SIAM REVIEW
Vol. 64, No. 1, pp. 3–56

© 2022 Society for Industrial and Applied Mathematics

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

High-Dimensional Gaussian Sampling: A Review and a Unifying Approach Based on a Stochastic Proximal Point Algorithm*

Maxime Vono†
Nicolas Dobigeon‡
Pierre Chainais§

Abstract. Efficient sampling from a high-dimensional Gaussian distribution is an old but high-stakes issue. Vanilla Cholesky samplers imply a computational cost and memory requirements that can rapidly become prohibitive in high dimensions. To tackle these issues, multiple methods have been proposed from different communities ranging from iterative numerical linear algebra to Markov chain Monte Carlo (MCMC) approaches. Surprisingly, no complete review and comparison of these methods has been conducted. This paper aims to review all these approaches by pointing out their differences, close relations, benefits, and limitations. In addition to reviewing the state of the art, this paper proposes a unifying Gaussian simulation framework by deriving a stochastic counterpart of the celebrated proximal point algorithm in optimization. This framework offers a novel and unifying revisiting of most of the existing MCMC approaches while also extending them. Guidelines to choosing the appropriate Gaussian simulation method for a given sampling problem in high dimensions are proposed and illustrated with numerical examples.

Key words. Gaussian distribution, high-dimensional sampling, linear system, Markov chain Monte Carlo, proximal point algorithm

AMS subject classifications. 65C10, 68U20, 62H12

DOI. 10.1137/20M1371026

Contents

*Received by the editors October 2, 2020; accepted for publication (in revised form) August 9, 2021; published electronically February 3, 2022.
https://doi.org/10.1137/20M1371026
Funding: This work was partially supported by the ANR-3IA Artificial and Natural Intelligence Toulouse Institute (ANITI) under grant agreement ANITI ANR-19-PI3A-0004, by the ANR project "Chaire IA Sherlock," ANR-20-CHIA-0031-01, as well as by national support within *Programme d'investissements d'avenir*, ANR-16-IDEX-0004 ULNE. The work of the third author was supported by his 3IA Chair Sherlock funded by the ANR, under grant agreement ANR-20-CHIA-0031-01, Centrale Lille, the I-Site Lille-Nord-Europe, and the region Hauts-de-France.

†Lagrange Mathematics and Computing Research Center, Huawei, 75007 Paris, France (maxime.vono@gmail.com).
‡University of Toulouse, IRIT/INP-ENSEEIHT, 31071 Toulouse, France, and Institut Universitaire de France (IUF), France (nicolas.dobigeon@enseeiht.fr).
§University of Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France (pierre.chainais@centralelille.fr).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

1. Introduction. If there was only one continuous probability distribution that we needed to know, it would certainly be the Gaussian (also known as *normal*) distribution. Many nice properties of the Gaussian distribution can be listed, such as its infinite divisibility, maximum entropy property, or its description based on the use of the first two cumulants only (mean and variance). However, its popularity and ubiquity result from two essential properties, namely, the central limit theorem and the statistical interpretation of ordinary least squares, which often motivate its use to describe random noises or residual errors in various applications (e.g., inverse problems in signal and image processing). The first property originates from gambling theory. The binomial distribution, which models the probabilities of successes and failures after a given number of trials, was approximated by a Gaussian distribution in the seminal work by de Moivre [73]. This famous approximation is a specific instance of the central limit theorem, which states that the sum of a sufficiently large number of independent and identically distributed (i.i.d.) random variables with finite variance converges in distribution toward a Gaussian random variable. Capitalizing on this theorem, a lot of complex random events have been approximated using the Gaussian distribution, sometimes called the *bell curve*. Another well-known use of the Gaussian distribution has been in the search for an error distribution in empirical sciences. For instance, since the end of the 16th century, astronomers have been interested in data summaries to describe their observations. They found that the estimate defined by the arithmetic mean of the observations was related to the resolution of a least mean square problem under the assumption of Gaussian measurement errors [46]. The assumption of Gaussian noise has now become so common that it is sometimes implicit in many applications.

Motivated by all these features, the Gaussian distribution is omnipresent in problems far beyond the statistics community. In statistical machine learning and signal processing, Gaussian posterior distributions commonly appear when hierarchical Bayesian models are derived [37, 67, 80, 83], in particular, when the exponential family is involved. As archetypal examples, models based on Gaussian Markov random fields or conditional autoregressions assume that parameters of interest (associated with observations) come from a joint Gaussian distribution with a structured covariance matrix reflecting their interactions [92]. Such models have found applications in spatial statistics [12, 23], image analysis [36, 56], graphical structures [39], and semiparametric regression and splines [29]. We can also cite the discretization schemes of stochastic differential equations involving Brownian motion, which led to a Gaussian sampling step [27, 89, 106], texture synthesis [34], and time series prediction [16]. Indeed, the Gaussian distribution is also intimately connected to diffusion processes and statistical physics.

When the dimension of the problem is small or moderate, sampling from this distribution is an old solved problem that raises no particular difficulty. In high-

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

dimensional settings this multivariate sampling task can become computationally demanding, which may prevent us from using statistically sound methods for real-world applications. Therefore, even recently, a host of works have focused on the derivation of *efficient* high-dimensional Gaussian sampling methods. Before summarizing our contributions and main results, in what follows, we discuss what we mean by *complexity* and *efficiency*, in light of the most common sampling technique, i.e., exploiting the Cholesky factorization.

Computational and Storage Complexities: Notation. In what follows, we will use the mathematical notation $\Theta(\cdot)$ and $\mathcal{O}(\cdot)$ to describe the complexities of the computation and the storage required by the sampling algorithms. We recall that $f(d) = \mathcal{O}(g(d))$ if there exists $c > 0$ such that $f(d) \leq cg(d)$ when $d \to \infty$. We use $f(d) = \Theta(g(d))$ if there exist $c_1, c_2 > 0$ such that $c_1 g(d) \leq f(d) \leq c_2 g(d)$ when $d \to \infty$.

