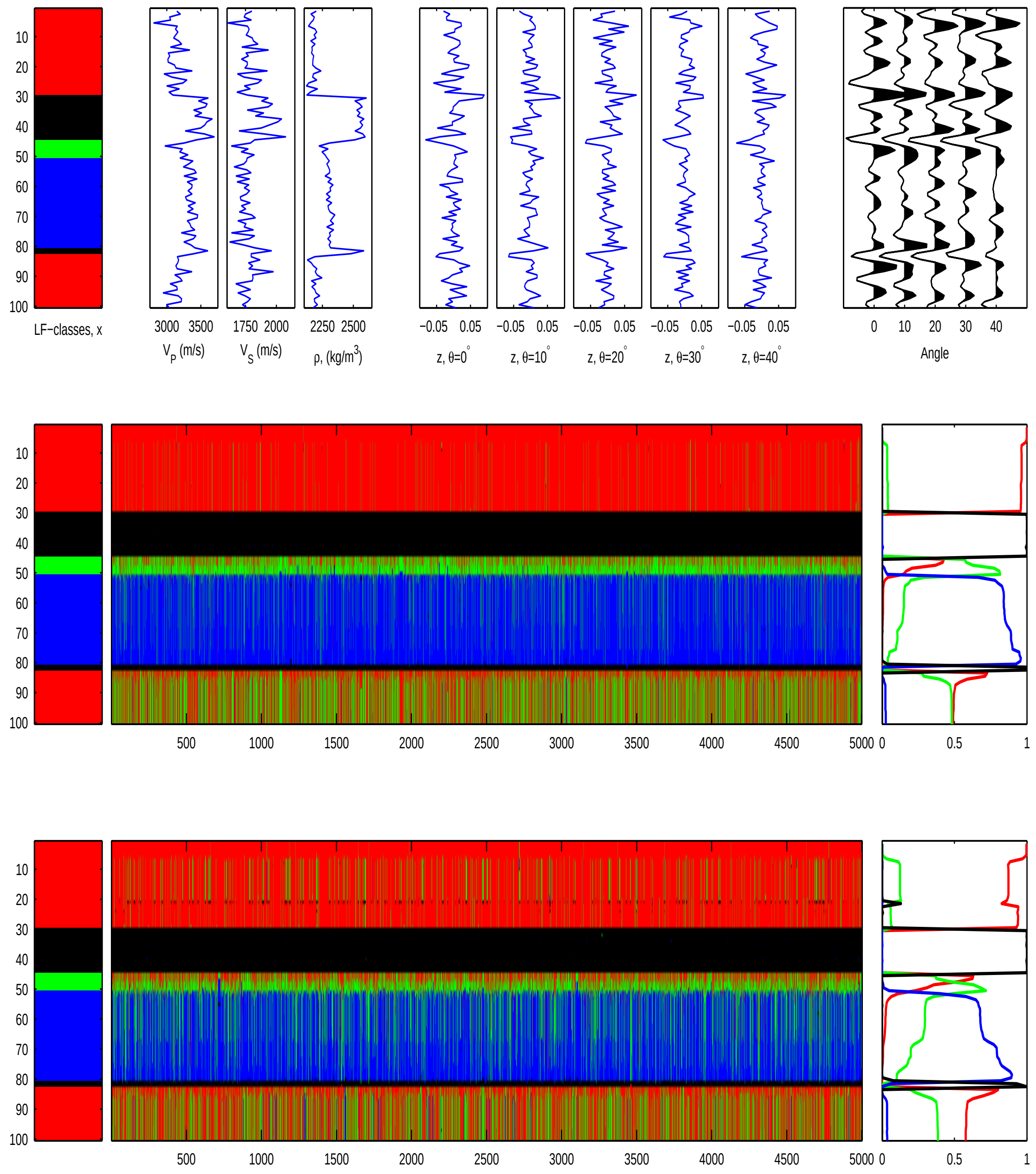


Figure 3: Simulation results for parameter set P1: The upper row gives, from left to right, the simulated “true” x, the simulated “true” elastic parameters, z and d. The middle and lower rows contain posterior simulation results when condition on z and d, respectively. From left to right, the two rows give the true x (for easier comparison), each state of the Metropolis–Hastings algorithm plotted side by side, and resulting marginal posterior probabilities for each x_i. Red, green, blue and black represent gas-, oil- and brine-saturated sandstone and shale, respectively.

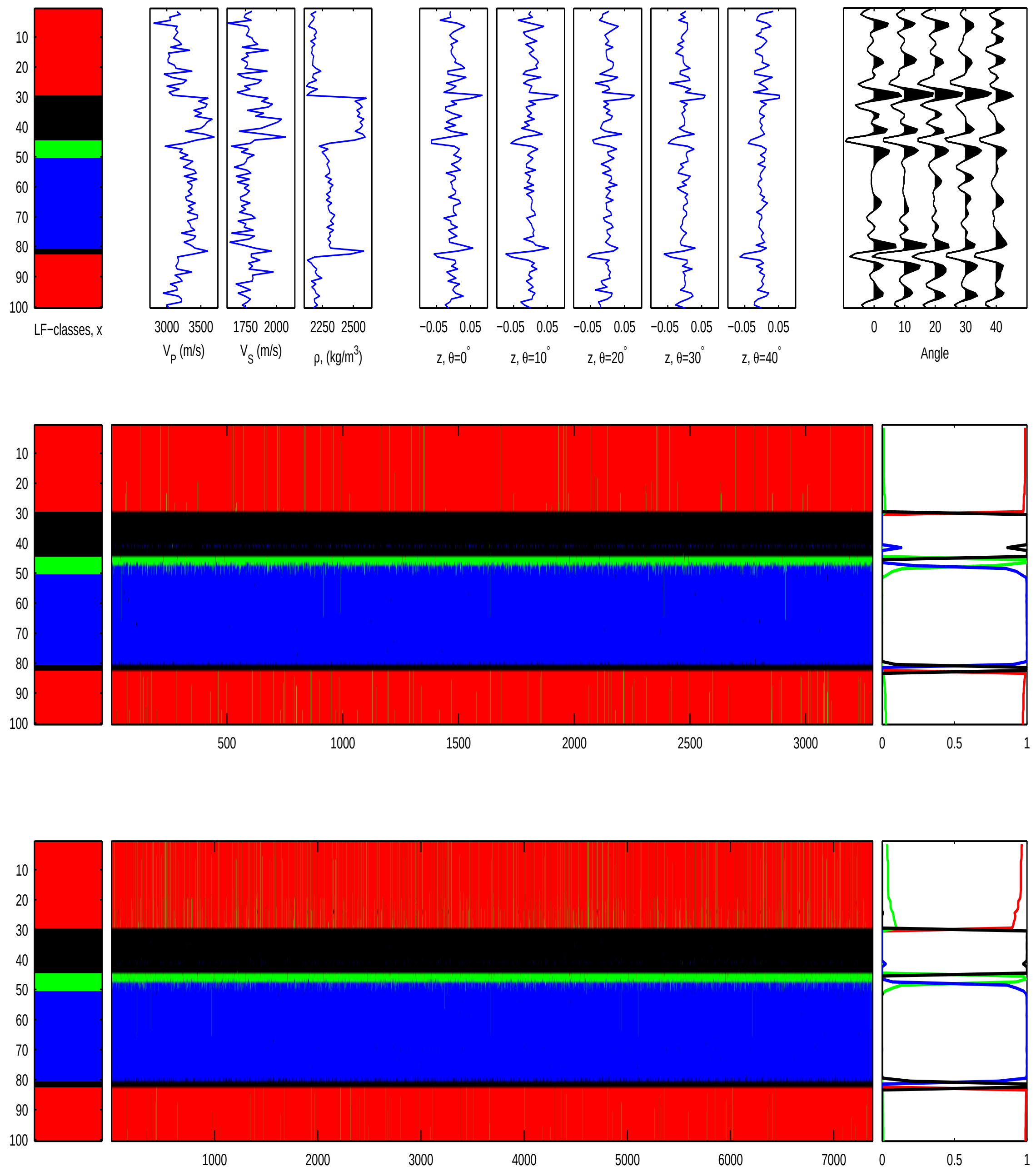


Figure 4: Simulation results for parameter set P2: See Figure 3 for an explanation of the different parts of the figure.

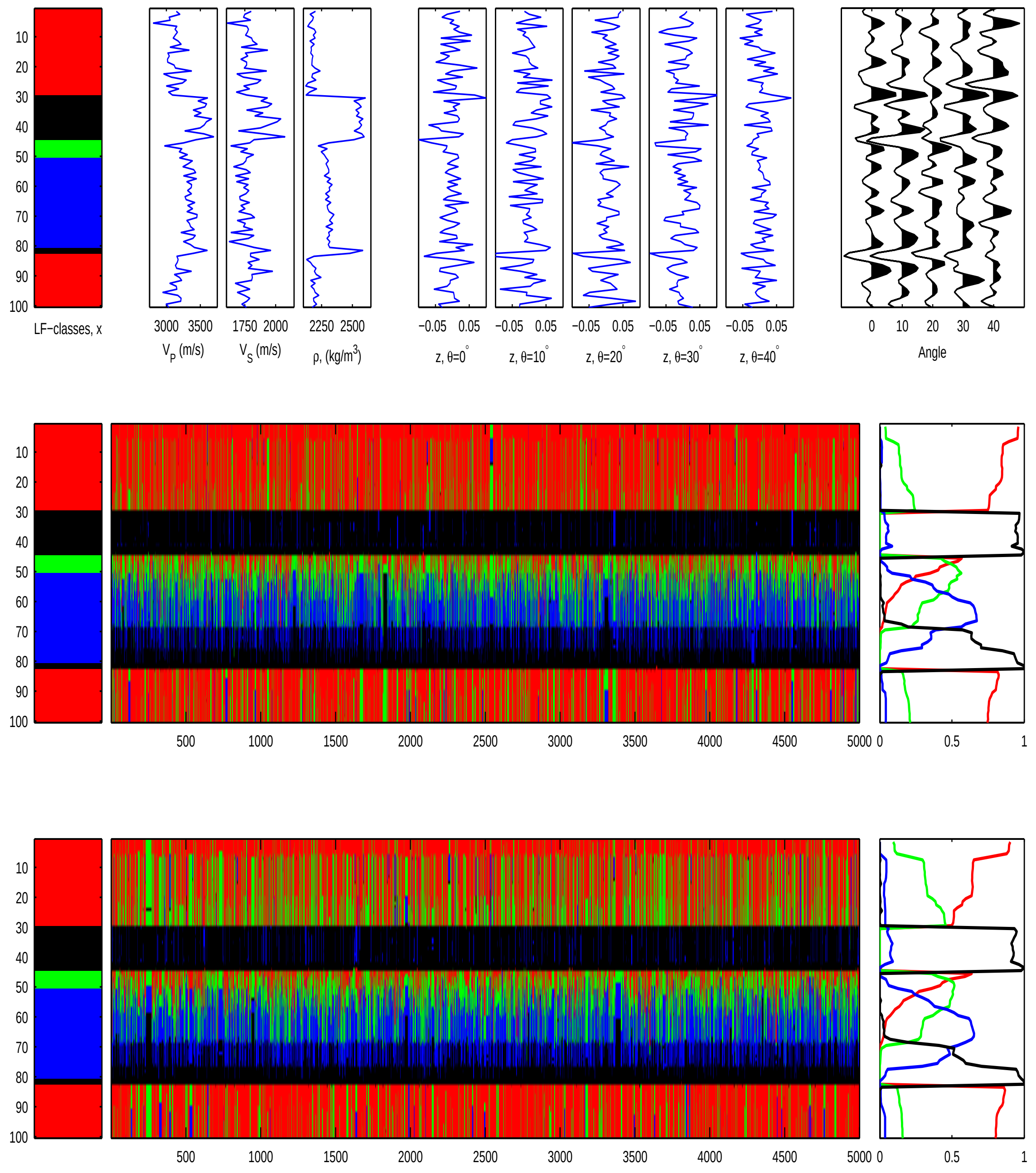


Figure 5: Simulation results for parameter set P3: See Figure 3 for an explanation of the different parts of the figure.

