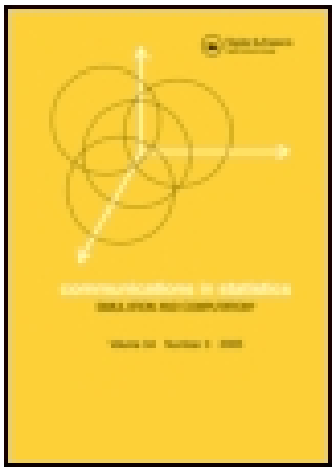

Communications in Statistics - Simulation and Computation
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/lssp20

Optimal Direction Gibbs Sampler for Truncated Multivariate Normal Distributions

J. Andrés Christen[a], Colin Fox[b] & Mario Santana-Cibrian[c]
[a] Centro de Investigación en Matemáticas, A.C. (CIMAT) Guanajuato, Gto., Mexico
[b] Department of Physics, University of Otago Dunedin, New Zealand
[c] Centro de Investigación en Matemáticas, A.C. (CIMAT) Guanajuato, Gto., Mexico
Accepted author version posted online: 30 Jun 2015.

To cite this article: J. Andrés Christen, Colin Fox & Mario Santana-Cibrian (2015): Optimal Direction Gibbs Sampler for Truncated Multivariate Normal Distributions, Communications in Statistics - Simulation and Computation, DOI: 10.1080/03610918.2015.1053926

To link to this article: http://dx.doi.org/10.1080/03610918.2015.1053926

Optimal Direction Gibbs Sampler for Truncated Multivariate Normal Distributions

J. Andrés Christen
Centro de Investigación en Matemáticas, A.C. (CIMAT)
Guanajuato, Gto., Mexico
jac@cimat.mx

Colin Fox
Department of Physics, University of Otago
Dunedin, New Zealand
fox@physics.otago.ac.nz

Mario Santana-Cibrian
Centro de Investigación en Matemáticas, A.C. (CIMAT)
Guanajuato, Gto., Mexico
mariosc@cimat.mx

Abstract

Generalized Gibbs samplers simulate from any direction, not necessarily limited to the coordinate directions of the parameters of the objective function. We study how to optimally choose such directions in a random scan Gibbs sampler setting. We consider that optimal directions will be those that minimize the Kullback-Leibler divergence of two Markov chain Monte Carlo steps. Two distributions over direction are proposed for the multivariate Normal objective function. The resulting algorithms are used to simulate from a truncated multivariate Normal distribution, and the performance of our algorithms is compared with the performance of two algorithms based on the Gibbs sampler.

Keywords: Bayesian inference; MCMC; Gibbs sampler; Simulation, Truncated multivariate Normal

1 Introduction

The Gibbs sampler is a Markov chain Monte Carlo (MCMC) simulation algorithm that sequentially samples from the conditional distributions over subspaces of the full target distribution space. Traditionally, each subspace is chosen by selecting one variable in the target distribution and simulating from the corresponding "full conditional" distribution. The standard Gibbs sampler then systematically samples from all full conditionals to create an irreducible Markov chain; this is the algorithm presented to the Bayesian world in Gelfand and Smith (1990) that initiated the MCMC boom in statistics, although MCMC was already developed in the 1970's (Hastings, 1970; Peskun, 1973). The traditional Gibbs sampler may then be seen as choosing a canonical direction, i.e., selecting a variable to move and simulating from the (full) conditional distribution along that direction. The standard Gibbs sampler only considers directions given by the basis chosen to represent the objective distribution.

A natural question to ask is whether we may take any other direction in the Gibbs sampler and certainly how then to decide which direction to take. Arbitrary directions may indeed be selected, with a new point of the Markov chain chosen by simulating from the conditional distribution along that direction (Liu, 2008). However, it is not clear how to choose such directions and what criterion to use to optimize the resulting chain. Here we address this problem by trying to reduce the dependence in the chain, thereby obtaining more quasi-independent samples with fewer iterations. We use the mutual information (Cover and Thomas, 1991) between two consecutive samples of the Markov chain to measure dependence. To our knowledge, this is a score seldom used in this context.

Our goal is to use the resulting algorithm to sample from a truncated multivariate Normal (TMVN) distribution. This is an important task in many research areas in statistics such as Bayesian Normal linear regression subject to linear inequality restrictions (Rodriguez-Yam et al., 2004), censored data models (Gelfand et al., 1992), order restricted (or isotonic) regressions (Robert,

1995), truncated multivariate probit models in market research (Yu and Tian, 2011), and so on. TMVN sampling is also common in inverse problems, when estimating parameters in a stable system of differential equations leads to the linear problem $\boldsymbol{Bx} = \boldsymbol{y}$, but natural restrictions on the parameter space, such as a positivity constraint $\boldsymbol{x} \geq 0$, result in a truncated multivariate Normal posterior distribution (Calvetti et al., 2008; Kaipio and Somersalo, 2004).

There are several methods for sampling from a TMVN (Breslaw, 1994; Kotecha and Djuric, 1999; Damien and Walker, 2001; Robert, 1995; Yu and Tian, 2011). However, the most common and easiest to implement methods are based on the Gibbs sampler. These methods work well in many situations but may be very slow if we have high correlation as well as high dimensionality. We will show that our algorithm is especially suited for these difficult cases.

The paper is organized as follows. In Section 2 we present the directional Gibbs sampler and the optimality criterion. In Section 3 we present the proposed algorithm for the case where the target distribution is a multivariate Normal (MVN) distribution. Section 4 proposes a variant of the algorithm to sample from the TMVN. In section 5 the performance of the algorithms are evaluated in several examples and also compared with two existing Gibbs samplers. Finally, Section 6 presents a discussion of our results.

2 Generalized Gibbs sampler

Let π be the objective distribution, e.g. the posterior distribution of interest. Let $\boldsymbol{X} \in \mathbb{R}^n$ be a random variable with density $\pi(\boldsymbol{x})$. The component-wise Gibbs sampler is a MCMC sampling algorithm that simulates systematically or randomly from the conditional distributions

$$f_{X_i|\boldsymbol{X}_{-i}}(x_i|\boldsymbol{x}_{-i}) \propto \pi(\boldsymbol{x}), \tag{1}$$

where the notation

$$\boldsymbol{v}_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$$

represents the $(n-1)$-dimensional vector created by deleting the i-th entry from the n-dimension vector $\boldsymbol{v}$. Equation (1) represents univariate distributions that are the conditional distributions along the axis for the basis chosen to represent $\boldsymbol{X}$, the so called "full conditional distributions" (Robert and Casella, 2004).