Table 1 *Vanilla Cholesky sampling. Computational and storage requirements to produce one sample from an arbitrary d-dimensional Gaussian distribution.*

d	Vanilla Cholesky sampler	
	$\Theta(d^3)$ flops	$\Theta(d^2)$ memory requirement
10^3	3.34×10^8	4 MB
10^4	3.33×10^{11}	0.4 GB
10^5	3.33×10^{14}	40 GB
10^6	3.33×10^{17}	4 TB

Algorithmic Efficiency: Definition. To sample from a given d-dimensional Gaussian distribution, the most common sampling algorithm is based on the Cholesky factorization [91]. Let us recall that the Cholesky factorization of a symmetric positive-definite matrix $\mathbf{Q} \in \mathbb{R}^{d \times d}$ is a decomposition of the form [92, section 2.4]

$$\mathbf{Q} = \mathbf{C}\mathbf{C}^\top, \tag{1.1}$$

where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is a lower triangular matrix with real and positive diagonal entries. The computation of the Cholesky factor $\mathbf{C}$ for dense matrices requires $\Theta(d^3)$ floating point operations (flops), that is, arithmetic operations such as additions, subtractions, multiplications, or divisions [41]; see also subsection 3.1.1 for details. In addition, the Cholesky factor, which involves at most $d(d+1)/2$ nonzero entries, must be stored. In the general case, this implies a global memory requirement of $\Theta(d^2)$. In high-dimensional settings ($d \gg 1$), both these numerical complexity and storage requirements rapidly become prohibitive for standard computers. Table 1 illustrates this claim by listing the number of flops (using 64-bit numbers also called *double precision*) and storage space required by the vanilla Cholesky sampler in high dimensions. Note that for $d \geq 10^5$, which is classical in image processing problems, for instance, the memory requirement of the Cholesky sampler exceeds the memory capacity of today's standard laptops. To mitigate these computational issues, much work has focused on the derivation of surrogate high-dimensional Gaussian sampling methods. Compared to Cholesky sampling, these surrogate samplers involve additional sources of approximation in finite-time sampling and as such intend to trade off *computation* and *storage* requirements against sampling *accuracy*. Throughout this review, we will say that a Gaussian sampling procedure is "efficient" if, in order to produce a sample with reasonable accuracy, the number of flops and memory required

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

are significantly lower than that of the Cholesky sampler. For the sake of clarity, at the end of each section presenting an existing Gaussian sampler, we will highlight its theoretical relative efficiency with respect to (w.r.t.) vanilla Cholesky sampling with a dedicated paragraph. As a typical example, some approaches reviewed in this paper will only require $\mathcal{O}(d^2)$ flops and $\Theta(d)$ storage space, which are lower than Cholesky complexities by one power of d.

Contributions. To the authors' knowledge, no systematic comparison of existing Gaussian sampling approaches is available in the literature. This is probably due to the huge number of contributions related to this task from distinct communities. Hence, it is not always clear which method is best suited to a given Gaussian sampling task in high dimensions, and what the main similarities and differences among them are. To this end, this paper both reviews the main sampling techniques dedicated to an arbitrary high-dimensional Gaussian distribution (see Table 7 for a synthetic overview) and derives general guidelines for practitioners in determining the appropriate sampling approach when Cholesky sampling is not possible (see Figure 12 for a schematic overview). Beyond that review, we propose to put most of the state-of-the-art Markov chain Monte Carlo (MCMC) methods under a common umbrella by deriving a unifying Gaussian sampling framework.

Main Results. Our main results are summarized below.

- At the expense of some approximation, we show that state-of-the-art Gaussian sampling algorithms are indeed more efficient than Cholesky sampling in high dimensions. Their computational complexity, memory requirements, and accuracy are summarized in Table 7.
- Among existing Gaussian samplers, some provide i.i.d. samples while others (e.g., MCMC-based ones) produce correlated samples. Interestingly, we show in section 6 for several experiments that MCMC approaches might perform better than samplers providing i.i.d. samples. This relative efficiency is particularly important in the case where many samples are required.
- In section 5, we show that most of the existing MCMC approaches can be seen as special instances of a unifying framework that is a stochastic counterpart of the proximal point algorithm (PPA) [90]. The proposed Gaussian sampling framework also allows us to both extend existing algorithms by proposing new sampling alternatives and to draw a one-to-one equivalence among MCMC samplers proposed by distinct communities such as those based on matrix splitting [32, 54] and data augmentation [66, 67].
- Finally, we show that the choice of an appropriate Gaussian sampling approach is a subtle compromise between several issues such as the need to obtain accurate samples or the existence of a natural decomposition of the precision matrix. We provide in Figure 12 simple guidelines for practitioners in the form of a decision tree to help them choose the appropriate Gaussian sampler based on these parameters.

Structure of the Paper. This paper is structured as follows. In section 2, we present the multivariate Gaussian sampling problem along with its simple and more complicated instances. In particular, we will list and illustrate the main difficulties associated with sampling from a high-dimensional Gaussian distribution with an arbitrary covariance matrix. These difficulties have motivated many works that propose surrogate sampling approaches. The latter are presented in sections 3 and 4. More precisely, section 3 presents Gaussian sampling schemes which have been derived by

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

adapting ideas from numerical linear algebra. In section 4, we review another class of sampling techniques, namely, MCMC approaches, which build a Markov chain admitting the Gaussian distribution of interest (or a close approximation) as a stationary distribution. In section 5, we propose to shed new light on most of these MCMC methods by embedding them into a unifying framework based on a stochastic version of the PPA. In section 6, we illustrate and compare the reviewed approaches w.r.t. archetypal experimental scenarios. Finally, section 7 makes concluding remarks. A guide to the notation used in this paper and technical details associated to each section are given in the appendices.

Software. All the methods reviewed in this paper have been implemented and made available in a companion package written in Python called `PyGauss`. In addition, `PyGauss` contains a Jupyter notebook which allows the reader to reproduce all the figures and numerical results in this paper. This package and its associated documentation can be found online.[1]

2. Gaussian Sampling: Problem, Instances, and Issues.

This section highlights the Gaussian sampling problem considered, its already-surveyed special instances, and its main issues. By recalling these specific instances, this section will also define the limits of this paper in terms of the reviewed approaches. Note that for the sake of simplicity, we shall abusively use the same notation for both a random variable and its realization.

2.1. Definitions and Notation.

Throughout this review, we will use capital letters (e.g., Π) to denote a probability distribution and corresponding small letters (e.g., π) to refer to its associated probability density function (p.d.f.). We address the problem of sampling from a d-dimensional Gaussian distribution $\Pi \triangleq \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where d may be large. Assume throughout that the covariance matrix $\boldsymbol{\Sigma}$ is positive definite, that is, for all $\boldsymbol{\theta} \in \mathbb{R}^d \setminus \{\mathbf{0}_d\}$, $\boldsymbol{\theta}^\top \boldsymbol{\Sigma} \boldsymbol{\theta} > 0$. Hence, its inverse $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$, called the *precision* matrix, exists and is also positive definite. When $\boldsymbol{\Sigma}$ is not of full rank, the distribution Π is said to be degenerate and does not admit a density w.r.t. the d-dimensional Lebesgue measure. Its p.d.f. with respect to the d-dimensional Lebesgue measure, for all $\boldsymbol{\theta} \in \mathbb{R}^d$, can be written as

$$\pi(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu})\right), \tag{2.1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$, respectively, stand for the mean vector and the covariance matrix of the considered Gaussian distribution.