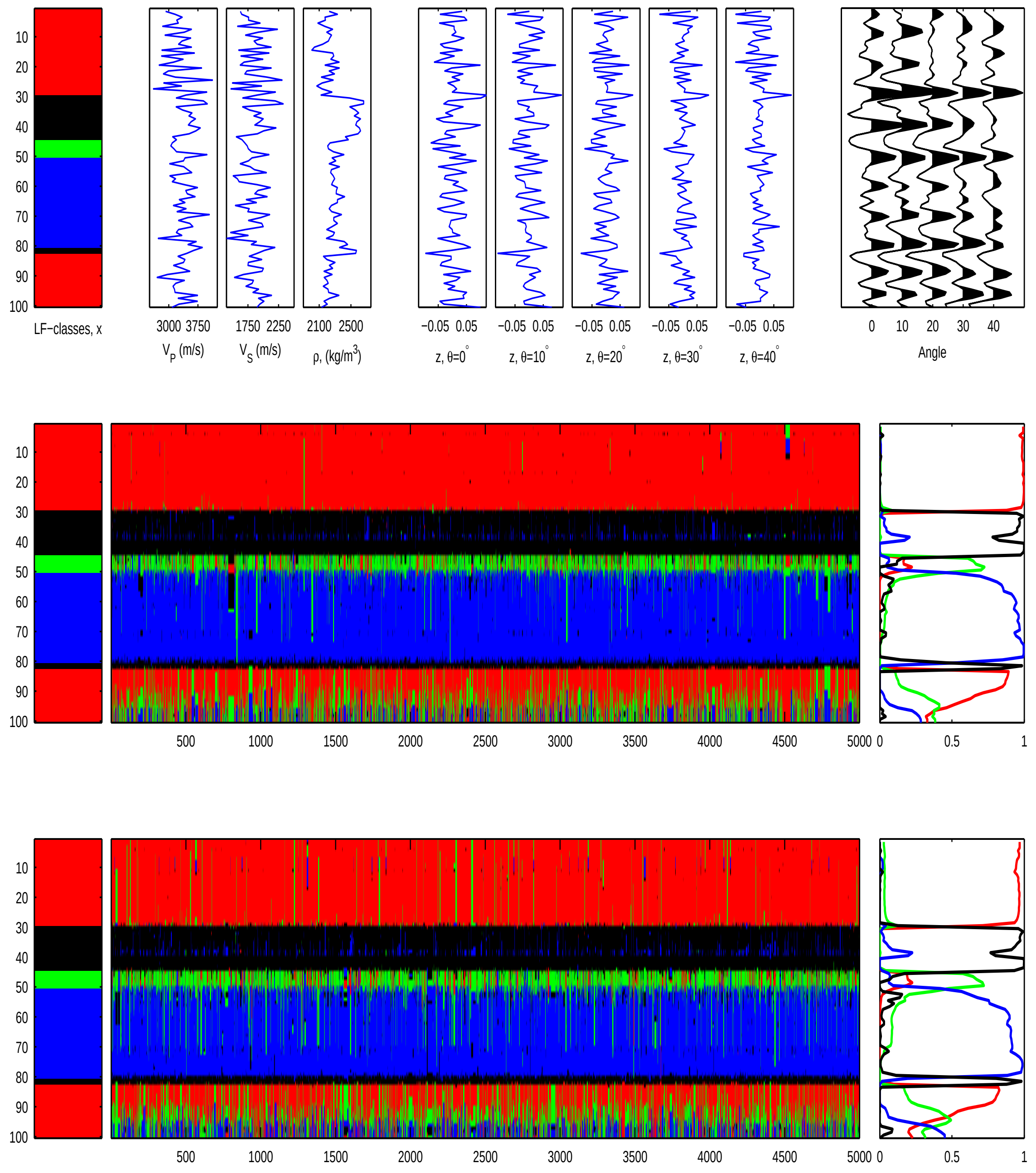
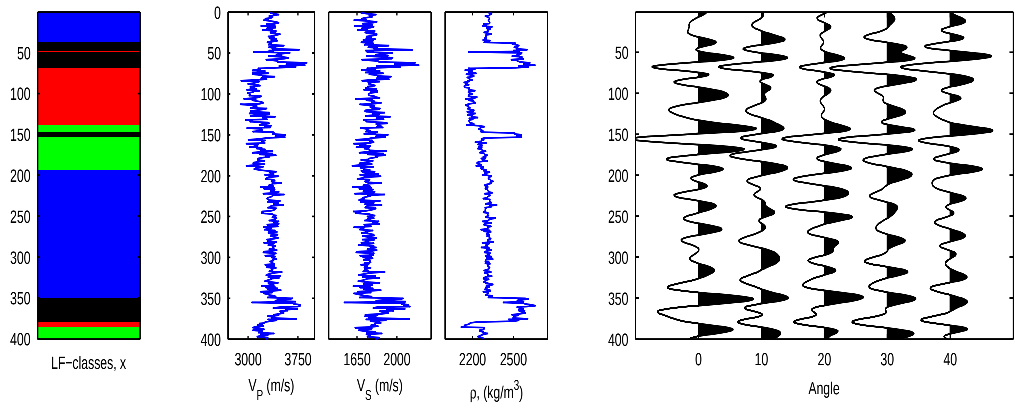
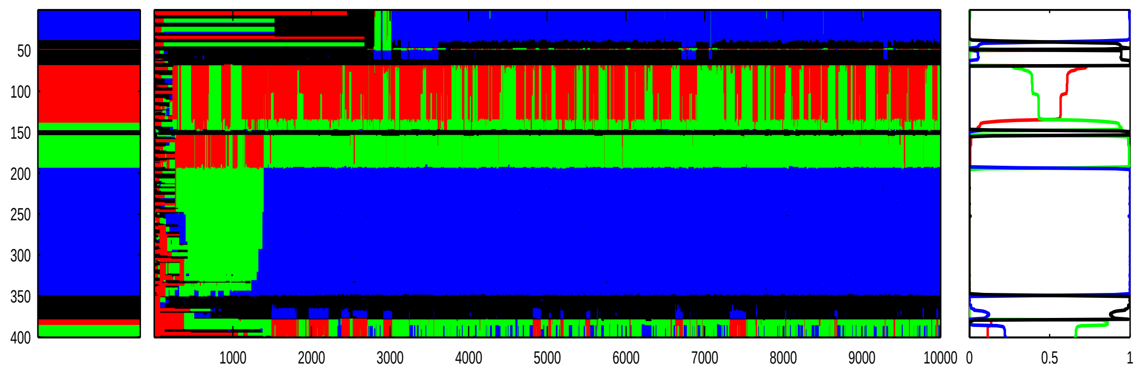


Figure 6: Simulation results for parameter set P4: See Figure 3 for an explanation of the different parts of the figure.





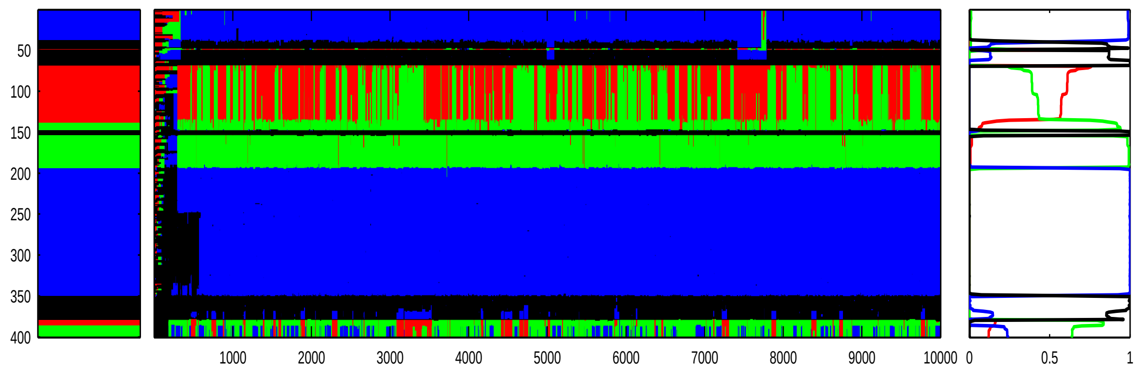


Figure 7: Simulation results for parameter set P5: The upper row gives, from left to right, the simulated "true" x, the simulated "true" elastic parameters and d. The two lower rows contain posterior simulation results when condition on d. The middle and lower rows show runs initiated with all $x_i = 1$ and $x_i = 4$, respectively. From left to right, the two rows give the true x (for easier comparison), each state of the Metropolis–Hastings algorithm plotted side by side, and resulting marginal posterior probabilities for each x_i. Red, green, blue and black represent gas-, oil- and brine-saturated sandstone and shale, respectively.

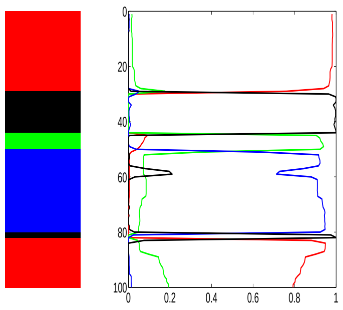
discussed in Remark 2. We use blocks of 100 nodes and in each iteration randomly choose one of the intervals $[1, 100], [51, 150], \ldots, [301, 400]$ to update. Again we run with the two initial states mentioned above. Figure 7 corresponds to Figures 3 to 6, except that the two lower rows both show simulation results when condition in d, one for each of the two initial states mentioned. We see that the convergence is much slower in this case, we can easily see the burn-in phase in the plots. The mixing of the chain is also somewhat less favourable in this case.

5.4 Inversion results

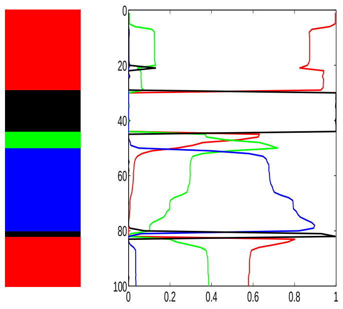
In this section we summarise the inversion results for the simulation exercises introduced above. We limit the attention to the runs where we condition on the simulated seismic data d. Comparing the marginal posterior probabilities in the bottom right corner in Figures 3 to 7 with the corresponding "true" x-values we see how the inversion results, as expected, depend on the signal-to-noise ratio $\text{SN}^\star$. For each of the parameter sets P1 to P4 we repeat the analysis for ten different realisations of y, z and d, still keeping the same realisation for x to allow easier comparison. Figure 8 shows the resulting marginal posterior probabilities for three cases with parameter set P1. Using the maximum marginal posterior probability classifier this figure also gives the resulting confusion matrices. As one should expect, the prediction ability varies somewhat between the different simulated data sets. Figure 9 shows corresponding results for each of P1 to P4 averaged over the ten runs. Again we see how the prediction ability deteriorates as $\text{SN}^\star$ becomes lower. Still, we find the results quite impressive given that the data is as shown in the upper right corners of Figures 3 to 6. It is close to impossible to identify the true x-values by just looking at the data. Studying the confusion matrices more in detail we see shale (black) are best identified, that gas-saturated sandstone (red) are mostly confused with oil-saturated sandstone (green), and oil-saturated sandstone (green) mostly confused with brine-saturated sandstone (blue). This is all as one should expect from Figure 2, the cloud of shale is well separated from the others and the clouds of gas- and oil-saturated sandstone are closer then the clouds of gas- and brine-saturated sandstone.

5.5 A small sensitivity study

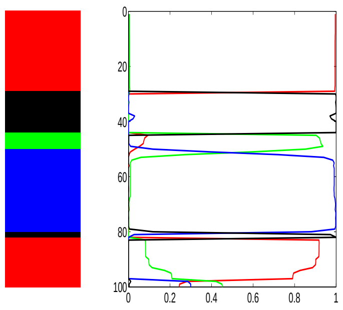
Clearly the inversion results above are better than what one can expect for real seismic data. In the simulation exercises we know, and use, the correct model and parameter values that generated the data. It is beyond the scope of this paper to do inversion of real seismic data, our focus is the algorithms for handling the hidden Markov chain with two hidden layers. However, as a first small test of the consequences of using a wrong model in the inversion we have done a simulation exercise where the data is generated according to our forward model with parameter set P1, whereas parameter set P4 is used in the inversion. Our interest is both how this influences the inversion results and what the consequences are for the convergence and mixing of the Metropolis–Hastings algorithm. After a little try and error, as discussed above, we settled on a value $\varepsilon = 2 \cdot 10^{-3}$ for the algorithm tuning parameter, which resulted in an acceptance rate of 0.49. Again we made two Markov chain runs, one initiated with all $x_i = 1$ and one with with all $x_i = 4$. Convergence seems to occur after very few iterations and the mixing is good, see one of the simulated chains in Figure 10. Corresponding to Figure 9, Figure 11 gives inversion results averaged over ten realisations of y, z and d. The shale (black) and brine-saturated sandstone are still well identified, whereas this is no longer true for gas-



	gas	oil	brine	shale
gas	0.9271	0.0675	0.0034	0.0020
oil	0.0670	0.9253	0.0072	0.0006
brine	0.0030	0.0775	0.8997	0.0198
shale	0	0.0001	0.0070	0.9929



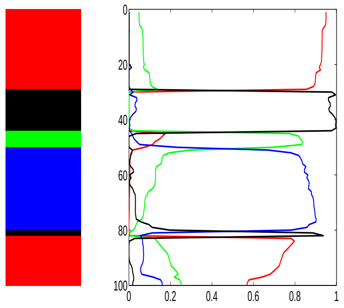
	gas	oil	brine	shale
gas	0.7985	0.1862	0.0121	0.0031
oil	0.4485	0.5394	0.0120	0.0002
brine	0.0200	0.2450	0.7320	0.0030
shale	0.0002	0.0007	0.0042	0.9950



	gas	oil	brine	shale
gas	0.9117	0.0690	0.0188	0.0004
oil	0.0666	0.9108	0.0225	0.0002
brine	0.0004	0.0364	0.9585	0.0047
shale	0	0.0000	0.0050	0.9950

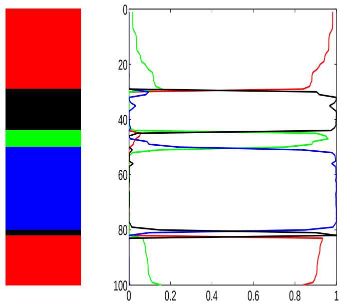
Figure 8: For different simulated data sets d in each row, the figure shows, from left to right, the “true” x, resulting marginal probabilities and confusion matrix using the maximum marginal probability classifier. Red, green, blue and black represent gas-, oil- and brine-saturated sandstone and shale, respectively.