Using (1), a Markov chain $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots$ is created with transition kernel

$$K_i(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}) = f_{X_i|\boldsymbol{X}_{-i}}(x_i^{(t+1)}|\boldsymbol{x}_{-i}^{(t)})1(\boldsymbol{x}_{-i}^{(t+1)} = \boldsymbol{x}_{-i}^{(t)}).$$

That is, the i-th kernel changes only the i-th coordinate by simulating from the full conditional distribution $f_{X_i|\boldsymbol{X}_{-i}}(\cdot|\boldsymbol{x}_{-i}^{(t)})$. In random scan Gibbs, the complete transition kernel is defined by

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} w_i K_i(\boldsymbol{x}, \boldsymbol{y}),$$

for some weights $w_i \geq 0$, $\sum_{i=1}^{n} w_i = 1$; we prefer a random scan since it creates a reversible Markov chain leading to many desirable analytical properties (Geyer, 1992).

The direction Gibbs sampler generalizes this idea by choosing a direction $\boldsymbol{e} \in \mathbb{R}^n$, $||\boldsymbol{e}|| = 1$, and sampling from the conditional distribution along that direction. This can be written as

$$\boldsymbol{X}^{(t+1)} = \boldsymbol{x}^{(t)} + r\boldsymbol{e},$$

where the length $r \in \mathbb{R}$ has distribution proportional to $\pi(\boldsymbol{x}^{(t)} + r\boldsymbol{e})$ (Liu, 2008). It can be seen that the transition kernel is in detailed balance with π and, by assuring π-irreducibility, the Markov chain has π as ergodic distribution. The natural question to ask is how to choose $\boldsymbol{e}$ to optimize the convergence (mixing) of the Markov chain? Indeed, once irreducibility is assured, any chain will have the correct ergodic distribution π but performance will depend on how dependent are $\boldsymbol{X}^{(t+1)}$ and $\boldsymbol{X}^{(t)}$. In this context, a convenient, although less known dependence measure, is the *mutual information I* between random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ (Cover and Thomas, 1991), which measures the Kullback-Leibler divergence between the joint model $f_{\boldsymbol{X},\boldsymbol{Y}}$ and the independent alternative $f_{\boldsymbol{X}} f_{\boldsymbol{Y}}$,

that is,

$$I(\boldsymbol{Y}, \boldsymbol{X}) = \int \int f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}) \log \frac{f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x})}{f_{\boldsymbol{Y}}(\boldsymbol{y}) f_{\boldsymbol{X}}(\boldsymbol{x})} d\boldsymbol{x} d\boldsymbol{y}.$$

In our case, we have $\boldsymbol{X} = \boldsymbol{X}^{(t)}$ and $\boldsymbol{Y} = \boldsymbol{X}^{(t+1)}$. Assuming that $\boldsymbol{X} \sim \pi$, we see that $f_{\boldsymbol{Y}}(\boldsymbol{y}) = \pi(\boldsymbol{y})$ and $f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}) = \pi(\boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{y})$. Therefore, the mutual information above may be calculated as

$$I(\boldsymbol{Y}, \boldsymbol{X}) = \int \int \pi(\boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{y}) \log \frac{K(\boldsymbol{x}, \boldsymbol{y})}{\pi(\boldsymbol{y})} d\boldsymbol{y} d\boldsymbol{x}. \tag{2}$$

The idea is to choose directions for which $I(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$ is minimized. Since I is a Kullback-Leibler divergence it is well defined; $I \geq 0$ and $I = 0$ if and only if $\boldsymbol{X}^{(t+1)}$ and $\boldsymbol{X}^{(t)}$ are independent, i.e. $f_{\boldsymbol{X},\boldsymbol{Y}} = f_{\boldsymbol{X}} f_{\boldsymbol{Y}}$ a.s.. Finding directions $\boldsymbol{e}$ for which I is minimum will provide our optimization criterion to obtain optimal direction Gibbs samplers.

3 The multivariate Normal case

In this section we will assume π to be a multivariate Normal distribution with mean column vector $\boldsymbol{\mu}$ and $n \times n$ *precision* matrix $\boldsymbol{A}$, which is the inverse of the variance-covariance matrix. This is a case that lends itself to calculation of $I(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$.

As explained in Section 2, given state $\boldsymbol{x}$ and direction $\boldsymbol{e} \in \mathbb{R}^n$, $\|\boldsymbol{e}\| = 1$, the chain moves from $\boldsymbol{X}^{(t)} = \boldsymbol{x}$ to

$$\boldsymbol{Y} = \boldsymbol{X}^{(t+1)} = \boldsymbol{x} + r\boldsymbol{e}$$

where $r \in \mathbb{R}$ has probability density function g proportional to $\pi(\boldsymbol{x} + r\boldsymbol{e})$. That is,

$$g(r|\boldsymbol{e}, \boldsymbol{x}) \propto \exp\left\{-\frac{1}{2}(\boldsymbol{v} + r\boldsymbol{e})^{\mathrm{T}} \boldsymbol{A}(\boldsymbol{v} + r\boldsymbol{e})\right\}$$

where $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{\mu}$. After some algebra we see that

$$r|\boldsymbol{e}, \boldsymbol{x} \sim N\left(-\frac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}, \boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}\right)$$

in which the *precision* is $\boldsymbol{e}^{\mathrm{T}}A\boldsymbol{e}$. By setting $\boldsymbol{e} = \boldsymbol{e}_i$, the i-th standard basis vector, one obtains

$Y_i \sim N\left(\mu_i - \boldsymbol{A}_{ii}^{-1}\boldsymbol{A}_{i,-i}(\boldsymbol{x}_{-i} - \boldsymbol{\mu}_{-i}), \boldsymbol{A}_{ii}\right)$, which is the full conditional distribution for a multivariate Normal distribution for entry i, thus returning to the usual Gibbs sampler.

From this we see that the transition kernel corresponding to direction $\boldsymbol{e}$ is

$$K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}{2}\left(\boldsymbol{e}^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{x}) + \frac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}\right)^2\right\} 1(\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{x})\boldsymbol{e}).$$

Note that $\boldsymbol{y} - \boldsymbol{x} = r\boldsymbol{e}$ and, since $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} = 1$, $r = \boldsymbol{e}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x})$, $\boldsymbol{y}$ is restricted to the line $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{x})\boldsymbol{e}$. The mutual information $I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$ of the Gibbs sampler given direction $\boldsymbol{e}$, as in (2) is then

$$I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)}) = C + \frac{1}{2}\log \boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}, \tag{3}$$

where C is a constant that does not depend on $\boldsymbol{e}$. See Appendix A for further details of the calculation of $I_{\boldsymbol{e}}$.