For some approaches and applications, working with the precision $\mathbf{Q}$ rather than with the covariance $\boldsymbol{\Sigma}$ will be more convenient (e.g., for conditional autoregressive models or hierarchical Bayesian models; see also section 4). In this paper, we choose to present existing approaches by working directly with $\mathbf{Q}$ for the sake of simplicity. When $\mathbf{Q}$ is unknown but $\boldsymbol{\Sigma}$ is available instead, simple and straightforward algebraic manipulations can be used to implement the same approaches without increasing their computational complexity. Sampling from $\mathcal{N}\left(\boldsymbol{\mu}, \mathbf{Q}^{-1}\right)$ raises several important issues which are mainly related to the structure of $\mathbf{Q}$. In the following paragraphs, we will detail some special instances of (2.1) and well-known associated sampling strategies before focusing on the general Gaussian sampling problem considered in this paper.

[1] http://github.com/mvono/PyGauss.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

2.2. Usual Special Instances. For completeness, this subsection recalls special cases of Gaussian sampling tasks that will not be detailed later but are common building blocks. Instead, we point out appropriate references for the interested reader. These special instances include basic univariate sampling and the scenarios where $\mathbf{Q}$ is (i) a diagonal matrix, (ii) a band matrix, or (iii) a circulant Toeplitz matrix. Again, with basic algebraic manipulations, the same samplers can be used when $\mathbf{\Sigma}$ has one of these specific structures.

2.2.1. Univariate Gaussian Sampling. The most simple Gaussian sampling problem boils down to drawing univariate Gaussian random variables with mean $\mu \in \mathbb{R}$ and precision $q > 0$. Generating the latter quickly and with high accuracy has been the topic of much research work in the last 70 years. Such methods can be loosely speaking divided into four groups, namely, (i) cumulative density function (c.d.f.) inversion, (ii) transformation, (iii) rejection, and (iv) recursive methods; they are now well documented. Interested readers are invited to refer to the comprehensive overview in [101] for more details. For instance, Algorithm 2.1 details the well-known Box–Muller method, which transforms a pair of independent uniform random variables into a pair of Gaussian random variables by exploiting the radial symmetry of the two-dimensional normal distribution.

Algorithm 2.1 Box–Muller sampler

1: Draw $u_1,\, u_2 \overset{\text{i.i.d.}}{\sim} \mathcal{U}((0,1])$.
2: Set $\tilde{u}_1 = \sqrt{-2\log(u_1)}$.
3: Set $\tilde{u}_2 = 2\pi u_2$.
4: **return** $(\theta_1, \theta_2) = \left(\mu + \frac{\tilde{u}_1}{\sqrt{q}}\sin(\tilde{u}_2), \mu + \frac{\tilde{u}_1}{\sqrt{q}}\cos(\tilde{u}_2)\right)$.

2.2.2. Multivariate Gaussian Sampling with Diagonal Precision Matrix. Let us extend the previous sampling problem and now assume that one wants to generate a d-dimensional Gaussian vector $\boldsymbol{\theta}$ with mean $\boldsymbol{\mu}$ and diagonal precision matrix $\mathbf{Q} = \operatorname{diag}(q_1, \ldots, q_d)$. The d components of $\boldsymbol{\theta}$ being independent, this problem is as simple as the univariate one since we can sample the d components in parallel independently. A pseudocode of the corresponding sampling algorithm is given in Algorithm 2.2.

Algorithm 2.2 Sampler when $\mathbf{Q}$ is a diagonal matrix

1: **for** $i \in [d]$ **do** ▷ In some programming languages, this loop can be vectorized.
2: Draw $\theta_i \sim \mathcal{N}\left(\mu_i, 1/q_i\right)$.
3: **end for**
4: **return** $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^\top$.

Algorithmic Efficiency. By using, for instance, Algorithm 2.1 for step 2, Algorithm 2.2 admits a computational complexity of $\Theta(d)$ and a storage capacity of $\Theta(d)$. In this specific scenario, these requirements are significantly less than those of vanilla Cholesky sampling, whose complexities are recalled in Table 1.

When $\mathbf{Q}$ is not diagonal, we can no longer sample the d components of $\boldsymbol{\theta}$ independently. Thus, more sophisticated sampling methods must be used. For well-structured matrices $\mathbf{Q}$, we show in the following sections that it is possible to draw the random vector of interest more efficiently than with vanilla Cholesky sampling.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

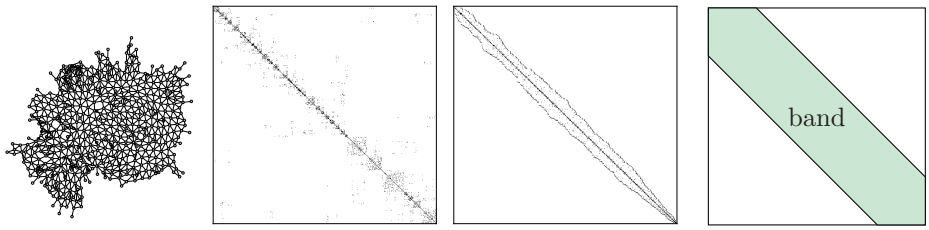


Fig. 1 *From left to right: Example of an undirected graph defined on the* 544 *regions of Germany where those sharing a common border are considered as neighbors, its associated precision matrix* $\mathbf{Q}$ *(bandwidth* $b = 522$*), its reordered precision matrix* $\mathbf{PQP}^\top$ *(*$b = 43$*) where* $\mathbf{P}$ *is a permutation matrix, and a drawing of a band matrix. For the three matrices, the white entries are equal to zero.*

2.2.3. Multivariate Gaussian Sampling with Sparse or Band Matrix $\mathbf{Q}$. A lot of standard Gaussian sampling approaches leverage the sparsity of the matrix $\mathbf{Q}$. Sparse precision matrices appear, for instance, when Gaussian Markov random fields (GMRFs) are considered, as illustrated in Figure 1. In this figure, German regions are represented graphically, where each edge between two regions represents a common border. These edges can then be described by an adjacency matrix which plays the role of the precision matrix $\mathbf{Q}$ of a GMRF. Since there are few neighbors for each region, $\mathbf{Q}$ is symmetric and sparse. By permuting the rows and columns of $\mathbf{Q}$, one can build a so-called *band matrix* with minimal bandwidth b, where b is the smallest integer $b < d$ such that $Q_{ij} = 0\ \forall i > j + b$ [91]. Note that band matrices also appear naturally in specific applications, e.g., when the latter involve finite impulse response linear filters [50]. Problems with such structured (sparse or band) matrices have been extensively studied in the literature and this paper will not cover them explicitly. We provide in Algorithm 2.3 the main steps to obtaining a Gaussian vector $\boldsymbol{\theta}$ from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ in this scenario and refer the interested reader to [92] for more details.