P1:



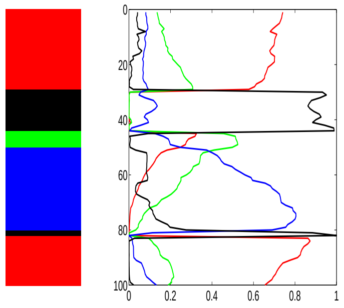
	gas	oil	brine	shale
gas	0.8413	0.1214	0.0328	0.0045
oil	0.1329	0.7969	0.0638	0.0063
brine	0.0037	0.1077	0.8568	0.0318
shale	0.0002	0.0020	0.0260	0.9718

P2:



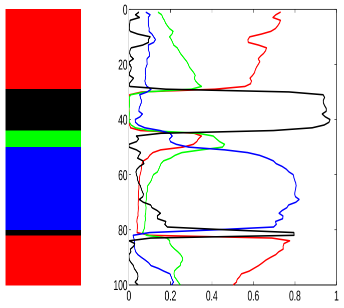
	gas	oil	brine	shale
gas	0.9267	0.0729	0.0003	0.0000
oil	0.0324	0.8889	0.0719	0.0069
brine	0	0.0064	0.9852	0.0083
shale	0	0.0017	0.0212	0.9772

P3:



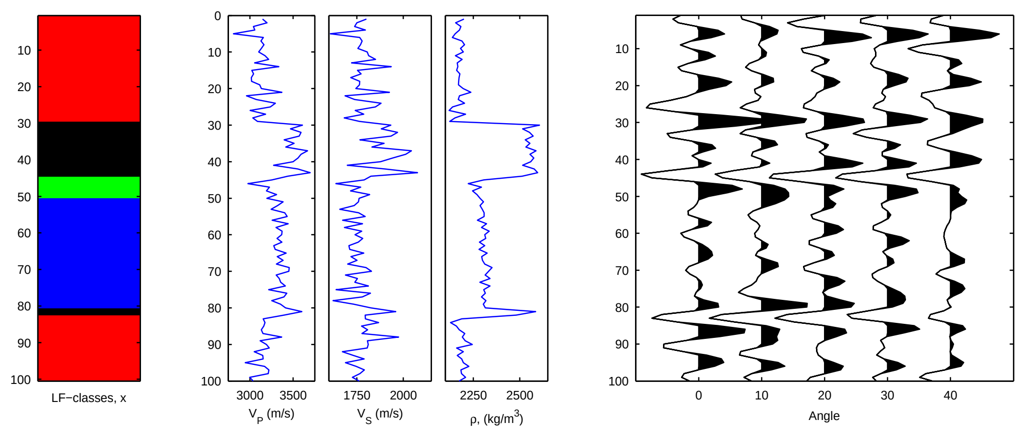
	gas	oil	brine	shale
gas	0.7169	0.1936	0.0690	0.0205
oil	0.2879	0.5049	0.1873	0.0199
brine	0.0568	0.2049	0.6411	0.0973
shale	0.0010	0.0025	0.0782	0.9183

P4:



	gas	oil	brine	shale
gas	0.6354	0.2382	0.0982	0.0282
oil	0.3162	0.4182	0.2198	0.0458
brine	0.0575	0.1203	0.7278	0.0943
shale	0.0162	0.0234	0.0611	0.8992

Figure 9: Inversion results for each of the parameter sets P1 to P4: ‘ From left to right, the figures shows the “true” x-values, marginal posterior probabilities and confusion matrix, averaged over ten realisations of y, z and d. Red, green, blue and black represent gas-, oil- and brine-saturated sandstone and shale, respectively.



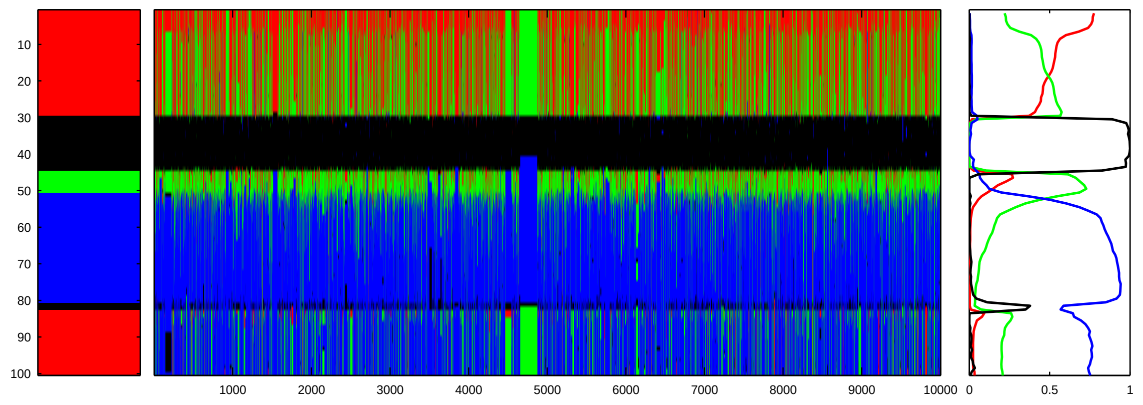
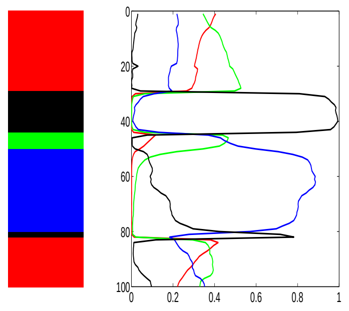


Figure 10: Sensitivity study: In the first row from left to right, the truth x, the true elastic parameters and the synthetic seismic data d. In the second row from left to right, the truth x, all the samples of x from a simulation x, y and z given d, with initial state $x = [0, \dots, 0]^T$ and the resulting marginal probabilities from the simulation.



	gas	oil	brine	shale
gas	0.3216	0.4169	0.2378	0.0237
oil	0.0793	0.4304	0.4722	0.0181
brine	0.0017	0.0317	0.8120	0.1546
shale	0.0030	0.0109	0.0584	0.9277

Figure 11: Inversion results for the sensitivity study. We have the average result for all the ten inversions. Red, green, blue and black represent gas-, oil-, brine-saturated sandstone and shale, respectively.

and oil-saturated sandstone.

6 Closing remarks

We have revisited the seismic inversion problem as a hidden Markov model with both continuous and discrete hidden variables. We split the model into a switching linear Gaussian model and a Gaussian linear model. To handle the first part computationally we propose an approximate forward-backward algorithm. In a number of simulation exercises we demonstrate the effectiveness of the approximation and how this makes inversion of the seismic model computationally feasible. The approximate algorithm includes a tuning parameter ε. To choose a value for ε one must compromise between memory usage and computation time on one side and approximation accuracy on the other. We have found no automatic way to set the value of ε, but our experience is that it is relatively easy to find a reasonable value by try and error. What makes the choice of ε non-trivial is that it is used to decide what terms to drop in the forward recursions when information from the data is available from one side only. The correct importance of the various terms becomes available first after the following backward recursions.

We think the inversion problem in the switching linear Gaussian model for seismic inversion is harder than the problems previously considered for switching linear dynamical systems (Zoeter and Heskes, 2006; Bar-Shalom and Li, 1998) and switching state space models (Barber, 2006). Within an interval with the same value for x, the seismic data does not depend on the mean value of the continuous variable. By the difference taken in (11) the mean value of the continuous variable influences the data only when the value of x is changing. Otherwise the data only depends on x via the covariance matrix of the continuous variable. This induces larger posterior uncertainty in x and it becomes correspondingly more important to have an approximate forward-backward algorithm that realistically represents this uncertainty. Thus, we think the importance of including more Gaussian terms in the forward recursion is larger for the seismic model than for the cases previously considered in Zoeter and Heskes (2006), Bar-Shalom and Li (1998) and Barber (2006).

We define an approximate forward recursion by dropping Gaussian terms with small weights. In the references mentioned above an approximation is obtained by taking a single Gaussian density that (approximately) represents the whole Gaussian mixture. It is clearly also possible to define an approximate forward recursion by following an intermediate strategy, finding groups of terms in the Gaussian mixture that have similar mean and covariance and approximate these by a single Gaussian term. However, the computational cost of finding what terms to merge is quadratic in the number of terms, whereas the cost of finding what terms to drop is linear in the number of terms. Thus, unless the number of Gaussian terms necessary to obtain a sufficient good approximation is dramatically reduced when using the merging strategy, our simple dropping strategy is preferable. We have done a little experimentation with the merging strategy for our seismic inversion model, but without any success. However, we think the merging strategy may have a potential if the continuous variable y is univariate.

The focus of this paper is the computational problem associated with the hidden Markov seismic model. We have not considered inversion of real seismic data. To answer a real inversion problem one must also solve the associated parameter estimation problem. Preliminary experimentation with maximum likelihood estimation from simulated data indicates that it is not possible to estimate all the model parameters from available seismic data. Either one must adopt a Bayesian view with informative priors, or information about (at least some of) the parameters must be obtained from other sources.

References

Aki, K. and Richards, P. G. (1980). *Quantitative seismology: Theory and methods*, W. H. Freeman and Company.

Bar-Shalom, Y. and Li, X.-R. (1998). *Estimation and Tracking: Principles, Techniques and Software*, Artech House, Norwood, MA.

Barber, D. (2006). Expectation correction for smoothed inference in switching linear dynamical systems, *Journal of machine learning research* **7**: 2515 – 2540.

Berryman, J. (1995). Mixture theories for rock properties, *in* T. J. Ahrens (ed.), *Rock Physics and Phase Relations, A Handbook of Physical Constants*, Am. Geophys. Union, Washington, D.C., pp. 205–228.

Buland, A., Kolbjørnsen, O. and Omre, H. (2003). Rapid spatially coupled AVO inversion in the Fourier domain, *Geophysics* **68**: 824–836.

Buland, A. and Omre, H. (2003). Bayesian linearized AVO inversion, *Geophysics* **68**: 185–198.

Cappé, O., Moulines, E. and Rydén, T. (2005). *Inference in Hidden Markov Models*, Springer.

Han, X. L. and Green, P. J. (1992). Metropolis methods, Gaussian proposals, and antithetic variables *in* P. Barone, A. Frigessi and M. Piccioni (eds), *Stochastic Models, Statistical Methods and Algorithms in Image Analysis, number 74 in Lecture Notes in Statistics, Springer, Berlin,* pp. 142–164.

Hastings, W. (1970). Monte Carlo sampling using Markov chains and their applications, *Biometrica* **57**: 97–109.

Künsch, H. (2000). State space models and hidden Markov models, *in* O. Barndorff-Nielsen, D. Cox and C. Klüppelberg (eds), *Complex Stochastic Systems*, number 87 in *Monographs on Statsitics and Applied Probability*, Chapman & Hall/CRC.

Larsen, A. L., Ulvmoen, M., Omre, H. and Buland, A. (2006). Bayesian lithology/fluid prediction and simulation on the basis of a Markov-chain prior model, *Geophysics* **71 issue 5**: R69–R78.

MacDonald, I. and Zucchini, W. (1997). *Hidden Markov and Other Models for Discrete-Valued Time Series*, Chapman and Hall.

Scott, A. L. (2002). Bayesian methods for hidden Markov models: Recursive compution in the 21st century, *Journal of the American Statistical Association* **97**: 337–351.

Sheriff, R. E. and Geldart, L. P. (1995). *Exploration Seismology*, Cambridge university press.

Zoeter, O. and Heskes, T. (2006). Deterministic approximate inference techniques for conditionally Gaussian state space models, *Statistics and Computing* **16**: 279–292.

Paper IV

Empirical comparison of two Bayesian lithology–fluid prediction algorithms

Hugo Hammer and Marit Ulvmoen
Department of Mathematical Sciences
Norwegian University of Science and Technology
Trondheim, Norway

Abstract

We consider a Bayesian model for doing lithology–fluid prediction from prestack (amplitude versus offset) seismic data. Related to the Bayesian model, we look at two inversion algorithms. The first algorithm simulates from the posterior distribution with no approximations, but the algorithm is quite computer demanding. The second inversion algorithm introduces an approximation in the likelihood model and is in this way able to evaluate the resulting approximate posterior distribution very rapidly. The consequences of the approximation for the inversion result are not clear. The objective of this paper is to evaluate the consequences of the approximation in a synthetic but realistic empirical case study. The consequences are evaluated by comparing the inversion results from the two inversion algorithms. In the case study we observe that, dependent on the parameters in the model, typically the approximate likelihood model preserves between 55% and 80% of the information in the original likelihood model. The consequences of the approximation increase when the amount of noise in the model increases. The approximation works better when most of the variability is in the rock physics model and it is little seismic noise, compared to the opposite.