3.1 Choosing a set of directions

We need now a distribution h for directions to be chosen to generate an *irreducible* Gibbs sampler. According to (3) the best direction is the one that minimizes $C + \frac{1}{2}\log \boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}$. However, we cannot simply choose the best direction as the resulting algorithm will not be irreducible and clearly we will not be sampling from π. The chain must be π-irreducible in order to have ergodic distribution π. Indeed, if directions have distribution h with support in the whole sphere $\mathbb{S}^n$, then the resulting Markov chain is irreducible with transition kernel given by

$$K(\boldsymbol{x}, \boldsymbol{y}) = \int K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) h(\boldsymbol{e}) d\boldsymbol{e}.$$

An alternative will be to actually optimize over the set of all possible direction distributions h that generate irreducible chains. However, this formulation is far less convenient and involves a complex optimization over a function space. Instead we follow a more heuristic approach, as explained below.

Kaufman and Smith (1998) argue that an optimal direction distribution is

$$h(\boldsymbol{e}) \propto \sup_{\boldsymbol{x}\in\mathcal{X},r\in\mathbb{R}} \left\{ \int \pi(\boldsymbol{x}+\tau\boldsymbol{e})d\tau \frac{|r|^{n-1}}{\pi(\boldsymbol{x}+r\boldsymbol{e})} \right\},$$

in the sense of optimizing the geometric rate of convergence of the resulting Gibbs sampler. However, this only applies for π with bounded support $\mathcal{X}$. Little else has been said regarding the optimal directions for generalized Gibbs samplers. In general we cannot control the term $\frac{|r|^{n-1}}{\pi(\boldsymbol{x}+r\boldsymbol{e})}$ for unbounded support. However, this suggests choosing the direction distribution

$$h(\boldsymbol{e}) \propto \sup_{\boldsymbol{x}\in\mathcal{R}^n} \left\{ \int \pi(\boldsymbol{x}+\tau\boldsymbol{e})d\tau \right\}.$$

For the Normal case, it is not difficult to see that

$$\int \pi(\boldsymbol{x}+\tau\boldsymbol{e})d\tau \leq \frac{|\boldsymbol{A}|^{\frac{1}{2}}}{(2\pi)^{(n-1)/2}(\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e})^{1/2}} \exp\left\{ -\frac{1}{2}\frac{\left(\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})\right)^2}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}} \right\}.$$

Maximizing over $\boldsymbol{x}$, i.e. for $\boldsymbol{x}=\boldsymbol{\mu}$, we obtain the direction distribution

$$h_1(\boldsymbol{e}) \propto (\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e})^{-\frac{1}{2}}.$$

Note that one can minimize $I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$ by maximizing $\exp\{-I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})\}$. Then, choosing $h_1(\boldsymbol{e}) \propto (\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e})^{-\frac{1}{2}}$ will naturally choose directions with low $I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$, as can be seen from (3).

To sample from h_1, first note that

$$h_1(\boldsymbol{e}) \propto \int \pi(\boldsymbol{\mu}+\tau\boldsymbol{e})d\tau \propto (\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e})^{-\frac{1}{2}}.$$

If we simulate $\boldsymbol{e}_u$ from a multivariate Normal centred at the origin with precision matrix $\boldsymbol{A}$, and take $\boldsymbol{e} = \boldsymbol{e}_u/\|\boldsymbol{e}_u\|$, it is clear that $\boldsymbol{e} \sim h_1$. This density has the whole sphere $\mathbb{S}^n$ as its support and thus results in an ergodic chain. This justifies the choice of direction distribution made by Calvetti et al. (2008).

We also entertain an alternative direction distribution, as follows. Take the directions $\boldsymbol{e}$ as the

eigenvectors of the precision matrix $\boldsymbol{A}$, so $\boldsymbol{e} \in \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}$. The i-th direction will be selected with probability proportional to λ_i^{-b}, where λ_i is the eigenvalue corresponding to the i-th eigenvector , $i = 1, 2, \ldots, n$, and b is a random variable with distribution $Beta(\alpha, \beta)$. Then

$$h_2(\boldsymbol{e}_i) = k\,(\lambda_i)^{-b}\,,$$

where $k = \left(\sum_{i=1}^{n} \lambda_i^{-b}\right)^{-1}$. It is easy to see that direction $\boldsymbol{e}_n$ corresponding to the lowest eigenvalue λ_n of $\boldsymbol{A}$ is optimal. Note that

$$\begin{aligned}
\min_{\|\boldsymbol{e}\|=1} I_{\boldsymbol{e}}\left(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)}\right) &= \min_{\|\boldsymbol{e}\|=1}\left\{C + \frac{1}{2}\log\left(\boldsymbol{e}^{\mathrm{T}} H(\boldsymbol{x})\boldsymbol{e}\right)\right\} \\
&= C + \frac{1}{2}\log\left(\min_{\|\boldsymbol{e}\|=1}\left\{\boldsymbol{e}^{\mathrm{T}} H(\boldsymbol{x})\boldsymbol{e}\right\}\right) \\
&= C + \frac{1}{2}\log\lambda_n.
\end{aligned}$$

The minimum is reached when $\boldsymbol{e} = \boldsymbol{e}_n$, the eigenvector associated to λ_n.

Since all eigenvectors have a positive probability of being chosen, and these form a basis for $\mathbb{R}^n$, the resulting direction Gibbs sampler will be ergodic. In this case the distribution of $\boldsymbol{e}$ is discrete which may result in computational advantages.

The resulting algorithm works as follows: At $\boldsymbol{X}^{(t)} = \boldsymbol{x}$,

- Propose a direction $\boldsymbol{e}$ from h_i; $i = 1, 2$.
- Propose a length r from a $N(\mu_r, \tau_r)$, with mean $\mu_r = -\dfrac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}$ and precision $\tau_r = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}$.
- Set $\boldsymbol{X}^{(t+1)} = \boldsymbol{x} + r\boldsymbol{e}$.

Note that so far nothing has been achieved since sampling from a multivariate Normal distribution requires several samples from basically the same multivariate Normal. Rather, our aim is to produce an efficient sampler when we have a truncated multivariate Normal, as explained in the following.

4 Truncated multivariate Normal distribution

Suppose we have a multivariate Normal distribution π with precision $(n \times n)$ matrix $\boldsymbol{A}$ and mean vector $\boldsymbol{\mu}$ but truncate the support to $x_i \in (a_i, b_i)$, $-\infty \leq a_i < b_i \leq \infty$, $i = 1, \ldots, n$. The probability density function of this TMVN can be written succinctly as

$$\pi(\boldsymbol{x}) = \frac{\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})\right\}}{\int_{\boldsymbol{a}}^{\boldsymbol{b}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})\right\} d\boldsymbol{x}}.$$

Sampling from this TMVN can be done in a very simple way using a naive rejection algorithm: sampling from the corresponding full multivariate Normal and then keep only the samples that satisfy $x_i \in (a_i, b_i), \forall i \in 1, \ldots, n$. However, this approach will be very inefficient in many cases.