Algorithmic Efficiency. Algorithm 2.3 is a specific instance of Cholesky sampling for band precision matrices. In this specific scenario, Algorithm 2.3 admits a computational complexity of $\Theta(b^2 d)$ flops and a memory space of $\Theta(bd)$, since the band matrix $\mathbf{Q}$ can be stored in a so-called "$\mathbf{Q}$.array" [41, section 1.2.5]. When $b \ll d$, one observes that these computational and storage requirements are smaller than those of vanilla Cholesky sampling by an order of magnitude w.r.t. d. Similar computational savings can be obtained in the sparse case [92].

Algorithm 2.3 Sampler when $\mathbf{Q}$ is a band matrix

1: Set $\mathbf{C} = \text{chol}(\mathbf{Q})$. ▷ Build the Cholesky factor $\mathbf{C}$ of $\mathbf{Q}$; see [92, section 2.4].
2: Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
3: **for** $i \in [d]$ **do** ▷ Solve $\mathbf{C}^\top \mathbf{w} = \mathbf{z}$ w.r.t. $\mathbf{w}$ by backward substitution.
4: Set $j = d - i + 1$.
5: Set $m_1 = \min\{j + b, d\}$.
6: Set $w_j = \dfrac{1}{C_{jj}} \left(z_j - \displaystyle\sum_{k=j+1}^{m_1} C_{kj} w_k \right)$.
7: **end for**
8: **return** $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{w}$.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

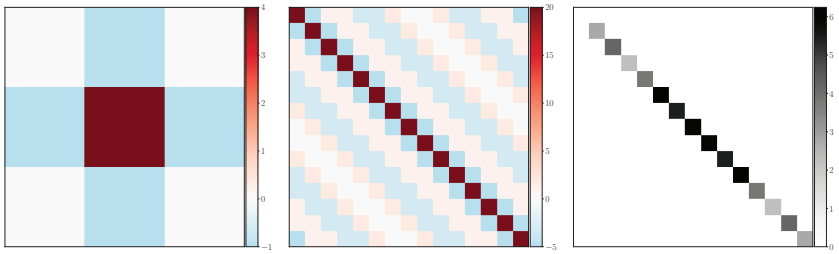


Fig. 2 *From left to right: Example of a* 3×3 *Laplacian filter, the associated circulant precision matrix* $\mathbf{Q} = \boldsymbol{\Delta}^\top \boldsymbol{\Delta}$ *for convolution with periodic boundary conditions, and its counterpart diagonal matrix* $\mathbf{F}\mathbf{Q}\mathbf{F}^\mathsf{H}$ *in the Fourier domain, where* $\mathbf{F}$ *and its Hermitian conjugate* $\mathbf{F}^\mathsf{H}$ *are unitary matrices associated with the Fourier and inverse Fourier transforms.*

2.2.4. Multivariate Gaussian Sampling with Block Circulant (Toeplitz) Matrix $\mathbf{Q}$ with Circulant (Toeplitz) Blocks. An important special case of (2.1) which has already been surveyed [92] is when $\mathbf{Q}$ is a block circulant matrix with circulant blocks, that is,

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \dots & \mathbf{Q}_M \\ \mathbf{Q}_M & \mathbf{Q}_1 & \dots & \mathbf{Q}_{M-1} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{Q}_2 & \mathbf{Q}_3 & \dots & \mathbf{Q}_1 \end{pmatrix}, \tag{2.2}$$

where $\{\mathbf{Q}_i\}_{i\in[M]}$ are M circulant matrices. Such structured matrices frequently appear in image processing problems since they translate the convolution operator corresponding to linear and shift-invariant filters. As an illustration, Figure 2 shows the circulant structure of the precision matrix associated with the Gaussian distribution with density $\pi(\boldsymbol{\theta}) \propto \exp(-\|\boldsymbol{\Delta}\boldsymbol{\theta}\|^2/2)$. Here, the vector $\boldsymbol{\theta} \in \mathbb{R}^d$ is an image reshaped in lexicographic order and $\boldsymbol{\Delta}$ is the Laplacian differential operator with periodic boundaries, also called the Laplacian filter. In this case the precision matrix $\mathbf{Q} = \boldsymbol{\Delta}^\top \boldsymbol{\Delta}$ is a circulant matrix [78] so that it is diagonalizable in the Fourier domain. Therefore, sampling from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ can be performed in this domain as shown in Algorithm 2.4. For Gaussian distributions with more general Toeplitz precision matrices, $\mathbf{Q}$ can be replaced by its circulant approximation and then Algorithm 2.4 can be used; see [92] for more details. Although not considered in this paper, other approaches to generating stationary Gaussian processes [59] have been considered, such as the spectral [71, 97] and turning bands [65] methods.

Algorithmic Efficiency. Thanks to the use of the fast Fourier transform [26, 109], Algorithm 2.4 admits a computational complexity of $\mathcal{O}(d\log(d))$ flops. In addition, note that only d-dimensional vectors have to be stored, which implies a memory requirement of $\Theta(d)$. Overall, these complexities are significantly smaller than those of vanilla Cholesky sampling and as such Algorithm 2.4 can be considered to be "efficient."

2.2.5. Truncated and Intrinsic Gaussian Distributions. Several works have focused on sampling from various probability distributions closely related to the Gaussian distribution on $\mathbb{R}^d$. Two cases are worth mentioning here, namely, the *truncated* and so-called *intrinsic* Gaussian distributions. Truncated Gaussian distributions on

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 2.4 Sampler when $\mathbf{Q}$ is a block circulant matrix with circulant blocks

Input: M and N, the number of blocks, and the size of each block, respectively.

1: Compute $\mathbf{F} = \mathbf{F}_M \otimes \mathbf{F}_N$. ▷ $\mathbf{F}_M$ is the $M \times M$ unitary matrix associated to the Fourier transform and $\otimes$ denotes the tensor product.
2: Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
3: Set $\mathbf{\Lambda}_{\mathbf{q}} = \mathrm{diag}(\mathbf{q})$. ▷ $\mathbf{q}$ is the d-dimensional vector built by stacking the first columns of each circulant block of $\mathbf{Q}$.
4: Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{F}^{\mathsf{H}} \mathbf{\Lambda}_{\mathbf{q}}^{-1/2} \mathbf{F}\mathbf{z}$.
5: **return** $\boldsymbol{\theta}$.