Keywords: amplitude versus offset seismic data, Bayesian model, empirical comparison, forward–backward algorithm, lithology–fluid prediction, seismic inversion

1 Introduction

In lithology–fluid (LF) prediction we want to determine the lithology and fluid properties of a reservoir using inversion techniques on seismic data. For an introduction to and presentation of several techniques for doing LF prediction we recommend Avseth et al. (2005) and references therein. We focus on LF prediction based on amplitude-versus-offset (AVO) seismic data in one vertical profile. Before the data are used for inversion, noise effects like moveout, multiples and the effect of geometrical spreading and absorption are removed. The data are also prestack migrated such that any dip related effects are removed. We focus on the Bayesian model presented in Larsen et al. (2006) and Hammer and Tjelmeland (2008). As a prior model for the LF properties a stationary Markov chain is used. Further, the model consists of a likelihood model that relates the LF properties to a set of elastic material properties and the AVO seismic data. The model is closely related to Buland and Omre (2003) and Buland et al. (2003).

To evaluate the model given in Larsen et al. (2006) and Hammer and Tjelmeland (2008) is a challenging task. Direct sampling from the posterior distribution is infeasible and single site Metropolis–Hastings algorithms give extremely slow convergence. Hammer and Tjelmeland (2008) simulate from the model using a more sophisticated block Metropolis–Hastings (MH) algorithm that proposes changes for all the variables in the model in each iteration. Larsen et al. (2006) introduce an approximation in the likelihood model and evaluate the resulting approximate posterior exactly and rapidly. For example, the algorithm needs about a second to compute the approximate posterior distribution by recursion and generates therefore independent realizations every half millisecond. In comparison, the algorithm in Hammer and Tjelmeland (2008) needs hours to estimate properties like marginal distributions or maximum posterior value and generates independent realizations about every minute.

Central in both of the inversion algorithms is a Hidden Markov model (HMM). In a HMM we suppose that the observations are a noisy function of an underlying and unknown Markov process. In traditional HMM the underlying process is a Markov chain. Then the posterior is in fact a non-stationary Markov chain and can be evaluated analytically, exact and very efficiently using the Forward-backward (FB) algorithm (Scott, 2002). The FB algorithm consists of two parts. In the first part we iterate forward computing all the transition propabilities in the non-stationary Markov chain. In the second part we iterate backward evaluating the posterior distribution.

What consequences the approximation in Larsen et al. (2006) have on the inversion results are not clear. The objective of this paper is to evaluate these consequences in an empirical case study. Since the MH algorithm in Hammer and Tjelmeland (2008) simulates from the posterior distribution without approximations, we are able to analyse the consequence of the approximations by comparing the approximate inversion results with the exact inversion results.

The paper is organised as follows. In Section 2 we present the Bayesian model from Larsen et al. (2006) and Hammer and Tjelmeland (2008). In Section 3 we give a review of the inversion methods in the same two papers. Further, in Section 4 we discuss criteria to compare the inversion results and finally in Section 5 we analyse the consequence of the approximation in Larsen et al. (2006) in a synthetic but realistic case study.

2 Bayesian seismic model

The Bayesian seismic model relates the LF properties along a vertical profile with elastic material properties and observed AVO seismic data. The Bayesian model presented is identical to the one used in Hammer and Tjelmeland (2008) and only differs from Larsen et al. (2006) in the formulation of the rock physics model. We follow the notation in Larsen et al. (2006). We define a discretised Bayesian formulation. We use $i = 1, \ldots, n$ to denote n travel times along the vertical profile. Let π_i denote LF classes and let α_i, β_i and γ_i denote P-wave velocity, S-wave velocity and density, respectively, at position i. Further define $m_i = (m_{i1}, m_{i2}, m_{i3}) = (\ln(\alpha_i), \ln(\beta_i), \ln(\gamma_i))$. We consider s offset values $\theta_1, \ldots, \theta_s$ and let $c_i = (c_{i1}, \ldots, c_{is})^T \in \mathbb{R}^s$ and $d_i = (d_{i1}, \ldots, d_{is})^T \in \mathbb{R}^s$ denote reflection coefficients and seismic data, respectively, at position i for the s different offsets.

2.1 Prior model

As a prior model for the LF classes along the vertical profile, $\pi = (\pi_1, \ldots, \pi_n)^T$, we assume a stationary Markov chain

$$\mathrm{P} = [p(\pi_i|\pi_{i-1})]^L_{\pi_{i-1},\pi_i=1}. \tag{1}$$

Further, let $p(\pi_1)$ denote the marginal distribution for π_1.

2.2 Likelihood model

The distribution for $m = (m_1^T, \dots, m_n^T)^T$ given π is related to a rock physics model (Mavko et al., 1998). We assume the relation to be Gaussian distributed

$$[m_i|\pi_i] \sim \mathrm{N}(m_i|\mu(\pi_i), \Sigma(\pi_i)), \tag{2}$$

where $\mathrm{N}(x|\mu, \Sigma)$ denotes a multivariate Gaussian distribution evaluated in x, with expectation μ and covariance matrix Σ. We assume $\mu(\pi_i)$ and $\Sigma(\pi_i)$ to be known and that $m_1, \dots, m_n$ are conditionally independent given π.

Further, we assume that the reflection coefficients, $c = (c_1^T, \dots, c_n^T)^T$, to be related to m through a weak contrast approximation of the Zoeppritz equations (Aki and Richards, 1980; Buland and Omre, 2003)

$$c_i = \Lambda^T \cdot \left(\frac{m_{i+1} - m_{i-1}}{2} \right) \text{ for } i = 2, \dots, n-1, \tag{3}$$

where

$$\Lambda = \begin{bmatrix} a_\alpha(\theta_1) & a_\alpha(\theta_2) & \cdots & a_\alpha(\theta_s) \\ a_\beta(\theta_1) & a_\beta(\theta_2) & \cdots & a_\beta(\theta_s) \\ a_\rho(\theta_1) & a_\rho(\theta_2) & \cdots & a_\rho(\theta_s) \end{bmatrix} \tag{4}$$

and

$$a_\alpha(\theta) = \frac{1}{2}(1 + \tan^2(\theta)) \tag{5}$$

$$a_\beta(\theta) = -4\overline{\beta/\alpha}^2 \sin^2(\theta) \tag{6}$$

$$a_\rho(\theta) = -4\left(\overline{\beta/\alpha}^2 \sin^2(\theta) \right). \tag{7}$$

The differences in (3) represent approximations to derivatives originating from the continous version of the model. To avoid boundary problems for $i = 1$ and $i = n$ in (3), we use forward and backward differences, respectively. The relation between m and c may be written on a linear matrix form. Let $A = I_n \otimes \Lambda^T$, where I_n is the $n \times n$ identity matrix and $\otimes$ represent the Kronecker product. Moreover, let D be a matrix representing the approximations to the derivatives in (3). We then may write (3) on a linear form as $c = ADm$.

The seismic data $d = (d_1^T, \dots, d_n^T)^T$ are related to the reflection coefficients through a wavelet convolution

$$d_{ij} = \sum_{u=-k}^{k} w(u, \theta_j) \cdot c_{i-u,j} + \varepsilon_{ij}, \tag{8}$$

where $w(\cdot, \cdot)$ is a wavelet function. We assume ε_{ij} to be coloured Gaussian noise, given as

$$\varepsilon_{ij} = \sum_{u=-k}^{k} w(u, \theta_j)\varepsilon^1_{i-u,j} + \varepsilon^2_{i,j} \tag{9}$$

where $\varepsilon^1_{i,j}$ and $\varepsilon^2_{i,j}$ are independent Gaussian white noise with $\mathrm{Var}(\varepsilon^1_{i,j}) = \sigma_1^2$ and $\mathrm{Var}(\varepsilon^1_{i,j}) = \sigma_2^2$. We see that the first part of the noise has the same waveform as the wavelet while the second part is white nose. The relation in (8) may also be written on linear matrix form

$$d = Wc + W\varepsilon^1 + \varepsilon^2 = Wc + \varepsilon. \tag{10}$$

Finally, by defining $G = WAD$, we get

$$d = WADm + W\varepsilon^1 + \varepsilon^2 = Gm + \varepsilon. \tag{11}$$

Remark 1. As mentioned above, the Bayesian model in Larsen et al. (2006) only differs from the model presented above in the rock physics formulation. Larsen et al. (2006) assume $p(m_i|\pi_i)$ to be an empirical distribution in stead of the Gaussian assumption in (2) and are in this way able to model more general rock physics models. In this paper we only consider the rock physics model in (2).

2.3 Posterior distribution

The posterior distribution resulting from the prior and likelihood models presented above can be written as

$$p(\pi|d) \propto \int \dots \int p(d|m) \prod_{i=1}^{n} p(m_i|\pi_i) \mathrm{d}m \prod_{i=1}^{n} p(\pi_i|\pi_{i-1}). \tag{12}$$

Here $p(d|m)$ and $p(m_i|\pi_i)$, refer to the relations in (11) and (2), respectively. In $p(\pi|d)$ and below, we let $p(\pi_1) = p(\pi_1|\pi_0)$ for notational convenience. We integrate out the elastic parameters, m, from the posterior distribution to emphasise that our focus is on the LF classes, π, in the posterior distribution. Larsen et al. (2006) introduce an approximation in the likelihood which makes it possible to evaluate the approximate posterior analytically and fast. The resulting approximate posterior distribution can be written on the form

$$\hat{p}(\pi|d) \propto \prod_{i=1}^{n} l(d|\pi_i) p(\pi_i|\pi_{i-1}). \tag{13}$$

In the next section we give the details in the approximation which results in $\hat{p}(\pi|d)$ and discuss how to simulate from $p(\pi|d)$ and $\hat{p}(\pi|d)$.

3 Inversion algorithms

In Sections 3.1 and 3.2 we give a review of the inversion methods in Larsen et al. (2006) and Hammer and Tjelmeland (2008), respectively.

3.1 Fast and approximate inversion

In this section we present the fast and approximate inversion method of Larsen et al. (2006). From (12), we may write

$$p(\pi|d) \propto \int \dots \int \frac{p(m|d)}{p(m)} \prod_{i=1}^{n} p(m_i|\pi_i) \mathrm{d}m \prod_{i=1}^{n} p(\pi_i|\pi_{i-1}). \tag{14}$$

The distribution $p(m)$ is a mixture of L^n Gaussian components. We are not able to evaluate this distribution or $p(m|d)$ analytically. Instead we substitute $p(m)$ and $p(m|d)$ with distributions $p^\star(m)$ and $p^\star(m|d)$, respectively, but in such a way that the distribution $p(\pi|d)$ is not changed. We acheve this by letting $p^\star(m)$ and $p^\star(m|d)$ satisfy the relation

$$p^\star(m|d) \propto p(d|m) p^\star(m). \tag{15}$$

Here $p(d|m)$ denotes (11) in the likelihood model. Based on (15) we can then write (14) as

$$p(\pi|d) \propto \int \dots \int \frac{p^\star(m|d)}{p^\star(m)} \prod_{i=1}^{n} p(m_i|\pi_i)\mathrm{d}m \prod_{i=1}^{n} p(\pi_i|\pi_{i-1}). \tag{16}$$

We observe that if let $p^\star(m)$ be Gaussian, then also $p^\star(d|m)$ becomes Gaussian based on (15). We therefore let $p^\star(m)$ be Gaussian with the possibility of spatial dependencies

$$p^\star(m) = \mathrm{N}(m; \mu_m^\star, \Sigma_m^\star), \tag{17}$$

where $\Sigma_m^\star = c(\delta) \otimes \Sigma_{m0}^\star$. Here $c(\delta)$ is a correlation function to model spatial dependencies along the vertical profile and $\Sigma_{m0}^\star$ is a 3×3 matrix representing intervariable covariance structure for the elastic parameters.

As a basis for a fast inversion we introduce an approximation by removing the spatial dependence in $p^\star(m)$ and $p^\star(m|d)$ in (16). This means that the covariance matrices in $p^\star(m)$ and $p^\star(m|d)$ are 3×3 block diagonal, only containing the intervariable dependencies. Note that we use $p^\star(m)$ with spatial dependencies when we calculate $p^\star(m|d)$ in (15) before we remove the spatial dependencies. An approximate posterior can then be written as

$$\hat{p}(\pi|d) \propto \prod_{i=1}^{n} \left[p(\pi_i|\pi_{i-1}) \int \frac{p^\star(m_i|d)}{p^\star(m_i)} p(m_i|\pi_i)\mathrm{d}m_i \right]. \tag{18}$$

Since we assume that the rock-physics model $p(m_i|\pi_i)$ is Gaussian distributed, the integrals in (18) are analytically available. The approximate posterior $\hat{p}(\pi|d)$ can then be written as given in (13). The posterior $\hat{p}(\pi|d)$ is in fact a first order hidden Markov model. The approximate posterior can therefore be exactly assessed very rapidly using the Forward-backward (FB) algorithm (Scott, 2002) shortly described in the introduction. Realizations can be generated extremely fast from $\hat{p}(\pi|d)$ of course.