Most of the available methods to sample from a TMVN are based on the Gibbs sampler. Kotecha and Djuric (1999) use the fact that the full conditional distributions of a truncated multivariate Normal are truncated univariate Normal distributions. Hence Gibbs sampling requires simulating from one-dimensional truncated Normal distributions which can be done in a very simple and efficient manner (Robert, 1995; Kotecha and Djuric, 1999). An interesting scheme is presented by Damien and Walker (2001) who use a slice sampler. This is essentially a Gibbs sampler over a space augmented by one variable that turns the full conditional distributions into uniform distributions.

In this section we will show that the algorithm presented in Section 3 may be used to sample from the TMVN distribution with some minor changes.

As mentioned before, we can sample from the MVN by selecting a direction $\boldsymbol{e}$ and the step size r which produces $\boldsymbol{X}^{(t+1)} = \boldsymbol{x} + r\boldsymbol{e}$. Note that, for the TMVN, it is required that $\boldsymbol{a} < \boldsymbol{x} + r\boldsymbol{e} < \boldsymbol{b}$; then $a_i < x_i + re_i < b_i \; \forall i \in \{1, \ldots, n\}$. This puts constraints over r of the form

$$\begin{cases} \dfrac{a_i - x_i}{e_i} < r < \dfrac{b_i - x_i}{e_i} & \forall e_i > 0, \\ \dfrac{b_i - x_i}{e_i} < r < \dfrac{a_i - x_i}{e_i} & \forall e_i < 0. \end{cases}$$

We do not need to worry about the case $e_i = 0$ since it puts no restriction over r, since the i-th coordinate is not being changed. By taking $r \in (c, d)$, with

$$c = \max\left(\left\{\frac{a_i - x_i}{e_i} : e_i > 0\right\} \cup \left\{\frac{b_i - x_i}{e_i} : e_i < 0\right\}\right), \tag{4}$$

$$d = \min\left(\left\{\frac{a_i - x_i}{e_i} : e_i < 0\right\} \cup \left\{\frac{b_i - x_i}{e_i} : e_i > 0\right\}\right), \tag{5}$$

we guarantee that $\boldsymbol{a} < \boldsymbol{X}^{(t+1)} < \boldsymbol{b}$. Since we already know that $r|\boldsymbol{e}, \boldsymbol{x}^{(t)}$ follows a Normal distribution, then the restriction $r \in (c, d)$ implies that $r|\boldsymbol{e}, \boldsymbol{x}^{(t)}, c, d$ follows a univariate truncated Normal (TN) distribution.

The algorithm then proceeds as follows: At $\boldsymbol{X}^{(t)} = \boldsymbol{x}$,

- Propose a direction $\boldsymbol{e}$ from h_i, $i = 1, 2$.
- Propose a length r from a TN(μ_r, τ_r, c, d), with mean $\mu_r = -\dfrac{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{x} - \boldsymbol{\mu})}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}$, precision $\tau_r = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}$, and c and d as in (4) and (5).
- Set $\boldsymbol{X}^{(t+1)} = \boldsymbol{x} + r\boldsymbol{e}$.

We will refer to this algorithm as ODG1 or ODG2 when the direction distribution used is h_1 or h_2, respectively.

More general constraints for the support of the TMVN may also be considered. For example, the algorithm presented here can handle a set of linear inequality constraints that may be written as $\boldsymbol{B}\boldsymbol{X} < \boldsymbol{b}$. This is usually done by transforming the coordinates and changing the problem to $\boldsymbol{a}' < \boldsymbol{X}' < \boldsymbol{b}'$; once the sampling is done, the inverse transformation is performed to return to the original coordinates (Rodriguez-Yam et al., 2004). We will not discuss this case in this paper since it reduces the problem to the one we are already studying.

5 Examples

In this Section we will compare the algorithms ODG1 and ODG2 with the Gibbs samplers presented in Kotecha and Djuric (1999) and Damien and Walker (2001) using several instances of TMVN. We will refer to the last two algorithms as KD and DW, respectively.

Consider a n-dimensional TMVN with mean vector $\boldsymbol{\mu} = (\sqrt{1/n}, \ldots, \sqrt{1/n})$ and precision matrix $\boldsymbol{A}$, for $n = 2, 5, 10, 20$. The support will be restricted to $x_i \geq 0$, i.e. all entries are positive. It is important to note that the truncated support remains unbounded. We will not discuss how to sample from a tightly bounded support since that would represent basically uniform sampling in a complex domain, which is a substantially different sampling problem.

The precision matrix is obtained as $\boldsymbol{A} = \boldsymbol{P}^{\mathrm{T}}\boldsymbol{\Lambda}$. Here, $\boldsymbol{P}$ is a random orthonormal matrix generated by using the QR decomposition of a $n \times n$ matrix of uniform random entries; $\boldsymbol{P}$ represents the orthonormal base of eigenvectors of $\boldsymbol{A}$. Furthermore, $\boldsymbol{\Lambda}$ is a diagonal matrix of the eigenvalues $\lambda_i = \sigma_i^{-2}$. We set the standard deviations in each principal (eigen) direction to $\lambda_i^{-1/2} = \sigma_i = i^{-\alpha/n}$. These represent decreasing standard deviations and are increasingly contrasting as α increases; $\alpha = 0$ results in an uncorrelated distribution. More contrasting standard deviations result in further correlated distributions.

We start by analyzing the case $n = 2$ to present some basic results and then we will show how these results extend to higher dimensions. We run 5000 iterations of each algorithm starting from $\boldsymbol{\mu}$, so there is no need for burn-in. Figure 1 shows the objective distributions for $\alpha = 0, 5, 10, 20$. Black dots correspond to samples from the full bivariate Normal distribution while color dots correspond to the samples from the TMVN obtained using each of the algorithms of interest. As mentioned before, as α increases the correlation leads to more difficult simulation regions. It can be seen that algorithms ODG1 and ODG2 perform well in all cases. Moreover, for $\alpha = 10$, algorithms DW and KD have great difficulties in exploring the whole region of interest after 5000 iterations. For $\alpha = 20$, both KD and DW generate samples concentrated in a small region only.

Figure 2 shows the estimated autocorrelation of each component of the TMVN. Note that for the case $\alpha = 0$, algorithms KD and DW have less autocorrelation between samples than the ODG1 and ODG2, although the autocorrelation levels are still comparable. However, as α increases, autocorrelation of KD and DW is far larger than for ODG1 and ODG2. This is the first indication that ODG algorithms are more efficient in cases where correlation is high.