$\mathcal{D} \subset \mathbb{R}^d$ admit, for any $\boldsymbol{\theta} \in \mathbb{R}^d$, p.d.f.s of the form

$$\pi_{\mathcal{D}}(\boldsymbol{\theta}) = \mathbf{1}_{\mathcal{D}}(\boldsymbol{\theta}) \cdot Z_{\mathcal{D}}^{-1} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu})\right) , \tag{2.3}$$

where $Z_{\mathcal{D}}$ is the appropriate normalizing constant and $\mathbf{1}_{\mathcal{D}}(\boldsymbol{\theta}) = 1$ if $\boldsymbol{\theta} \in \mathcal{D}$, and 0 otherwise. The subset $\mathcal{D}$ is usually defined by equalities and/or inequalities. Truncations on the hypercube are such that $\mathcal{D} = \prod_{i=1}^d [a_i, b_i]$, $(a_i, b_i) \in \mathbb{R}^2$, $1 \leq i \leq d$, and truncations on the simplex are such that $\mathcal{D} = \{\boldsymbol{\theta} \in \mathbb{R}^d \mid \sum_{i=1}^d \theta_i = 1\}$. Rejection and Gibbs sampling algorithms dedicated to these distributions can be found in [5, 60, 107].

Intrinsic Gaussian distributions are such that Σ is not of full rank, that is, $\mathbf{Q}$ may have eigenvalues equal to zero. This yields an improper Gaussian distribution often used as a prior in GMRFs to remove trend components [92]. Sampling from the latter can be done by identifying an appropriate subspace of $\mathbb{R}^d$, where the target distribution is proper, and then sampling from the proper Gaussian distribution on this subspace [12, 81].

All the usual special sampling problems above will not be considered in what follows since they have already been exhaustively reviewed in the literature.

2.3. Problem Statement: Sampling from a Gaussian Distribution with an Arbitrary Precision Matrix Q.

From now on, we will consider and review approaches aiming at sampling from an *arbitrary nonintrinsic multivariate* Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ with density defined in (2.1), i.e., without assuming any particular structure of the precision or covariance matrix. If $\mathbf{Q}$ is diagonal or well structured, we saw in subsection 2.2 that sampling can be performed more efficiently than vanilla Cholesky sampling, even in high dimension. When this matrix is unstructured and possibly dense, these methods with reduced numerical and storage complexities cannot be used anymore. In such settings the main challenges for Gaussian sampling are directly related to handling the precision $\mathbf{Q}$ (or covariance $\mathbf{\Sigma}$) matrix in high dimensions. Typical issues include the storage of the d^2 entries of the matrix $\mathbf{Q}$ (or $\mathbf{\Sigma}$) and expensive operations of order $\Theta(d^3)$ flops such as inversion or square roots, which become prohibitive when d is large. These challenges are illustrated below with an example that typically arises in statistical learning.

Example 2.1 (Bayesian ridge regression). Let us consider a ridge regression problem from a Bayesian perspective [13]. For the sake of simplicity and without loss of generality, assume that the observations $\mathbf{y} \in \mathbb{R}^n$ and the known predictor matrix

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

$\mathbf{X} \in \mathbb{R}^{n \times d}$ are such that

$$\sum_{i=1}^{n} y_i = 0 \ , \quad \sum_{i=1}^{n} X_{ij} = 0 \ , \quad \text{and} \quad \sum_{i=1}^{n} X_{ij}^2 = 1 \quad \text{for } j \in [d] \ . \tag{2.4}$$

Under these assumptions, we consider the statistical model associated with observations $\mathbf{y}$ written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \ , \tag{2.5}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_n, \sigma^2 \mathbf{I}_n)$. In this example, the standard deviation σ is known and fixed. The conditional prior distribution for $\boldsymbol{\theta}$ is chosen as Gaussian i.i.d., that is,

$$p(\boldsymbol{\theta} \mid \tau) \propto \exp\left(-\frac{1}{2\tau} ||\boldsymbol{\theta}||^2 \right) \ , \tag{2.6}$$

$$p(\tau) \propto \frac{1}{\tau} \mathbf{1}_{\mathbb{R}_+ \setminus \{0\}}(\tau) \ , \tag{2.7}$$

where $\tau > 0$ is an unknown variance parameter which is given a diffuse and improper (i.e., nonintegrable) Jeffrey's prior [53, 86]. The Bayes' rule then leads to the target joint posterior distribution with density

$$p(\boldsymbol{\theta}, \tau \mid \mathbf{y}) \propto \frac{1}{\tau} \mathbf{1}_{\mathbb{R}_+ \setminus \{0\}}(\tau) \ \exp\Big(-\frac{1}{2\tau} ||\boldsymbol{\theta}||^2 - \frac{1}{2\sigma^2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||^2 \Big) \ . \tag{2.8}$$

Sampling from this joint posterior distribution can be conducted using a Gibbs sampler [36, 87], which sequentially samples from the conditional posterior distributions. In particular, the conditional posterior distribution associated to $\boldsymbol{\theta}$ is Gaussian with precision matrix and mean vector

$$\mathbf{Q} = \frac{1}{\sigma^2} \mathbf{X}^\top \mathbf{X} + \tau^{-1} \mathbf{I}_d \ , \tag{2.9}$$

$$\boldsymbol{\mu} = \frac{1}{\sigma^2} \mathbf{Q}^{-1} \mathbf{X}^\top \mathbf{y} \ . \tag{2.10}$$

Challenges related to handling the matrix $\mathbf{Q}$ already appear in this classical and simple regression problem. Indeed, $\mathbf{Q}$ is possibly high-dimensional and dense, which potentially rules out its storage; see Table 1. The inversion required to compute the mean (2.10) may be very expensive as well. In addition, since τ is unknown, its value changes at each iteration of the Gibbs sampler used to sample from the joint distribution with density (2.8). Hence, precomputing the matrix $\mathbf{Q}^{-1}$ is not possible. As an illustration on real data, Figure 3 represents three examples of precision matrices[2] $\mathbf{X}^\top \mathbf{X}$ for the MNIST [57], leukemia [6], and CoEPrA [25] datasets. One can note that these precision matrices are potentially both high-dimensional and dense, penalizing their numerical inversion at each iteration of the Gibbs sampler.