3.2 MCMC simulation: Inversion with no approximations

In this section we summarise the Markov chain Monte Carlo (MCMC) simulation algorithm presented in Hammer and Tjelmeland (2008). The algorithm simulates from $p(\pi|d)$ with no approximations. To define an effective MCMC algorithm, we introduce $z = (z_1, \dots, z_n)^T$, where $z_i = (z_{i1}, \dots, z_{is})^T \in \mathbb{R}^s$ for $i = 1, \dots, n$,

$$z_i = c_i + \varepsilon_i^1 \tag{19}$$

and $\varepsilon_i^1 = (\varepsilon_{i1}^1, \dots, \varepsilon_{is}^1)^T$ as presented in the model in Section 2. Combining (8), (9) and (19) we get the relation between z and d,

$$d_{ij}|z \sim \mathrm{N}\left(d_{ij}; \sum_{u=-k}^{k} w(u, \theta_j) z_{i-u,j},\ \sigma_2^2 \right). \tag{20}$$

The posterior of interest is then

$$\begin{aligned} p(\pi, m, z|d) &\propto p(\pi)p(m|\pi)p(z|m)p(d|z) = \\ &\prod_{i=1}^{n} p(\pi_i|\pi_{i-1}) \cdot \prod_{i=1}^{n} \mathrm{N}\left(m_i|\mu(\pi_i), \Sigma(\pi_i)\right) \cdot \\ &\prod_{i=1}^{n} \mathrm{N}\left(z_i |\, \Lambda^T \left(\frac{m_{i+1} - m_{i-1}}{2} \right), \sigma_1^2 I_s \right) \cdot \mathrm{N}(d|Wz, \sigma_2^2 I_{ns}), \end{aligned} \tag{21}$$

where W is the wavelet matrix introduced in Section 2. To simulate effectively from (21) we construct an MCMC algorithm consisting of two steps in each iteration. The first step is a Gibbs step (Liu, 2001) where we jointly update m and z conditioned on π and d. We are able to do this efficiently since the resulting distribution is Gaussian. The second step is a Metropolis–Hastings step where we propose new values for π and m conditioned on z from a proposal distribution, $q(\pi, m|z)$. Then the states of π from each iteration of the MCMC algorithm are (dependent) realizations from the posterior distribution in $p(\pi|d)$ after convergence of the Markov chain.

The key part in the MCMC algorithm is the proposal distribution $q(\pi, m|z)$. If we where able to generate proposals from $p(\pi, m|z, d) \propto p(\pi, m|z)$ it would be a Gibbs step and we would always get acceptance for our proposals. Our goal is therefore to generate proposals from a distribution close to $p(\pi, m|z)$. We start by giving an overview of an algorithm that generate realizations from $p(\pi, m|z)$, but this algorithm is not feasible to implement on a computer. Secondly we describe how we introduce approximations which result in the proposal distribution $q(\pi, m|d)$. For a detailed description of the proposal distribution, we refer to the original paper Hammer and Tjelmeland (2008). Here we only give a short review.

So ideally we would like to generate samples from $p(\pi, m|z)$ which is, using (1), (2) and (20),

$$p(\pi, m|z) \propto p(\pi)p(m|\pi)p(z|m) = \prod_{i=1}^{n} p(\pi_i|\pi_{i-1}) \cdot \prod_{i=1}^{n} \mathrm{N}\left(m_i|\mu(\pi_i), \Sigma(\pi_i)\right) \cdot \prod_{i=1}^{n} \mathrm{N}\left(z_i|\, \Lambda^T \left(\frac{m_{i+1} - m_{i-1}}{2}\right), \sigma_1^2 I\right). \tag{22}$$

This is a Hidden Markov model of order two with two hidden layers, π and m, and one observed layer, z. The FB procedure described for the traditional HMMs in Section 1 can also be used for this model, but the computational efficiency will be much reduced.

The forward part of a forward-backward algorithm for $p(\pi, m|z)$ sequentially integrates out m_i and π_i for $i = 1, \ldots, n$. Because of the Markovian structure of $p(\pi, m|z)$ and that the π_i's are discrete and the m_i's are Gaussian these integrals are analytically available. However, the result after having integrated over $(\pi_i, m_i), i = 1, \ldots, k$ is a mixture of L^k Gaussian densities. Thus, the number of mixture terms grows exponentially and the exact algorithm is computationally feasible only for very small values of n. A corresponding approximate forward integration algorithm is defined by ignoring the less important Gaussian terms, keeping a number of mixture terms that makes the algorithm feasible. Thereafter to do the backward simulation is computationally straight forward. We use the probability distribution defined by this approximate forward-backward algorithm as proposal distribution in a Metropolis–Hastings scheme.

Remark 2. In MCMC simulation one always needs to be careful when it comes to the analysis of the convergence and mixing properties of the Markov chain. The MCMC algorithm described above converges fast and mixes very well, which makes the results reliable. For an analysis of the convergence and mixing properties of the MCMC algorithm, see Hammer and Tjelmeland (2008).

Remark 3. Our main focus will, in the rest of the paper, be on the marginal distribution for the LF classes. The fast and approximate algorithm in Section 3.1 calculates the marginal probabilities by the FB algorithm, while the MCMC algorithm in Section 3.2 estimates these quantities through simulation. This makes the approximate algorithm a much faster alternative.

4 Comparison criteria

We seek for ways to quantify the differences between the true posterior $p(\pi|d)$ and the approximate posterior $\hat{p}(\pi|d)$. As mentioned above, we evaluate $\hat{p}(\pi|d)$ and $p(\pi|d)$ using the algorithms in Sections 3.1 and 3.2, respectively. Our focus will be on the posterior marginal distributions for the LF classes along the vertical profile for the two methods. This is typically the quantities from the inversions (posterior distributions) of main interest. We may observe differences by e.g. inspecting the marginal probabilities visually, but to really quantify the differences, we use univariate measures.

A natural quantity to measure, is the ability the posterior distributions have in regaining the true π. This does not cover all the aspects of the posterior marginal distributions but at least a very important one. A technique to archive this is the use of confusion matrices. We construct $L \times L$ confusion matrices, $C(p) = [c_{i,j}]$ such that row i give the posterior probabilities for the different LF classes when LF class i is the truth. More specifically we define

$$c_{i,j} = \frac{\sum_{k=1}^{n} \mathrm{I}(\pi_k^0 = i) P(\pi_k = j|d),}{\sum_{k=1}^{n} \mathrm{I}(\pi_k^0 = i)} \tag{23}$$

where $\mathrm{I}(\cdot)$ represent the indicator function and π_k^0 the true LF class at node k. The posterior marginal probabilities $P(\pi_k = j|d)$ can either be calculated from $p(\pi|d)$ or $\hat{p}(\pi|d)$. Clearly, we want the diagonals in the confusion matrices to be close to one.

Now we want to measure to what extent a posterior distribution do wrong classifications related to the true π, and define

$$\Delta(p) = \frac{1}{n} \sum_{k=1}^{n} \sum_{i \neq \pi_k^0} P(\pi_k = i|d) = \frac{1}{n} \sum_{k=1}^{n} \left[1 - P(\pi_k = \pi_k^0|d)\right]. \tag{24}$$

We see that for each node in (24) we have the posterior probability for all the LF classes except the true LF class π_k^0. Naturally we want $\Delta(p)$ to be small. We may calculate (24) based on either $p(\pi|d)$ or $\hat{p}(\pi|d)$ and denote this $\Delta(p)$ and $\Delta(\hat{p})$, respectively. We may also want to calculate (24) based on the prior marginal probabilities, $p(\pi_k)$, and denote the measure $\Delta(pr)$.

It is natural to assume that in a practical situation some wrong classifications have more consequences then others. For example confusing oil with gas may not be as dramatic as confusing oil with some other rock type, for example shale. We incorporate this by defining an $L \times L$ loss matrix $\Gamma = [\gamma_{i,j}]$, where $\gamma_{i,j}$ quantify the consequence of classifying to the class j if i is the true LF class. A natural generalisation of (24) is then

$$\Delta_\Gamma(p) = \frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{L} \gamma_{\pi_k^0,i} \, P(\pi_k = i|d). \tag{25}$$

Similar to (24) we also want (25) to be small. Similar to above we may calculate (25) based on prior probabilities, $p(\pi|d)$ or $\hat{p}(\pi|d)$ and denote the resulting measures $\Delta_\Gamma(pr)$, $\Delta_\Gamma(p)$ and $\Delta_\Gamma(\hat{p})$, respectively.

We also introduce a metric to quantify the difference between discrete distributions. The metric is inspired by Endres and Schindelin (2003), but we multiply by the factor $1/(2\ln(2))$ so that the maximal distance between the two distributions will be 1. Let $p = (p_1, \dots, p_L)$

and $q = (q_1, \dots, q_L)$ represent two discrete distributions. We quantify the difference between p and q with

$$d(p,q) = \sqrt{\frac{1}{2\ln(2)} \sum_{i=1}^{L} \left(p_i \ln\left(\frac{2p_i}{p_i+q_i}\right) + q_i \ln\left(\frac{2q_i}{p_i+q_i}\right)\right)}. \tag{26}$$

We then have that $d(p,q)^2$ satisfies the requirement of being a metric. At an arbitrary node k, we are typically interested in measuring differences between the marginal prior probabilities $p(\pi_k)$, posterior marginal probabilities $p(\pi_k|d)$ and approximate posterior marginal probabilities $\hat{p}(\pi_k|d)$. We also want to compare these marginal probabilities against the true π by defining a "probability distribution" for the true π at a node being 1 for the true LF class and 0 for the others. More specifically, at an arbitrary node k we denote this "probability distribution" $p_k^0 = (p_{k1}^0, \dots, p_{kL}^0)$ with $p_{ki}^0 = \mathrm{I}(\pi_k^0 = i)$ for $i = 1, \dots, L$. It is natural to evaluate the differences between the distribution above for the whole profile. We compute the difference by taking the average of the differences for each node. For example for the difference between the true π and the prior distribution we compute and denote this as follows

$$\overline{d}(p^0, pr) = \frac{1}{n} \sum_{k=1}^{n} d(p_k^0, p(\pi_k)). \tag{27}$$

We denote the difference from the true π to $p(\pi|d)$ and $\hat{p}(\pi|d)$ with $\overline{d}(p^0, p)$ and $\overline{d}(p^0, \hat{p})$, respectively. We denote other combinations of differences similarly.

The posterior distribution is a result of both the prior distribution and the likelihood model. The approximation introduced to get $\hat{p}(\pi|d)$ is done in the likelihood. A natural quantification of the consequences of the approximations is then to see how much the likelihood models correct on the prior information. We then define based on (24), (25) and (26)

$$\rho_\Delta = \frac{\Delta(pr) - \Delta(\hat{p})}{\Delta(pr) - \Delta(p)} \tag{28}$$

$$\rho_{\Delta_\Gamma} = \frac{\Delta_\Gamma(pr) - \Delta_\Gamma(\hat{p})}{\Delta(pr) - \Delta_\Gamma(p)} \tag{29}$$

$$\rho_d = \frac{\overline{d}(p^0, pr) - \overline{d}(p^0, \hat{p})}{\overline{d}(p^0, pr) - \overline{d}(p^0, p)}. \tag{30}$$

If the approximations introduced in the likelihood work well, we will expect the numerator to be almost as large as the denominator in ρ_Δ, ρ_{Δ_Γ} and ρ_d. If the approximations works poorly, the approximate likelihood adds little information to the posterior distribution and the numerator will be small compared to the denominator. This means that ρ_Δ, ρ_{Δ_Γ} and ρ_d can be interpreted as quantifying the portion of the information in the true likelihood preserved by the approximate likelihood.

5 Simulation example

We are now interested in evaluating the approximate posterior $\hat{p}(\pi|d)$ against the correct posterior $p(\pi|d)$ in a synthetic but realistic seismic inversion simulation example. We evaluate $\hat{p}(\pi|d)$ using the fast FB algorithm in Section 3.1 and simulate from $p(\pi|d)$ using the MCMC

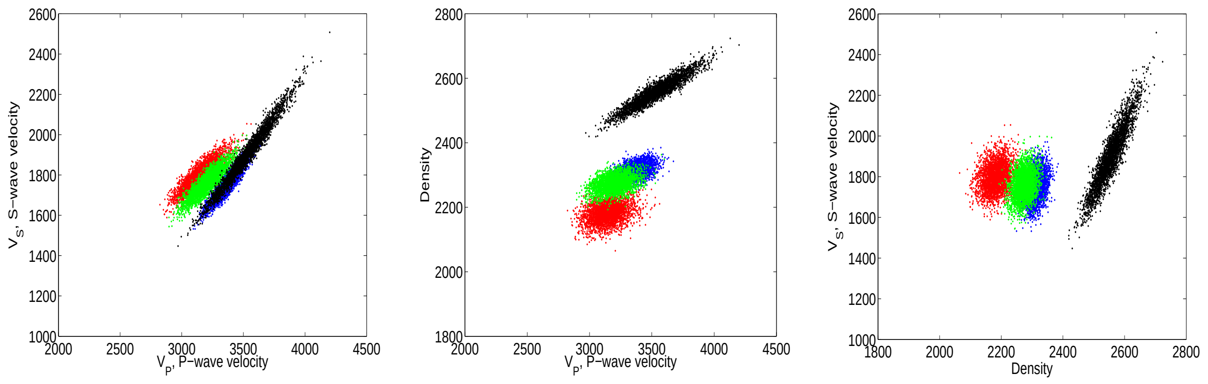


Figure 1: 3000 realizations from the rock physics model considered in the cases BC, NN, NL, MN and NZ. Red, green, blue and black represent gas, oil and brine saturated sandstone, and shale, respectively.

algorithm in Section 3.2. The general test design is to generate data from the forward model in Section 2 and secondly perform inversions using the two algorithms in Section 3. We evaluate the inversion results based on the criteria described in Section 4.