Since the algorithmic complexity in each case is different, in order to have a fair comparison we want to calculate the average CPU-time needed to obtain a quasi-independent sample. We estimate this by multiplying the average CPU time per iteration (CPUtime) by τ, the number of samples needed to obtain one pseudo-independent sample. We often use the Integrated Autocorrelation Time (IAT) (Geyer, 1992) to estimate τ. However, the IAT is not fully studied for non-reversible chains, which is the case for algorithms KD and DW since they are systematic Gibbs samplers. Instead, we calculate the Effective Sample Size (ESS) Liu (2008) and estimate τ as (m/ESS), where m is the length of the chain. Tables 1, 2, 3, 4 reports this quantities obtained from the previous examples in the two-dimensional case.

For $\alpha = 0$, algorithms KD and DW are more efficient than ODG1 and ODG2 since CPUtime$\times\tau$ is lower. However, as correlation increases, algorithms ODG1 and ODG2 outperform the Gibbs samplers, and in some cases by several orders of magnitude.

If we increase the dimensionality of the TMVN, the results are very similar. Tables 5, 6 and 7 show the values of CPUtime$\times\tau$ for dimensions $n = 5, 10, 20$ respectively. For each n, we choose α such that the ratio λ_n/λ_1, the maximum over the minimum eigenvalue of $\boldsymbol{A}$, which is the condition number of the precision matrix, has the form 2^k, for $k = 0, 5, 10, 20$. For example, if $n = 5$ we get $\alpha = 0, 5.4, 10.8, 21.5$. This makes the examples in higher dimensions comparable with the bivariate case.

For low correlations algorithms KD and DW are more efficient. It is important to notice that their advantage in efficiency does not come from their CPUtime, but from low correlation levels between samples. Note also that both algorithms are systematic Gibbs samplers which makes

CPUtime increase linearly with dimension. Nevertheless, even in the case of low correlations, the ODG algorithms remain fairy comparable in performance.

However, when correlation increases the ODG algorithms become more efficient and outperform the other choices. In this case, efficiency comes from both the low levels of correlation between samples and also from low CPUtime.

6 Discussion

Our optimal direction Gibbs sampler presents interesting characteristics in examples where standard Gibbs samplers are known to be inefficient. One of the main advantages is that CPUtime per iteration does not increase linearly with dimensionality. Also, the performance of our algorithms is remarkable in cases where correlation is very high, which are the difficult and very often interesting cases of study.

The ODG1 algorithm is more flexible since the direction distribution has the whole n-dimensional sphere as support. However, ODG2 has a very good performance also. Note that we could simply choose the parameter b of h_2 as fixed instead of random. However, if within one specific region of the objective distribution there exists one (or few) very contrasting eigenvalues, directions may get trapped in one "corridor" of the density, as it is the case for the TMVN with very high correlation. By taking $b \sim Beta(\alpha, \beta)$ we allow for the chance of selecting eigenvector directions with relative low eigenvalues, thus permitting better mixing and avoiding possible traps. In fact, potentially these parameters could be optimized to obtain even better efficiency.

It is very difficult to parallelize an MCMC algorithm since it is sequential by construction. Nevertheless, parallel computing can be use to accelerate some calculations inside the MCMC steps, for example, those involved in likelihood evaluations (Tibbits et al., 2011). This could be specially relevant in high dimensional examples. There are two specific steps where parallel computing can be used in our algorithm: 1) to evaluate the bounds for length size r presented in

equations (4) and (5); 2) to calculate proposal parameters, which in turn reduces to matrix algebra parallelization (Demmel et al., 2013). With the increasing availability of multi processor machines parallelizing could strongly improve the speed of our algorithm. We leave this for future research.

There is still work to be done regarding the optimal direction distributions. As explained in Section 3.1, an alternative would be to optimize over all possible direction distributions or, for example, optimize the *expected* mutual information restricted to all possible direction distributions that lead to irreducible chains. A challenging function optimization problem arises, defined with a complex restriction, that may lead to very interesting practical and theoretical results. We leave this investigation for future research.

There are several ideas to extend the actual ODG algorithms to non-Normal objective distributions, for example by using a local Normal approximation to the target distribution. We have experimented with this approach in some simple cases and seems to sample correctly from the appropriate target distribution. Even so, this non-Gaussian ODG sampling is still work in progress.

7 Acknowledgements

MSC thanks CONACyT for a PhD scholarship at CIMAT. JAC acknowledges partial financial support from CONACyT (Ciencia Básica) grant 128477-F. JAC and CF thank the Marsden Fund for financial support via contract UOO1015. The authors thank Diego Andrés Pérez Ruiz for his work related to this article as part of his MSc thesis.

References

Breslaw, J. A. (1994). Random sampling from a truncated multivriate normal distribution. *Applied Mathematics Letters 7*(1), 1–6.

Calvetti, D., A. Kuceyeski, and E. Somersalo (2008). Sampling-based analysis of a spatially distributed model for liver metabolism at steady state. *Multiscale Modeling and Simulation 7*(1), 407–431.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory*. John Wiley.

Damien, P. and S. G. Walker (2001). Sampling truncated normal, beta, and gamma densities. *Journal of Computational and Graphical Statistics 10*(2), 206–215.

Demmel, J., A. Fox, S. Kamil, B. Lipshitz, O. Schwartz, and O. Spillinger (2013). Communication-optimal parallel recursive rectangular matrix multiplication. In *27th International Symposium on Parallel and Distributed Processing (IPDPS)*, pp. 261–272. IEEE.

Gelfand, A. E. and A. F. M. Smith (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association 85*(410), 398–409.

Gelfand, A. E., A. F. M. Smith, and T.-M. Lee (1992). Bayesian analysis of constrained parameter and truncated data problems using Gibbs sampling. *Journal of the American Statistical Association 87*(418), 523–532.

Geyer, C. J. (1992). Practical Markov chain Monte Carlo. *Statistical Science 7*(4), 473–511.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*(1), 97–109.

Kaipio, J. and E. Somersalo (2004). *Statistical and Computational Inverse Problems*. Springer Verlag.

Kaufman, D. E. and R. L. Smith (1998). Direction choice for accelerated convergence in hit-and-run sampling. *Operations Research 46*(1), 84–95.

Kotecha, J. H. and P. M. Djuric (1999). Gibbs sampling approach for generation of truncated multivariate Gaussian random variables. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume 3, pp. 1757–1760.

Liu, J. S. (2008). *Monte Carlo Strategies in Scientific Computing* (2nd ed.). Springer Verlag.