Hosts of past contributions are related to high-dimensional Gaussian sampling: it is impossible to cite them in an exhaustive manner. As far as possible, the following review aims at gathering and citing the main contributions. We refer the reader

[2]When considering the dataset itself, $\mathbf{X}^\top \mathbf{X}$ is usually interpreted as the empirical covariance of the data $\mathbf{X}$. The reader should not be disturbed by the fact that, turning to the variable $\boldsymbol{\theta}$ to infer, $\mathbf{X}^\top \mathbf{X}$ will, however, play the role of a precision matrix.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

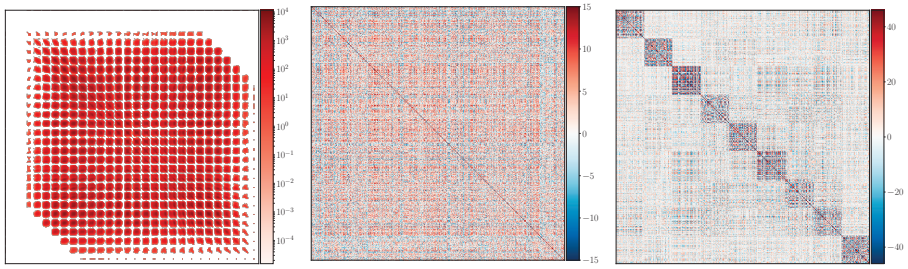


Fig. 3 *Examples of precision matrices* $\mathbf{X}^\top\mathbf{X}$ *for three datasets. Left: MNIST dataset* [57]*. Only the predictors associated to the digits* 5 *and* 3 *have been taken into account for the MNIST dataset* [57]*. Middle: Leukemia dataset* [6]*. For the leukemia dataset* [6]*, only the first* 5,000 *predictors (out of* 12,600*) have been taken into account. Right: CoEPrA dataset* [25]*.*

to references therein for more details. Next, sections 3 and 4 review the two main families of approaches that deal with the sampling issues raised above. In section 3, we deal with approaches derived from numerical linear algebra. On the other hand, section 4 deals with MCMC sampling approaches. Section 5 then proposes a unifying revisit to Gibbs samplers thanks to a stochastic counterpart of the PPA. Similarly to subsection 2.2, computational costs, storage requirements, and accuracy of the reviewed Gaussian sampling approaches will be detailed for each method in a dedicated paragraph entitled *Algorithmic Efficiency*. For a synthetic summary and comparison of these metrics, we refer the interested reader to Table 7.

3. Sampling Algorithms Derived from Numerical Linear Algebra. This section presents sampling approaches corresponding to direct adaptations of classical techniques used in numerical linear algebra [41]. They can be divided into three main groups: (i) factorization methods that consider appropriate decompositions of $\mathbf{Q}$, (ii) inverse square root approximation approaches where approximations of $\mathbf{Q}^{-1/2}$ are used to obtain samples from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ at a reduced cost compared to factorization approaches, and (iii) conjugate gradient based methods.

3.1. Factorization Methods. We begin this review with the most basic but computationally involved sampling techniques, namely, factorization approaches that were introduced in section 1. These methods exploit the positive definiteness of $\mathbf{Q}$ to decompose it as a product of simpler matrices and are essentially based on the celebrated Cholesky factorization [20]. Though helpful for problems in small or moderate dimensions, these basic sampling approaches fail to address, in high-dimensional scenarios, the computational and memory issues raised in subsection 2.3.

3.1.1. Cholesky Factorization. Since $\mathbf{Q}$ is symmetric and positive definite, we noted in (1.1) that there exists a unique lower triangular matrix $\mathbf{C} \in \mathbb{R}^{d\times d}$, called the Cholesky factor, with positive diagonal entries such that $\mathbf{Q} = \mathbf{C}\mathbf{C}^\top$ [41]. Algorithm 3.1 details how such a decomposition[3] can be used to obtain a sample $\boldsymbol{\theta}$ from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$.

Algorithmic Efficiency. In the general case where $\mathbf{Q}$ presents no particular structure, the computational cost is $\Theta(d^3)$ and the storage requirement is $\Theta(d^2)$; see also section 1

[3]When working with the covariance matrix rather than the precision matrix, the Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$ leads to the simpler step 3: $\mathbf{w} = \mathbf{L}\mathbf{z}$.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

and Table 1. If the dimension d is large but the matrix $\mathbf{Q}$ has a sparse structure, the computational and storage requirements of the Cholesky factorization can be reduced by reordering the components of $\mathbf{Q}$ to design an equivalent band matrix [91]; see subsection 2.2 and Algorithm 2.3.

Algorithm 3.1 Cholesky sampler

1: Set $\mathbf{C} = \text{chol}(\mathbf{Q})$. ▷ Build the Cholesky factor $\mathbf{C}$ of $\mathbf{Q}$; see [92, section 2.4].
2: Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
3: Solve $\mathbf{C}^\top \mathbf{w} = \mathbf{z}$ w.r.t. $\mathbf{w}$.
4: **return** $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{w}$.

3.1.2. Square Root Factorization. The Cholesky factorization in the previous paragraph was used to decompose $\mathbf{Q}$ into a product of a triangular matrix $\mathbf{C}$ and its transpose. Then a Gaussian sample was obtained by solving a triangular linear system. An extension of this approach was considered in [24] which performed a singular value decomposition (SVD) of the Cholesky factor $\mathbf{C}$, which yields $\mathbf{Q} = \mathbf{B}^2$ with $\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}\mathbf{U}^\top$, where $\boldsymbol{\Lambda}$ is diagonal and $\mathbf{U}$ is orthogonal. Similar to the Cholesky factor and given $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, this square root can then be used to obtain an arbitrary Gaussian sample by solving $\mathbf{B}\mathbf{w} = \mathbf{z}$ w.r.t. $\mathbf{w}$ and computing $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{w}$.

Algorithmic Efficiency. Although the square root factorization is interesting for establishing the existence of $\mathbf{B}$, the associated sampler is generally as computationally demanding as Algorithm 3.1 since the eigendecomposition of $\mathbf{Q}$ is not cheaper than finding its Cholesky factor. To avoid these computational problems and since samplers based on $\mathbf{B}$ boil down to computing $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{B}^{-1}\mathbf{z}$, some works have focused on approximations of the inverse square root $\mathbf{B}^{-1}$ of $\mathbf{Q}$ that require smaller computational and storage complexities.

3.2. Inverse Square Root Approximations. This idea of finding an efficient (compared to the costs associated to factorization approaches) way to approximate the inverse square root $\mathbf{B}^{-1}$ dates back at least to the 1980s with the work of Davis [24], who derived a polynomial approximation of the function $x \mapsto x^{1/2}$ to approximate the square root of a given covariance matrix. Other works used Krylov-based approaches, building on the Lanczos decomposition to approximate directly any matrix-vector product $\mathbf{B}^{-1}\mathbf{z}$ involving the square root $\mathbf{B}$. The following two paragraphs review these methods.