5.1 Model parameter values

We will evaluate the posteriors $p(\pi|d)$ and $\hat{p}(\pi|d)$ for seven different choices for the parameters in the model. The parameters we choose are closely related to the choices in Larsen et al. (2006) and Hammer and Tjelmeland (2008). We start by presenting one case, which we characterise as the base case and denote with BC. Secondly we present how the other six cases are different from BC. We use a length of the profile equal to $n = 100$. We consider $L = 4$ LF classes, representing gas, oil and brine saturated sandstone, and shale. We use the following prior transition matrix

$$P = \begin{bmatrix} 0.9441 & 0 & 0 & 0.0559 \\ 0.0431 & 0.9146 & 0 & 0.0424 \\ 0.0063 & 0.0230 & 0.9422 & 0.0284 \\ 0.0201 & 0.0202 & 0.1006 & 0.8591 \end{bmatrix} \tag{31}$$

defined upward along the vertical profile. The resulting marginal probabilities are $p(\pi_i) = [0.2419, 0.1552, 0.3830, 0.2199]$. The transition matrix is the same used in Hammer and Tjelmeland (2008) and closely related to the one in Larsen et al. (2006). The first row are the probabilities going from gas, the second row going from oil, the third brine saturated sandstone and the forth going from shale. We observe that we have zero probabilities for transitions from gas saturated sandstone to oil saturated sandstone, from gas saturated sandstone to brine saturated sandstone and from oil saturated sandstone to brine saturated sandstone, which seems reasonable due to gravity.

To illustrate the rock physics model, we generate 3000 realizations from $p(m_i|\pi_i)$ in (2) for each of the four LF classes and secondly calculate the resulting P-wave velocity, S-wave velocity and density, see Figure 1. In this figure and in the figures below, we let red, green, blue and black represent gas, oil and brine saturated sandstone, and shale, respectively. We

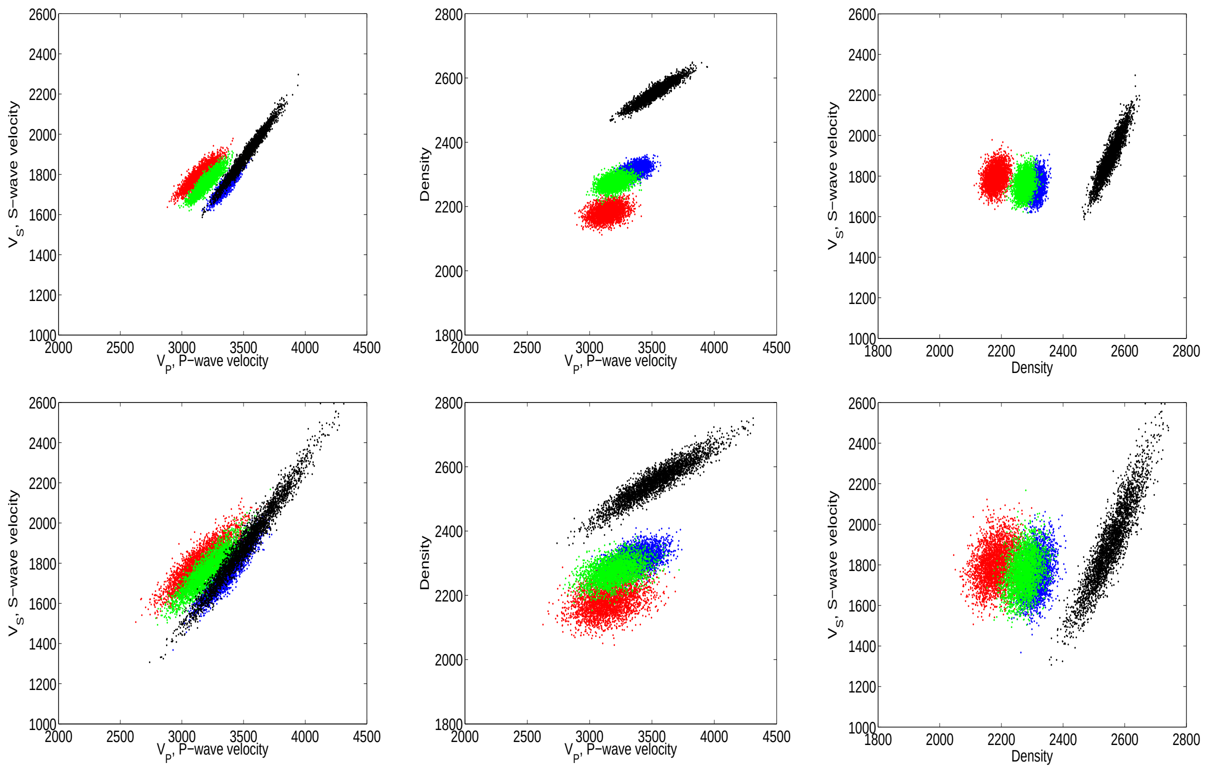


Figure 2: In row one and two we have 3000 realizations from the rock physics models considered in the cases RL and RM, respectively. Red, green, blue and black represent gas, oil, brine saturated sandstone and shale, respectively.

consider $s = 5$ angles $[0°, 10°, 20°, 30°, 40°]$ and use an angle independent Ricker wavelet

$$w(u) = (1 - 2(\pi\,\phi\,u)^2)\exp\left(-(\pi\,\phi\,u)^2\right),\; u = \{-k, \dots, k\} \qquad (32)$$

in (20), where we let $\phi = 0.11$ and $k = 10$.

In the base case BC, we set $\sigma_1 = 1.5 \cdot 10^{-2}$ and $\sigma_2 = 0.01\sigma_1$. This results in a reasonable Signal-to-noise ratio in the model. We return to this below. We consider six other cases. In the first case different from the BC, we change σ_1 to $5.0 \cdot 10^{-4}$ which can be interpreted as a case where we have essentially no noise added and denote the case NN. For the next two cases we change σ_1 to $0.85 \cdot 10^{-2}$ and $2.6 \cdot 10^{-2}$, which can be interpreted as cases with little and much noise and we denote the cases LN and MN, respectively. Similar to BC, we let in all the cases $\sigma_2 = 0.01\sigma_1$. For the next two cases we change the variability in the rock physics model. In the fifth case we divide the covariance matrices in (2) for the BC with two and in the sixth case we multiply the covariance matrices with two. The rock physics models are illustrated in Figure 2. We denote the cases RL and RM respectively, meaning rock physics models with less and more variability. For RL and RM we choose $\sigma_1 = 1.7 \cdot 10^{-2}$, $\sigma_1 = 0.95 \cdot 10^{-2}$, respectively and $\sigma_2 = 0.01\sigma_1$. We return to a motivation for these choices below. In the last case we change the prior transition matrix. We want to see if using a prior distribution with no zero elements, in contrast to (31), effects the difference between the posteriors $p(\pi|d)$ and $\hat{p}(\pi|d)$. In stead of (31) we use a transition matrix with 0.91 on the diagonal and 0.03 in all the other

Case	Description
BC	Base case
NN	No noise
LN	Little noise
MN	Much noise
RL	Rock physics model with less variability
RM	Rock physics model with more variability
NZ	Transition matrix with no zeros

Table 1: Description of the cases considered.

Case	Trans. matrix	Rock phys. model	σ_1	σ_2	SN	SN*
BC	(31)	Figure 1	$1.5 \cdot 10^{-2}$	$0.01\,\sigma_1$	2.91	1.27
NN	(31)	Figure 1	$5.0 \cdot 10^{-4}$	$0.01\,\sigma_1$	2350	3.49
LN	(31)	Figure 1	$0.85 \cdot 10^{-2}$	$0.01\,\sigma_1$	8.44	2.17
MN	(31)	Figure 1	$2.6 \cdot 10^{-2}$	$0.01\,\sigma_1$	0.95	0.64
RL	(31)	Row one Figure 2	$1.7 \cdot 10^{-2}$	$0.01\,\sigma_1$	1.90	1.28
RM	(31)	Row two Figure 2	$0.95 \cdot 10^{-2}$	$0.01\,\sigma_1$	8.44	1.31
NZ	(33)	Figure 1	$1.5 \cdot 10^{-2}$	$0.01\,\sigma_1$	3.02	1.32

Table 2: Parameter values and resulting Signal-to-noise ratios for the different cases considered.

elements

$$P_{NZ} = \begin{bmatrix} 0.91 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.91 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.91 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.91 \end{bmatrix}. \tag{33}$$

We denote this case NZ. For this case we use the same noise levels σ_1 and σ_2 as in BC, which result in a Signal-to-noise ratio equal to that in BC. All the cases are summarised in Tables 1 and 2.

The parameter sets in Table 2 are chosen to get reasonable Signal-to-noise ratios in the model. Similar to Hammer and Tjelmeland (2008) we find it natural to introduce two different interpretations of the Signal-to-noise ratio denoted with SN and SN*. For a more detailed description of the Signal-to-noise ratios, see Hammer and Tjelmeland (2008). For SN we consider m as the origin for the seismic signal and ε in (10) as the noise part. This is the way the Signal-to-noise ratio normally is defined in the seismic inversion literature, probably because the objective there is an inversion back to m and not all the way back to π. Typical values for SN in the literature are between 1.0 and 5.0. For SN* we consider π as the origin for the seismic signal and both the variability in the rock physics model in (2) and ε as noise parts. The Signal-to-noise values for the seven situations are given in Table 2. We see that SN* is approximately the same for the cases BC, RL, RM and NZ. It is easier to observe the effect of other changes in the model by holding SN* constant. The effect of changes in SN and SN* are studied by comparing the cases BC, NN, LN and MN.

In addition to the parameter choices in the model, we need to find a suitable correlation function $c(\delta)$ for the inversion method in Section 3.1. By generating samples of m from the stochastic model and inspecting the spatial correlations in the samples, we find a correlation function of the form

$$c(\delta) = \exp\left(-\delta^{1/2}/3\right) \tag{34}$$

suitable.

5.2 Results with discussion

In this section we analyse the difference between the posteriors $\hat{p}(\pi|d)$ and $p(\pi|d)$ through several simulations using the algorithms in Sections 3.1 and 3.2. For each of the seven cases in Table 1 we generate ten independent realizations of π from the prior (1) and resulting seismic data d using the forward model in Section 2. Secondly we run both of the inversion algorithms for all the resulting 70 sets of data d.

We start by visually inspecting realizations from the ten inversions in each of the cases given in Table 1. Figures 3 and 4 shows a few samples from one of the ten inversions for each of the cases given in Table 1. In the figures, the first column shows the seismic data d, the second column the truth π, the next four columns independent samples from the fast and approximate algorithm in Section 3.1 and in the final four columns four independent samples from the MCMC algorithm in Section 3.2. We see that for both inversion algorithms the amount of variability in the realisations increases when the noise level increases. We see that in all the cases considered the realizations have similarities with the truth π. For the cases NN and LN there are little difference between the realizations from the same inversion algorithm, while for MN there are large differences, as one would expect. It also seems like the realizations from $\hat{p}(\pi|d)$ have some systematic differences compared to the true π while we do not see the same effect for $p(\pi|d)$. Except from these observations, it is hard to say anything more precise about the differences between the posteriors $p(\pi|d)$ and $\hat{p}(\pi|d)$.

A step further is to look at marginal probabilities for one set of seismic data for each case. This is given in Figures 5 and 6. In the two figures, in the first column we have the truth π. In the second and the fourth columns we have the marginal probabilities for the posteriors $\hat{p}(\pi|d)$ and $p(\pi|d)$, respectively. The first can be computed by the FB recursive algorithm while the latter must be assessed by sampling. In the third and the fifth columns we have the distance the marginal probabilities in each node are from the truth using the metric in (26). In the last column we have the difference between the marginal probabilities in columns two and four. It seems like the true posterior $p(\pi|d)$ over all are closer to the true π compared to the approximate posterior $\hat{p}(\pi|d)$ as one would expect.

To be able to draw more definite conclusions we need to summarise the results for all the ten inversions for each of the seven cases. In Table 3 we present confusion matrices for all the seven cases given in Table 1. We have calculated the elements in the confusion matrix according to (23). We have in addition, for each case taken the average of the calculated confusion matrix for each of the ten inversions. We see that the true posterior $p(\pi|d)$ regains the truth better then $\hat{p}(\pi|d)$, as one would expect. However the probabilities along the diagonal is also mostly large for the approximate posterior $\hat{p}(\pi|d)$, which means that also this posterior contains valuable information about the true π.