Peskun, P. (1973). Optimum monte-carlo sampling using markov chians. *Biometrika 60*(3), 607–612.

Robert, C. P. (1995). Simulation of truncated normal variables. *Statistics and Computing 5*(2), 121–125.

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods* (2nd ed.). Springer Verlag.

Rodriguez-Yam, G., R. A. Davis, and L. L. Scharf (2004). Efficient Gibbs sampling of truncated multivariate normal with application to constrained linear regression. Technical report, Technical report, Colorado State University.

Tibbits, M. M., M. Haran, and J. C. Liechty (2011). Parallel multivariate slice sampling. *Statistics and Computing 21*(3), 415–430.

Yu, J. and G. Tian (2011). Efficient algorithms for generating truncated multivariate normal distributions. *Acta Mathematicae Applicatae Sinica, English Series 27*(4), 601–612.

A Derivation of the mutual information

As mentioned in Section 2, the mutual information between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is given by

$$I(\boldsymbol{Y}, \boldsymbol{X}) = \int \int f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}) \log \frac{f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x})}{f_{\boldsymbol{Y}}(\boldsymbol{y}) f_{\boldsymbol{X}}(\boldsymbol{x})} d\boldsymbol{x} d\boldsymbol{y}.$$

We want to calculate $I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)}, \boldsymbol{X}^{(t)})$, this is the mutual information between $\boldsymbol{Y} = \boldsymbol{X}^{(t+1)} = \boldsymbol{X}^{(t)} + r\boldsymbol{e}$ and $\boldsymbol{X} = \boldsymbol{X}^{(t)}$ restricted to direction $\boldsymbol{e}$. We suppose $\boldsymbol{X} \sim \pi$, where π is a multivariate Normal distribution with mean vector $\boldsymbol{\mu}$ and precision matrix $\boldsymbol{A}$.

In our case $f_{\boldsymbol{Y}}(\boldsymbol{y}) = \pi(\boldsymbol{y})$ and $f_{\boldsymbol{Y},\boldsymbol{X}}(\boldsymbol{y}, \boldsymbol{x}) = \pi(\boldsymbol{x}) K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y})$, with

$$K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e}}{2\pi} \right)^{\frac{1}{2}} \exp \left\{ -\frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e}}{2} \left(\boldsymbol{e}^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{x}) + \frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e}} \right)^2 \right\} 1(\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{x}) \boldsymbol{e}).$$

Therefore, the mutual information above can be calculated as

$$I(\boldsymbol{Y}, \boldsymbol{X}) = \int \int \pi(\boldsymbol{x}) K(\boldsymbol{x}, \boldsymbol{y}) \log \frac{K(\boldsymbol{x}, \boldsymbol{y})}{\pi(\boldsymbol{y})} d\boldsymbol{y} d\boldsymbol{x}. \tag{6}$$

Consider first the logarithmic term in (6), this is

$$\log \frac{K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y})}{\pi(\boldsymbol{y})} = C + \frac{1}{2} \log \boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e} - \frac{1}{2} \left[Q_1(\boldsymbol{e}, \boldsymbol{x}, \boldsymbol{y}) - Q_2(\boldsymbol{y}) \right],$$

where

$$\begin{aligned} C &= \frac{n-1}{2} \log 2\pi - \frac{1}{2} \log |\boldsymbol{A}|, \\ Q_1(\boldsymbol{e}, \boldsymbol{x}, \boldsymbol{y}) &= \boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e} \left[\boldsymbol{e}^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{x}) + \frac{\boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{v}}{\boldsymbol{e}' \boldsymbol{A} \boldsymbol{e}} \right]^2, \\ Q_2(\boldsymbol{y}) &= (\boldsymbol{y} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{A} (\boldsymbol{y} - \boldsymbol{\mu}). \end{aligned}$$

From this we see that

$$\int \log \frac{K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y})}{\pi(\boldsymbol{y})} K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = C - \frac{1}{2} + \frac{1}{2} \log \boldsymbol{e}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{e} + \frac{1}{2} \int Q_2(\boldsymbol{y}) K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}$$

since $\int Q_1(\boldsymbol{e}, \boldsymbol{x}, \boldsymbol{y}) K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = 1$. The integral $\int Q_2(\boldsymbol{y}) K_{\boldsymbol{e}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y}$ may be calculated by transform-

ing back to r since

$$\int Q_2(\boldsymbol{y})K_{\boldsymbol{e}}(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} = \int (r\boldsymbol{e}-\boldsymbol{v})^{\mathrm{T}}\boldsymbol{A}(r\boldsymbol{e}-\boldsymbol{v})g_{\boldsymbol{e}}(r)dr.$$

After some algebra one sees that

$$\int Q_2(\boldsymbol{y})K_{\boldsymbol{e}}(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} = 1 - \frac{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}} + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}.$$

Then

$$\int \log\frac{K_{\boldsymbol{e}}(\boldsymbol{x},\boldsymbol{y})}{\pi(\boldsymbol{y})}K_{\boldsymbol{e}}(\boldsymbol{x},\boldsymbol{y})d\boldsymbol{y} = C + \frac{1}{2}\log\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e} - \frac{1}{2}\frac{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}} + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}.$$

We need now to integrate with respect to $\pi(d\boldsymbol{x})$. We note that $\int \boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}\pi(\boldsymbol{x})d\boldsymbol{x} = n$. Moreover, the expected value of a quadratic form is

$$E[\boldsymbol{z}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{z}] = \mathrm{tr}(\boldsymbol{R}\boldsymbol{\Sigma}) + \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{\mu},$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and the variance-covariance matrix of $\boldsymbol{z}$. Letting $\boldsymbol{R} = \frac{\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}$ and since $E_\pi(\boldsymbol{v}) = \boldsymbol{0}$ we obtain

$$\begin{aligned}
E_\pi\left[\boldsymbol{v}'\frac{\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}'\boldsymbol{A}}{\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e}}\boldsymbol{v}\right] &= \int \frac{\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}}{\boldsymbol{e}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}}\pi(\boldsymbol{x})d(\boldsymbol{x}) \\
&= \frac{1}{\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e}}\mathrm{tr}(\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}'\boldsymbol{A}\boldsymbol{A}^{-1}) \\
&= \frac{1}{\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e}}\mathrm{tr}(\boldsymbol{A}\boldsymbol{e}\boldsymbol{e}') \\
&= \frac{1}{\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e}}\mathrm{tr}(\boldsymbol{e}\boldsymbol{A}\boldsymbol{e}') \\
&= 1.
\end{aligned}$$

Therefore

$$\begin{aligned}
I_{\boldsymbol{e}}(\boldsymbol{X}^{(t+1)},\boldsymbol{X}^{(t)}) &= C + n - \frac{1}{2} + \frac{1}{2}\log\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e} \\
&= \quad C_1 + \frac{1}{2}\log\boldsymbol{e}'\boldsymbol{A}\boldsymbol{e},
\end{aligned}$$

where $C_1 = C + n - \frac{1}{2}$.