3.2.1. Polynomial Approximation. In subsection 3.1, we showed that the square root of $\mathbf{Q}$ can be written as $\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}\mathbf{U}^\top$, which implies that $\mathbf{Q} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$. If f is a real continuous function, $\mathbf{Q}$ and $f(\mathbf{Q}) = \mathbf{U}f(\boldsymbol{\Lambda})\mathbf{U}^\top$ are diagonalizable with respect to the same eigenbasis $\mathbf{U}$, where $f(\boldsymbol{\Lambda}) \triangleq \text{diag}(f(\lambda_1), \dots, f(\lambda_d))$. This is a well-known result coming from the Taylor expansion of a real continuous function f. Hence, a function f such that $f(\mathbf{Q})$ is a good approximation of $\mathbf{B}^{-1} = \mathbf{Q}^{-1/2}$ has to be such that

$$f(\lambda_i) \approx 1/\sqrt{\lambda_i} \quad \forall i \in [d] \, .$$

Since f only needs to be evaluated at the points corresponding to the eigenvalues $\{\lambda_i\}_{i\in[d]}$ of $\mathbf{Q}$, it suffices to find a good approximation of $\mathbf{B}^{-1}$ on the interval $[\lambda_{\min}, \lambda_{\max}]$ whose extremal values can be lower and upper bounded easily using the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Gershgorin circle theorem [41, Theorem 7.2.1]. In the literature [24, 51, 82], the function f has been built using Chebyshev polynomials [70], which are a family $(T_k)_{k\in\mathbb{N}}$ of polynomials defined over $[-1,1]$ by

$$T_k(x) = \cos(k\alpha)\ , \quad \text{where } \ \forall \alpha \in \mathbb{R},\ x = \cos(\alpha)\ ,$$

or by the recursion

$$\begin{cases} T_0(x) = 1, \\ T_1(x) = x, \\ T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \quad (\forall k \geq 1)\ . \end{cases} \tag{3.1}$$

This family of polynomials $(T_k)_{k\in\mathbb{N}}$ exhibits several interesting properties: uniform convergence of the Chebyshev series toward an arbitrary Lipschitz-continuous function over $[-1,1]$ and near minimax property [70], along with fast computation of the coefficients of the series via the fast Fourier transform [84]. Algorithm 3.2 describes the steps to generating arbitrary Gaussian vectors using this polynomial approximation.

Algorithmic Efficiency. Contrary to factorization methods detailed in subsection 3.1, Algorithm 3.2 does not require the storage of $\mathbf{Q}$ since only the computation of matrix-vector products of the form $\mathbf{Qv}$ with $\mathbf{v} \in \mathbb{R}^d$ is necessary. Assuming that these operations can be computed efficiently in $\mathcal{O}(d^2)$ flops with some black-box routine, e.g., a fast wavelet transform [64], Algorithm 3.2 admits an overall computational cost of $\mathcal{O}(K_{\text{cheby}}d^2)$ flops and storage capacity of $\Theta(d)$, where K_{cheby} is a truncation parameter giving the order of the polynomial approximation. When $K_{\text{cheby}} \ll d$, Algorithm 3.2 becomes more computationally efficient than vanilla Cholesky sampling while admitting a reasonable memory overhead. For a sparse precision matrix $\mathbf{Q}$ composed of n_{nz} nonzero entries, the computational complexity can be reduced down to $\mathcal{O}(K_{\text{cheby}}n_{\text{nz}})$ flops. Note that compared to factorization approaches, Algorithm 3.2 involves an additional source of approximation coming from the order of the Chebyshev series K_{cheby}. Choosing this parameter in an adequate manner involves some fine-tuning or additional computationally intensive statistical tests [82].

3.2.2. Lanczos Approximation. Instead of approximating the inverse square root $\mathbf{B}^{-1}$, some approaches directly approximate the matrix-vector product $\mathbf{B}^{-1}\mathbf{z}$ by building on the Lanczos decomposition [7, 21, 51, 52, 98, 99]. The corresponding simulation-based algorithm is described in Algorithm 3.3. It iteratively builds an orthonormal basis $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_{K_{\text{kryl}}}\} \in \mathbb{R}^{d\times K_{\text{kryl}}}$ with $K_{\text{kryl}} \leq d$ for the Krylov subspace $\mathcal{K}_{K_{\text{kryl}}}(\mathbf{Q},\mathbf{z}) \triangleq \text{span}\{\mathbf{z}, \mathbf{Qz}, \ldots, \mathbf{Q}^{K_{\text{kryl}}-1}\mathbf{z}\}$, and a tridiagonal matrix $\mathbf{T} \approx \mathbf{H}^\top\mathbf{QH} \in \mathbb{R}^{K_{\text{kryl}}\times K_{\text{kryl}}}$. Using the orthogonality of $\mathbf{H}$, $\mathbf{z} = \mathbf{He}_1$, where $\mathbf{e}_1$ is the first canonical vector of $\mathbb{R}^{K_{\text{kryl}}}$, the final approximation is

$$\mathbf{B}^{-1}\mathbf{z} = \mathbf{Q}^{-1/2}\mathbf{z} \approx \|\mathbf{z}\|\,\mathbf{HT}^{-1/2}\mathbf{H}^\top\mathbf{He}_1 = \|\mathbf{z}\|\,\mathbf{HT}^{-1/2}\mathbf{e}_1\ . \tag{3.2}$$

As highlighted in (3.2), the main idea of the Lanczos approximation is to transform the computation of $\mathbf{Q}^{-1/2}\mathbf{z}$ into the computation of $\mathbf{T}^{-1/2}\mathbf{e}_1$, which is expected to be simpler since $\mathbf{T}$ is tridiagonal and of size $K_{\text{kryl}} \times K_{\text{kryl}}$.

Algorithmic Efficiency. Numerous approaches have been proposed to compute $\mathbf{T}^{-1/2}\mathbf{e}_1$ exactly or approximately, and they generally require $\mathcal{O}(K_{\text{kryl}}^2)$ flops [22, 44]; see also Algorithm 3.2. By using such approaches, Algorithm 3.3 admits a computational complexity of $\mathcal{O}(K_{\text{kryl}}d^2)$ and a memory requirement of $\mathcal{O}(K_{\text{kryl}}d)$. Similarly to K_{cheby} in

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 3.2 Approximate square root sampler using Chebyshev polynomials