Now we want to quantify the difference in $p(\pi|d)$ and $\hat{p}(\pi|d)$ using (24) and (25). As mentioned in Section 4 it is natural to expect that some wrong classifications have more

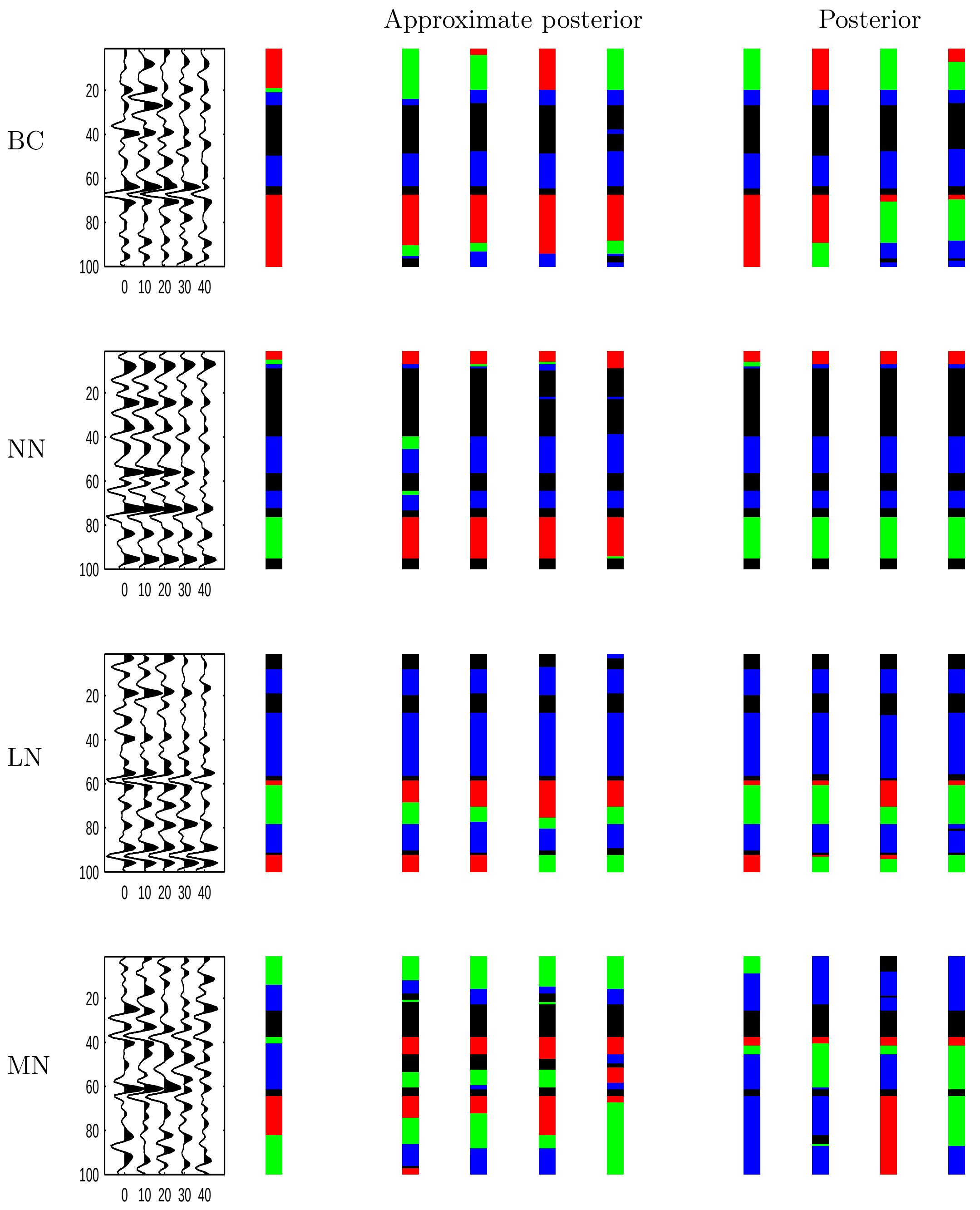


Figure 3: Row 1 to 4 give results for the cases BC, NN, LN and MN, respectively. In the first column we have the seismic data d, the second column the truth π, the next four columns independent samples from the fast and approximate algorithm in Section 3.1 and in the final four columns four independent samples from the MCMC algorithm in Section 3.2.

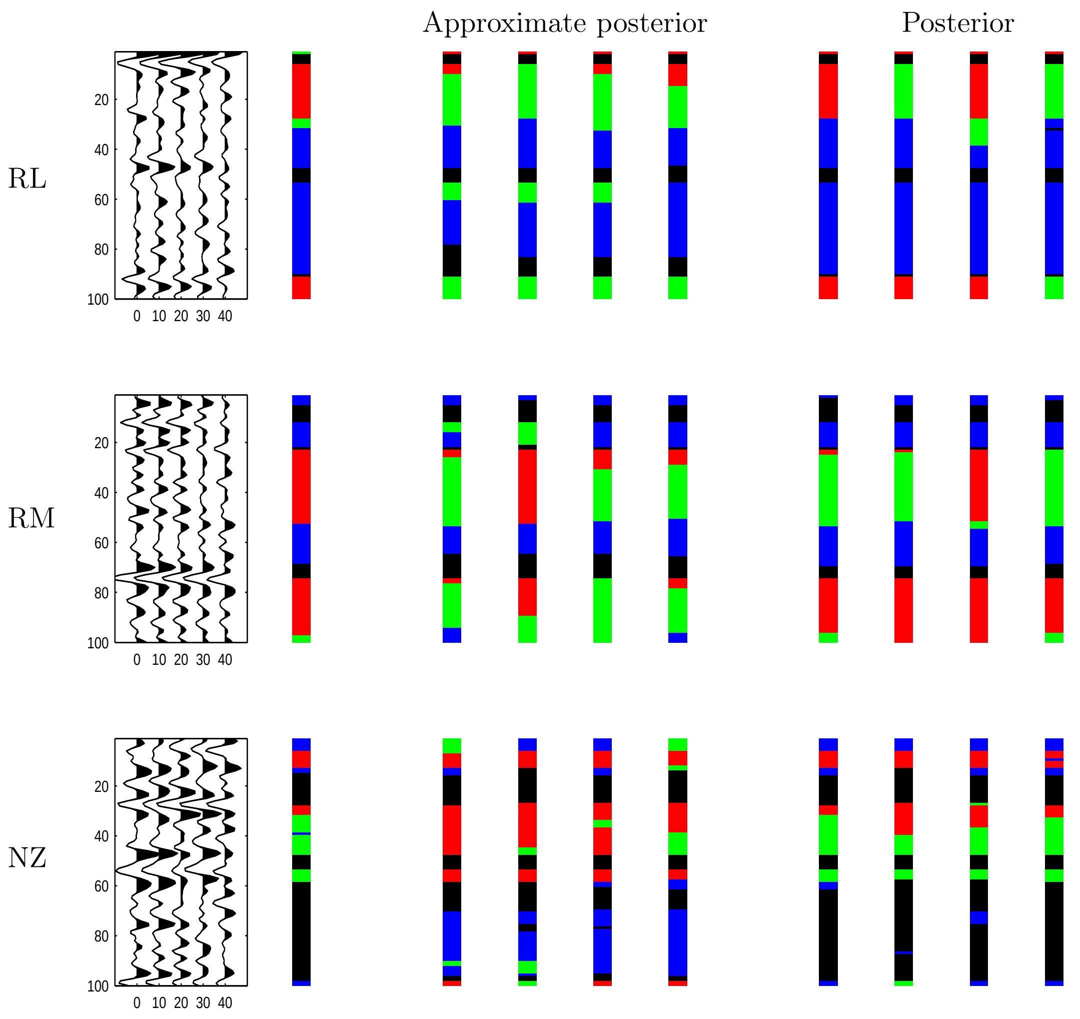


Figure 4: Row 1 to 3 give results for the cases RL, RM and NZ, respectively. In the first column we have the seismic data d, the second column the truth π, the next four columns independent samples from the fast and approximate algorithm in Section 3.1 and in the final four columns four independent samples from the MCMC algorithm in Section 3.2.

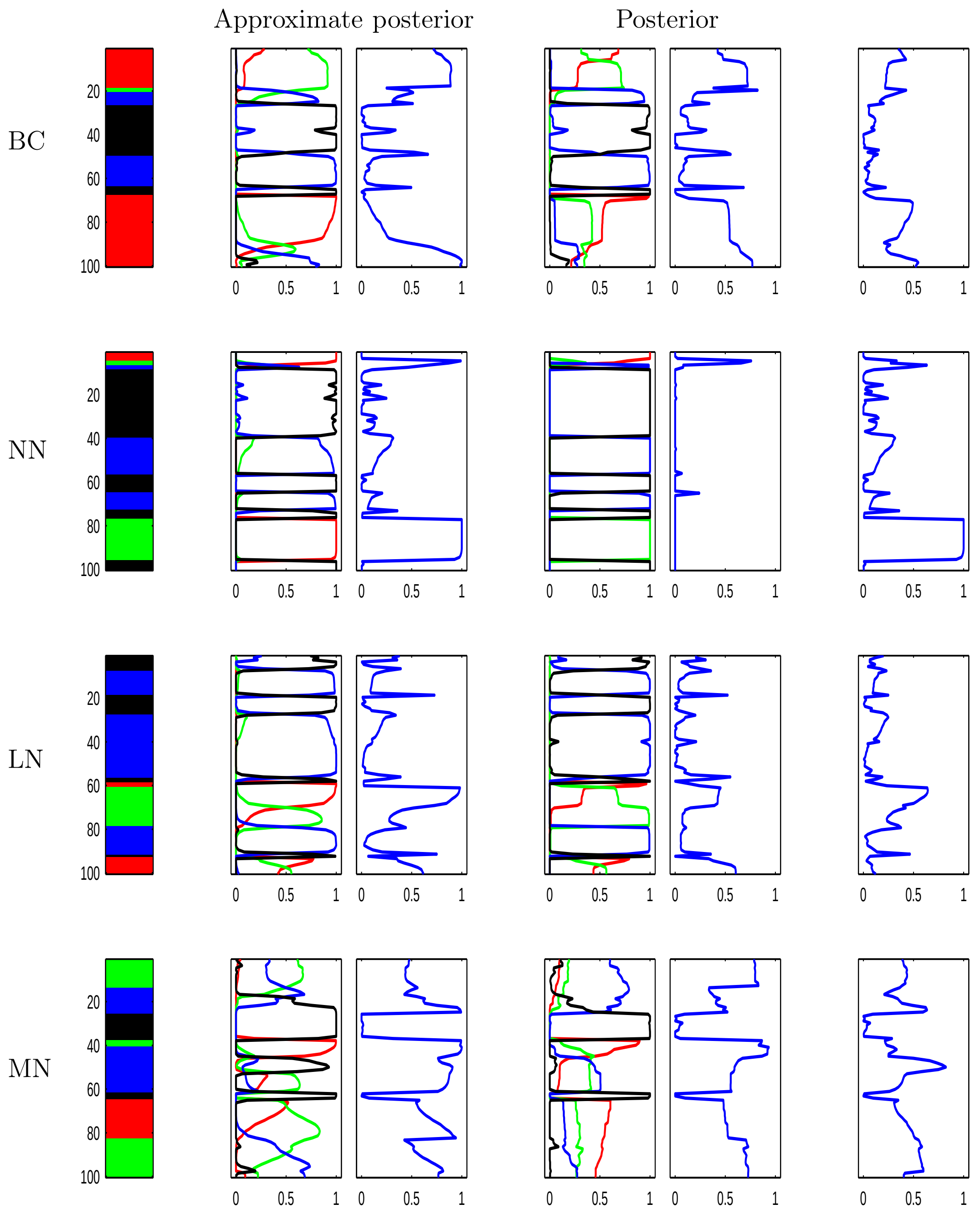


Figure 5: Row 1 to 4 give results for the cases BC, NN, LN and MN, respectively. In the first column we have the truth π. In the second and the fourth columns we have the marginal probabilities for the posteriors $\hat{p}(\pi|d)$ and $p(\pi|d)$, respectively. In the third and the fifth columns we have the distance the marginal probabilities in each node are from the truth using the metric in (26). In the last column we have the difference between the marginal probabilities in column two and four.