	ODG1	ODG2	KD	DW
ESS	1799.7	1937.5	5082.3	3587.6
τ	2.8	2.6	1.0	1.4
CPUtime	0.00045	0.00032	0.00059	0.00013
CPUtime$\times\tau$	0.00125	0.00083	0.00058	**0.00017**

Table 1: Effective Sample Size (ESS) for $\alpha = 0$; number of samples needed to get one pseudo-independent sample (τ) and CPU time per iteration (CPUtime). Each quantity is the average of 30 chains of 5000 iterations. CPUtime$\times\tau$ represents the average time to obtain one pseudo-independent sample. Numbers in bold indicate the most efficient algorithm.

	ODG1	ODG2	KD	DW
ESS	2100.1	1908.6	1440.9	1301.4
τ	2.4	2.6	3.5	3.8
CPUtime	0.00045	0.00033	0.00059	0.00012
CPUtime$\times\tau$	0.00107	0.00087	0.00204	**0.00048**

Table 2: Effective Sample Size (ESS) for $\alpha = 5$; number of samples needed to get one pseudo-independent sample (τ) and CPU time per iteration (CPUtime). Each quantity is the average of 30 chains of 5000 iterations. CPUtime$\times\tau$ represents the average time to get one pseudo-independent sample. Numbers in bold indicate the most efficient algorithm.

	ODG1	ODG2	KD	DW
ESS	2268.3	1932.1	55.0	49.8
τ	2.2	2.6	90.1	100.5
CPUtime	0.00044	0.00033	0.00057	0.00012
CPUtime$\times\tau$	0.00094	**0.00084**	0.05201	0.01217

Table 3: Effective Sample Size (ESS) for $\alpha = 10$; number of samples needed to get one pseudo-independent sample (τ) and CPU time per iteration (CPUtime). Each quantity is the average of 30 chains of 5000 iterations. CPUtime$\times\tau$ represents the average time to get one pseudo-independent sample. Numbers in bold indicate the most efficient algorithm.

	ODG1	ODG2	KD	DW
ESS	22351.9	1946.1	9.3	10.7
τ	2.2	2.6	538.8	467.7
CPUtime	0.00043	0.00032	0.00056	0.00012
CPUtime$\times\tau$	0.00094	**0.00082**	0.30369	0.05584

Table 4: Effective Sample Size (ESS) for $\alpha = 20$; number of samples needed to get one pseudo-independent sample (τ) and CPU time per iteration (CPUtime). Each quantity is the average of 30 chains of 5000 iterations. CPUtime$\times\tau$ represents the average time to get one pseudo-independent sample. Numbers in bold indicate the most efficient algorithm.

	ODG1	ODG2	KD	DW
$\alpha = 0$	0.00465	0.00671	0.00145	**0.00046**
$\alpha = 5.4$	0.00381	0.00313	0.00305	**0.00071**
$\alpha = 10.8$	0.00341	**0.00271**	0.03727	0.00789
$\alpha = 21.5$	**0.00176**	0.00261	0.50378	0.13555

Table 5: Average time to get one pseudo-independent sample (CPUtime$\times\tau$) for a five-dimensional TMVN. Each quantity is the average of 30 chains of 5000 iterations. Numbers in bold indicate the most efficient algorithm for a specific value of α.

	ODG1	ODG2	KD	DW
$\alpha = 0$	0.0142	0.0129	0.0031	**0.0011**
$\alpha = 7.5$	0.0112	0.0084	0.0049	**0.0013**
$\alpha = 15.1$	**0.0058**	0.0062	0.0776	0.0178
$\alpha = 30.1$	**0.0036**	0.0061	1.7291	0.3784

Table 6: Average time to get one pseudo-independent sample (CPUtime$\times\tau$) for a ten-dimensional TMVN. Each quantity is the average of 30 chains of 5000 iterations. Numbers in bold indicate the most efficient algorithm for a specific value of α.

	ODG1	ODG2	KD	DW
$\alpha = 0$	0.0452	0.0462	0.0059	**0.0027**
$\alpha = 11.6$	0.0413	0.0245	0.0077	**0.0028**
$\alpha = 23.1$	0.0259	**0.0151**	0.0569	0.0166
$\alpha =$46.3	0.0142	**0.0127**	3.5618	0.9832

Table 7: Average time to get one pseudo-independent sample (CPUtime$\times\tau$) for a 20-dimensional TMVN. Each quantity is the average of 30 chains of 5000 iterations. Numbers in bold indicate the most efficient algorithm for a specific value of α.