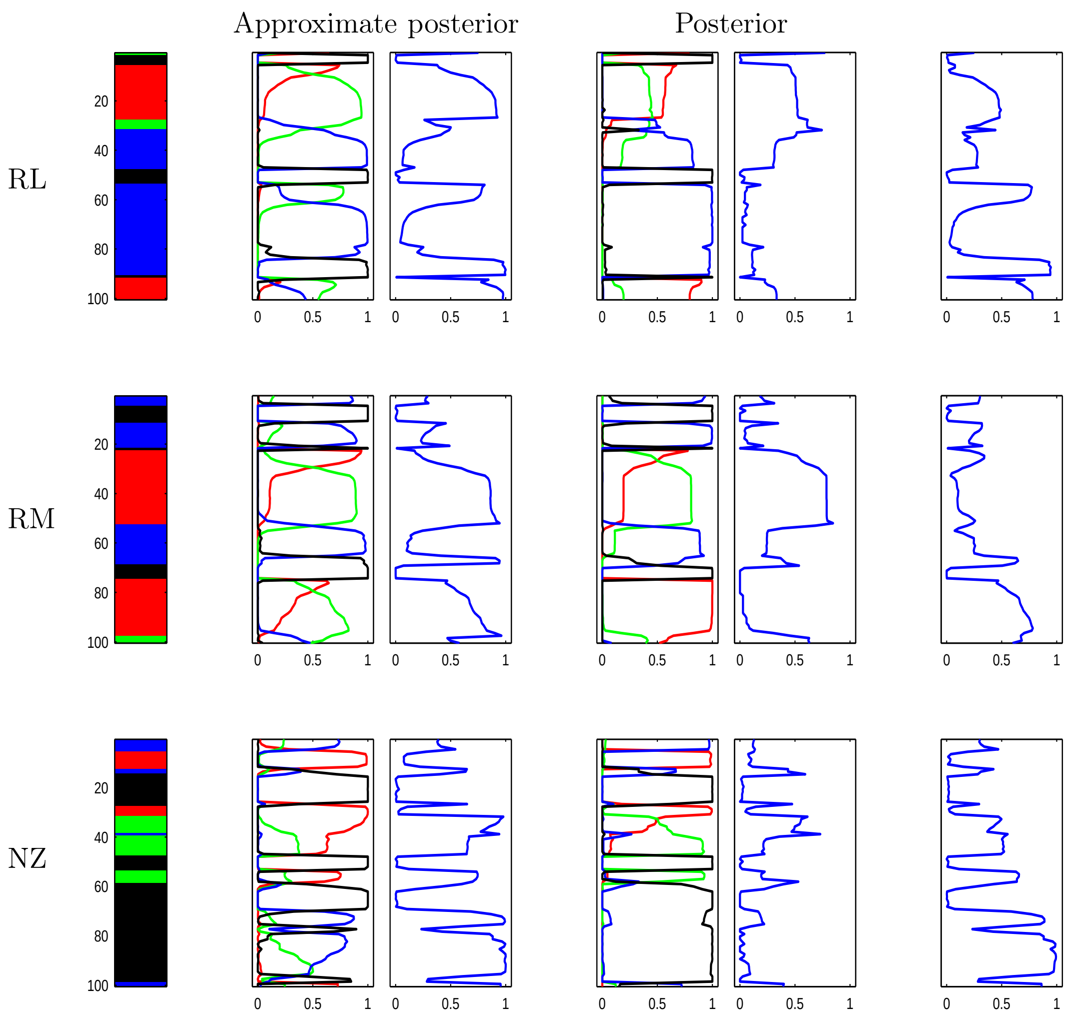


Figure 6: Row 1 to 3 give results for the cases RL, RM and NZ, respectively. In the first column we have the truth π. In the second and the fourth columns we have the marginal probabilities for the posteriors $\hat{p}(\pi|d)$ and $p(\pi|d)$, respectively. In the third and the fifth columns we have the distance the marginal probabilities in each node are from the truth using the metric in (26). In the last column we have the difference between the marginal probabilities in column two and four.

		Approximate posterior				Posterior			
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.459	0.372	0.159	0.001	0.794	0.168	0.035	0.003
BC	oil	0.488	0.451	0.061	0.000	0.410	0.495	0.092	0.003
	brine	0.035	0.196	0.639	0.130	0.003	0.053	0.874	0.069
	shale	0.013	0.002	0.047	0.939	0.001	0.002	0.078	0.920
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.881	0.118	0.000	0.000	0.996	0.004	0	0
NN	oil	0.542	0.447	0.010	0.000	0.018	0.976	0.007	0.000
	brine	0.002	0.044	0.936	0.018	0	0.000	0.997	0.003
	shale	0.001	0.001	0.038	0.961	0	0.000	0.006	0.994
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.959	0.041	0.000	0.000	0.983	0.017	0	0.000
LN	oil	0.224	0.595	0.171	0.010	0.278	0.614	0.108	0.001
	brine	0.014	0.250	0.600	0.136	0.001	0.014	0.953	0.032
	shale	0.006	0.032	0.214	0.748	0.000	0.000	0.081	0.919
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.253	0.458	0.232	0.057	0.623	0.200	0.141	0.037
MN	oil	0.429	0.280	0.213	0.078	0.413	0.313	0.253	0.021
	brine	0.115	0.218	0.474	0.194	0.047	0.137	0.741	0.072
	shale	0.057	0.048	0.141	0.754	0.013	0.029	0.222	0.735
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.759	0.237	0.002	0.002	0.888	0.111	0.000	0.000
RL	oil	0.190	0.333	0.471	0.006	0.113	0.585	0.298	0.003
	brine	0.049	0.311	0.542	0.098	0.003	0.079	0.900	0.018
	shale	0.003	0.012	0.134	0.850	0.001	0.004	0.092	0.903
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.716	0.214	0.061	0.010	0.751	0.234	0.009	0.005
RM	oil	0.635	0.286	0.071	0.008	0.416	0.542	0.035	0.007
	brine	0.014	0.186	0.689	0.112	0.002	0.043	0.892	0.063
	shale	0.019	0.031	0.062	0.888	0.001	0.007	0.073	0.919
		gas	oil	brine	shale	gas	oil	brine	shale
	gas	0.539	0.319	0.140	0.002	0.764	0.206	0.028	0.002
NZ	oil	0.308	0.332	0.336	0.024	0.258	0.632	0.096	0.014
	brine	0.090	0.294	0.407	0.209	0.039	0.287	0.644	0.031
	shale	0.016	0.054	0.137	0.794	0.003	0.025	0.101	0.872

Table 3: Confusion matrices. Row 1 to 7 give results for the cases given in Table 1.

	$\Delta(pr)$	$\Delta(\hat{p})$	$\Delta(p)$	$\Delta_\Gamma(pr)$	$\Delta_\Gamma(\hat{p})$	$\Delta_\Gamma(p)$	ρ_Δ	ρ_{Δ_Γ}
BC	0.722	0.391	0.177	0.503	0.178	0.056	0.605	0.733
NN	0.722	0.162	0.008	0.503	0.031	0.002	0.789	0.942
LN	0.722	0.320	0.120	0.503	0.168	0.034	0.671	0.703
MN	0.722	0.565	0.405	0.503	0.303	0.226	0.500	0.727
RL	0.722	0.388	0.140	0.503	0.261	0.086	0.557	0.562
RM	0.722	0.325	0.198	0.503	0.130	0.044	0.763	0.817
NZ	0.750	0.532	0.286	0.525	0.315	0.152	0.469	0.563

Table 4: Row 1 to 7 give results for the cases given in Table 1. In the columns one to three we have calculated (24) based on the prior distribution, $\hat{p}(\pi|d)$ and $p(\pi|d)$ respectively. In columns four to six we have the same based on (25). In columns seven and eight we have calculated ρ_Δ and ρ_{Δ_Γ} in (28) and (29), respectively.

	$\overline{d}(p^0, pr)$	$\overline{d}(p^0, \hat{p})$	$\overline{d}(p^0, p)$	$\overline{d}(\hat{p}, p)$	ρ_d
BC	0.719	0.430	0.249	0.373	0.615
NN	0.719	0.195	0.013	0.191	0.747
LN	0.719	0.368	0.163	0.353	0.637
MN	0.719	0.600	0.463	0.393	0.480
RL	0.719	0.427	0.214	0.340	0.567
RM	0.719	0.366	0.246	0.240	0.752
NZ	0.741	0.568	0.357	0.428	0.450

Table 5: Row 1 to 7 give results for the cases given in Table 1. For columns one to four we have calculated $\overline{d}(p^0, pr)$, $\overline{d}(p^0, \hat{p})$, $\overline{d}(p^0, p)$ and $\overline{d}(\hat{p}, p)$, respectively. In column five we have calculated ρ_d in (30). For each value, the result are an average over all the ten inversions.

serious consequences than others. We therefore evaluate (25), where we use

$$\Gamma = \begin{bmatrix} 0 & 0.1 & 1 & 1 \\ 0.1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0.1 \\ 1 & 1 & 0.1 & 0 \end{bmatrix}, \tag{35}$$

where the first row quantifies the negative consequences of classifying to gas, oil, brine saturated sandstone and shale if gas is the truth, the second to the forth row is the same for oil and brine santurated sandstone and shale respectively. We quantify that the consequence of confusing the type of hydrocarbon or confusing the non-hydrocarbon classes is not as dramatic as confusing hydrocarbon with non-hydrocarbon. The results are summarised in Table 4. Row 1 to 7 give results for the cases given in Table 1. In columns one to three we have calculated (24) based on the prior distribution, $\hat{p}(\pi|d)$ and $p(\pi|d)$, respectively. In columns four to six we have the same based on (25). In columns seven and eight we have calculated ρ_Δ and ρ_{Δ_Γ} in (28) and (29), respectively. We now quantify the difference between the posterior distributions considering the metric in (26). The results are summarised in Table 5. Row 1 to 7 give results for the cases given in Table 1. For columns one to four we have calculated $\overline{d}(p^0, pr)$, $\overline{d}(p^0, \hat{p})$, $\overline{d}(p^0, p)$ and $\overline{d}(\hat{p}, p)$, respectively. In column five we have calculated ρ_d in (30). Each value is an average of the result from each of the ten inversions.

From row one to four in Tables 4 and 5 we see that the consequences of the approximations increase when the amount of noise in the model increases. From row five and six we see that the consequence of the approximation are smallest for RM meaning that the approximations seem to work better when the variability in the model are mostly in the rock physics model and less in the noise part ε. We get the largest consequences for the case NZ where we have the prior transition matrix without zeros. Using a transition matrix without zeros may be seen as a weaker prior distribution and it then becomes more important to have a likelihood without approximations. From columns seven and eight in Table 4 and column five in Table 5 we see that the approximate likelihood typically preserves between 55% and 80% of the information in the true likelihood function.

6 Closing remarks

We have considered a Bayesian model for doing LF prediction from AVO seismic data. The model contains a Markov chain prior, wavelet convolution and a colored noise term. Related to the Bayesian model, we have considered two inversion algorithms. The first algorithm, from Larsen et al. (2006), introduces an approximation in the likelihood model and is in this way able to evaluate exactly the resulting approximate posterior very efficiently. The other algorithm, presented in Hammer and Tjelmeland (2008), is a more computer demanding alternative, but simulate from the posterior model without approximations. The objective of this paper have been to evaluate the approximations introduced in the efficient algorithm, by comparing the inversion results from the two algorithms.

We have presented a synthetic but realistic study where several different parameter sets in the Bayesian model have been considered. The conclusions is that the approximate likelihood model typically preserves between 55 and 80% of the information in the true likelihood model. The consequences of the approximations increase when the amount of seismic noise in the model increases. The approximations work better in a situation where most of the variability is in the rock physics model and little is seismic noise, compared to the opposite case.

For an inversion problem of the size considered in this paper the algorithm without approximations seems to be the best alternative. In a real situation, we are normally interested in inverting a large amount of traces and then the algorithm without approximations can end up with problems, because it is quite computer demanding. For this situation the approximate alternative in Larsen et al. (2006) is the best alternative. It is also worth to note that the Bayesian model considered in this paper contains several assumptions and approximations. It is an open and relevant question how much the approximations introduced in the approximate algorithm will have on inversion results from real seismic data compared to the impact of the assumptions and approximations in the Bayesian model.

References

Aki, K. and Richards, P. G. (1980). *Quantitative seismology: Theory and methods*, W. H. Freeman and Company.

Avseth, P., Mukerji, T. and Mavko, G. (2005). *Quantitative Seismic interpretation : Applying rock physics tools to reduce interpretation risk*, Cambridge University Press.

Buland, A., Kolbjørnsen, O. and Omre, H. (2003). Rapid spatially coupled AVO inversion in the Fourier domain, *Geophysics* **68**: 824–836.

Buland, A. and Omre, H. (2003). Bayesian linearized AVO inversion, *Geophysics* **68**: 185–198.

Endres, D. M. and Schindelin, J. E. (2003). A New Metric for Probability Distributions, *IEEE Trans. Inform. Theory* **49**: 1858–1860.

Hammer, H. and Tjelmeland, H. (2008). Approximative forward-backward algorithm for a three layer hidden Markov model - with applications to seismic inversion, *Technical Report S4-2008*, Department of Mathematical Sciences, Norwegian University of Science and Technology.

Larsen, A. L., Ulvmoen, M., Omre, H. and Buland, A. (2006). Bayesian lithology/fluid prediction and simulation on the basis of a Markov-chain prior model, *Geophysics* **71 issue 5**: R69–R78.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing*, Springer, Berlin.

Mavko, G., Mukerji, T. and Dvorkin, J. (1998). *The Rock Physics Handbook: Tools for Seismic Analysis of Porous Media*, Cambridge University Press.

Scott, A. L. (2002). Bayesian Methods for Hidden Markov Models: Recursive Compution in the 21st Century, *Journal of the American Statistical Association* **97**: 337–351.