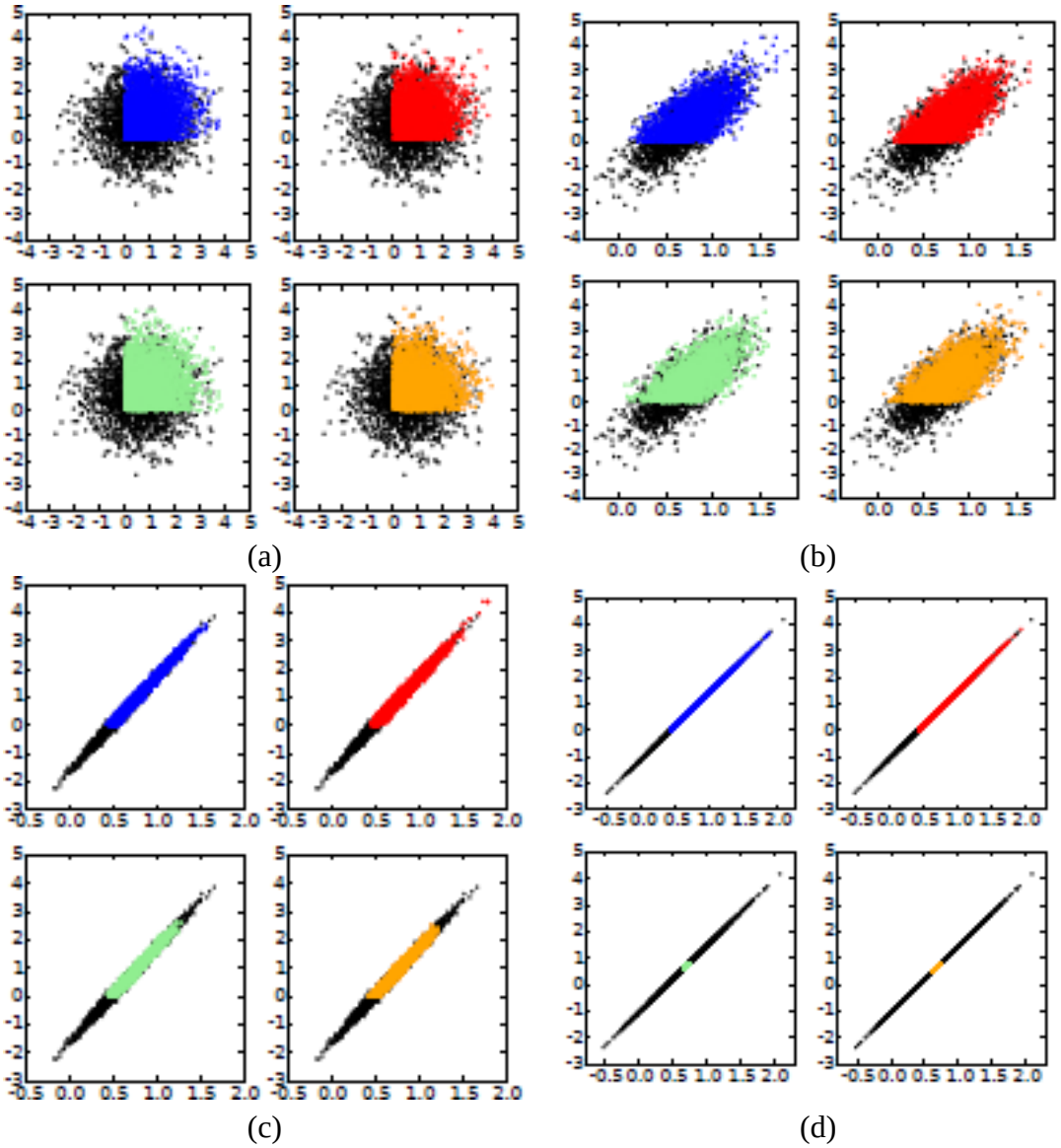

Figure 1: Samples of the full bivariate Normal distribution (black dots) and the TMVN with algorithms ODG1 (blue), ODG2 (red), KD (green) and DW (orange) after 5000 iterations for a) $\alpha = 0$, b) $\alpha = 5$, c) $\alpha = 10$, and d) $\alpha = 20$.

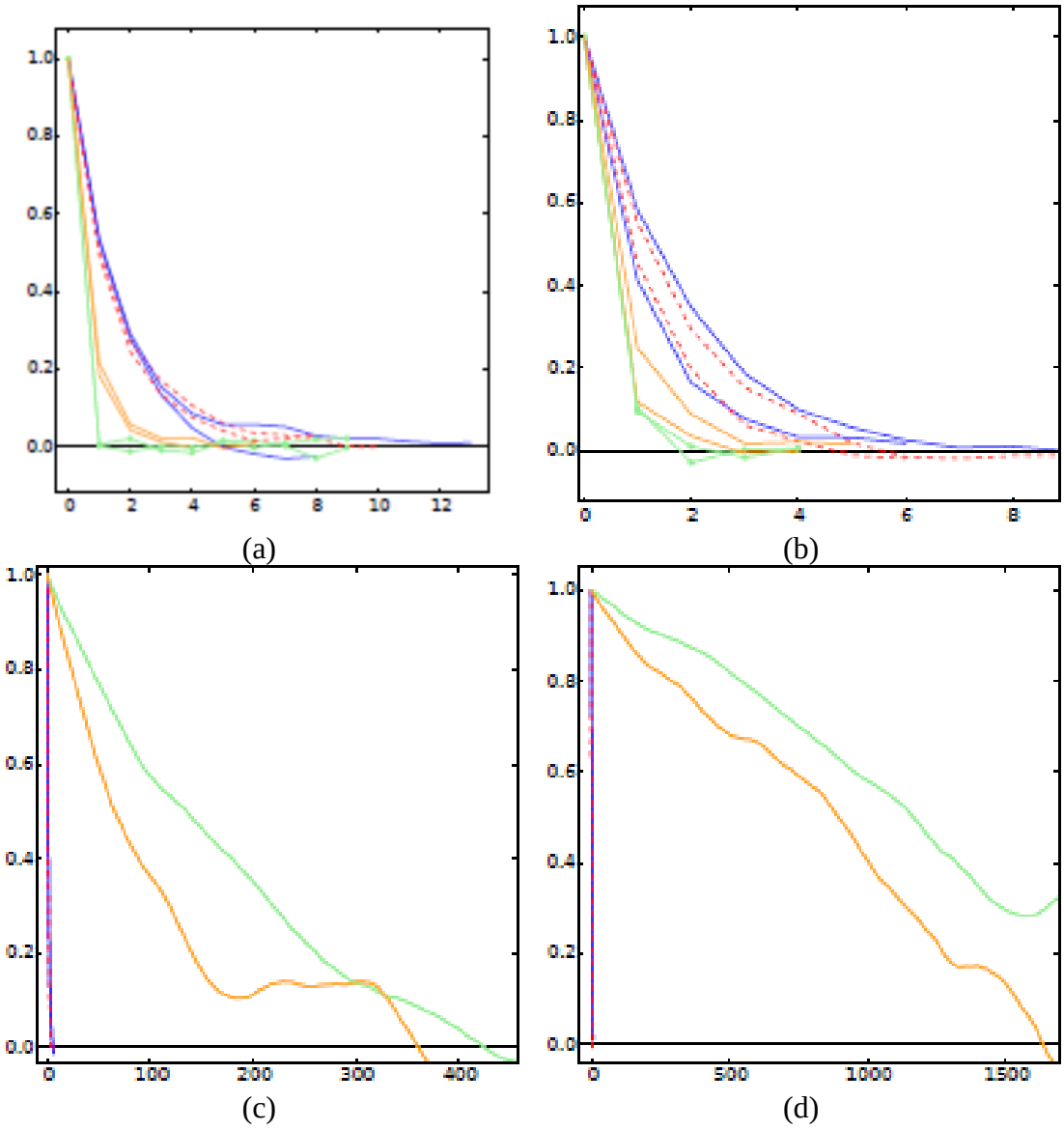

Figure 2: Estimated autocorrelation for the chains generated with algorithms ODG1 (blue), ODG2 (red), KD (green) and DW (orange) for a) $\alpha = 0$, b) $\alpha = 5$, c) $\alpha = 10$, and d) $\alpha = 20$. In examples c) and d), the autocorrelation for ODG1 and ODG2 is almost 0 after lag five.