Set $\lambda_l = 0$ and $\lambda_u = \max_{i\in[d]} \sum_{j\in[d]} |Q_{ij}|$.
for $j \in [\![0, K_{\text{cheby}}]\!]$ **do** ▷ Do the change of interval.
  Set $g_j = \left[\cos\left(\pi \frac{2j+1}{2K_{\text{cheby}}}\right) \frac{(\lambda_u - \lambda_l)}{2} + \frac{\lambda_u + \lambda_l}{2}\right]^{-1/2}$.
end for
for $k \in [\![0, K_{\text{cheby}}]\!]$ **do** ▷ Compute coefficients of the K_{cheby}-truncated Chebyshev series.
  Compute $c_k = \frac{2}{K_{\text{cheby}}} \sum_{j=0}^{K_{\text{cheby}}} g_j \cos\left(\pi k \frac{2j+1}{2K_{\text{cheby}}}\right)$.
end for
Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
Set $\alpha = \frac{2}{\lambda_u - \lambda_l}$ and $\beta = \frac{\lambda_u + \lambda_l}{\lambda_u - \lambda_l}$.
Initialize $\mathbf{u}_1 = \alpha\mathbf{Q}\mathbf{z} - \beta\mathbf{z}$ and $\mathbf{u}_0 = \mathbf{z}$.
Set $\mathbf{u} = \frac{1}{2}c_0\mathbf{u}_0 + c_1\mathbf{u}_1$ and $k = 2$.
while $k \leq K_{\text{cheby}}$ **do** ▷ Compute the K_{cheby}-truncated Chebyshev series.
  Compute $\mathbf{u}' = 2(\alpha\mathbf{Q}\mathbf{u}_1 - \beta\mathbf{u}_1) - \mathbf{u}_0$.
  Set $\mathbf{u} = \mathbf{u} + c_k\mathbf{u}'$.
  Set $\mathbf{u}_0 = \mathbf{u}_1$ and $\mathbf{u}_1 = \mathbf{u}'$.
  $k = k + 1$.
end while
Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{u}$. ▷ Build the Gaussian vector of interest.
return $\boldsymbol{\theta}$.

Algorithm 3.2, one can note that K_{kryl} represents a trade-off between computation, storage, and accuracy. As emphasized in [7, 98], adjusting this truncation parameter can be achieved by using the conjugate gradient (CG) algorithm. In addition to providing an approximate sampling technique when $K_{\text{kryl}} < d$, the main and well-known drawback of Algorithm 3.3 is that the basis $\mathbf{H}$ loses orthogonality in floating point arithmetic due to round-off errors. Some possibly complicated procedures to cope with this problem are surveyed in [100]. Finally, one major problem of the Lanczos decomposition is the construction and storage of the basis $\mathbf{H} \in \mathbb{R}^{d\times K_{\text{kryl}}}$, which becomes as large as $\mathbf{Q}$ when K_{kryl} tends to d. Two main approaches have been proposed to deal with this problem, namely, a so-called 2-pass strategy and a restarting strategy, both reviewed in [7, 52, 98]. In addition, preconditioning methods have been proposed to reduce the computational burden of Lanczos samplers [21].

3.2.3. Other Square Root Approximations. At least two other methods have been proposed to approximate the inverse square root $\mathbf{B}^{-1}$. Since these approaches have been less used than others, only their main principle is given, and we refer the interested reader to the corresponding references. The first one is the rational approximation of $\mathbf{B}^{-1}$ based on numerical quadrature of a contour integral [45], while the other one is a continuous deformation method based on a system of ordinary differential equations [4]. These two approaches are reviewed and illustrated using numerical examples in [7].

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 3.3 Approximate square root sampler using Lanczos decomposition

Draw $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
Set $\mathbf{r}^{(0)} = \mathbf{z}$, $\mathbf{h}^{(0)} = \mathbf{0}_d$, $\beta^{(0)} = \left\|\mathbf{r}^{(0)}\right\|$ and $\mathbf{h}^{(1)} = \mathbf{r}^{(0)}/\beta^{(0)}$.
for $k \in [K_{\text{kryl}}]$ **do**
    Set $\mathbf{w} = \mathbf{Q}\mathbf{h}^{(k)} - \beta^{(k-1)}\mathbf{h}^{(k-1)}$.
    Set $\alpha^{(k)} = \mathbf{w}^\top \mathbf{h}^{(k)}$.
    Set $\mathbf{w} = \mathbf{w} - \alpha^{(k)}\mathbf{h}^{(k)}$. ▷ Gram–Schmidt orthogonalization process.
    Set $\beta^{(k)} = \|\mathbf{w}\|$.
    Set $\mathbf{h}^{(k+1)} = \mathbf{w}/\beta^{(k)}$.
end for
Set $\mathbf{T} = \text{tridiag}(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\beta})$.
Set $\mathbf{H} = (\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(K_{\text{kryl}})})$.

Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \beta^{(0)}\mathbf{H}\mathbf{T}^{-1/2}\mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, \ldots, 0)^\top \in \mathbb{R}^{K_{\text{kryl}}}$.
return $\boldsymbol{\theta}$.

3.3. Conjugate Gradient–Based Samplers. Instead of building on factorization results, some approaches start from the finding that Gaussian densities with invertible precision matrix $\mathbf{Q}$ can be rewritten in a so-called *information form*, that is,

$$\pi(\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\top \mathbf{Q}\boldsymbol{\theta} + \mathbf{b}^\top \boldsymbol{\theta}\right) , \tag{3.3}$$

where $\mathbf{b} = \mathbf{Q}\boldsymbol{\mu}$ is called the potential vector. If one is able to draw a Gaussian vector $\mathbf{z}' \sim \mathcal{N}(\mathbf{0}_d, \mathbf{Q})$, then a sample $\boldsymbol{\theta}$ from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ is obtained by solving the linear system

$$\mathbf{Q}\boldsymbol{\theta} = \mathbf{b} + \mathbf{z}' , \tag{3.4}$$

where $\mathbf{Q}$ is positive definite, so that CG methods are relevant. This approach uses the affine transformation of a Gaussian vector $\mathbf{u} = \mathbf{b} + \mathbf{z}'$: if $\mathbf{u} \sim \mathcal{N}(\mathbf{Q}\boldsymbol{\mu}, \mathbf{Q})$, then $\mathbf{Q}^{-1}\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$.

3.3.1. Perturbation before Optimization. A first possibility to handling the perturbed linear problem (3.4) consists of first computing the potential vector $\mathbf{b}$, then perturbing this vector with the Gaussian vector $\mathbf{z}'$, and finally solving the linear system with numerical algebra techniques. This approach is detailed in Algorithm 3.4. While the computation of $\mathbf{b}$ is not difficult in general, drawing $\mathbf{z}'$ might be computationally involved. Hence, this sampling approach is of interest only if we are able to draw the Gaussian vector $\mathbf{z}'$ efficiently (i.e., in $\mathcal{O}(d^2)$ flops). This is, for instance, the case when $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ with $\mathbf{Q}_i = \mathbf{G}_i^\top \boldsymbol{\Lambda}_i^{-1}\mathbf{G}_i$ ($i \in [2]$), provided that the symmetric and positive-definite matrices $\{\boldsymbol{\Lambda}_i\}_{i\in[2]}$ have simple structures; see subsection 2.2. Such situations often arise when Bayesian hierarchical models are considered [86, Chapter 10]. For these scenarios, an efficient way to compute $\mathbf{b} + \mathbf{z}'$ has been proposed in [79] based on a local perturbation of the mean vectors $\{\boldsymbol{\mu}_i\}_{i\in[2]}$. Such an approach has been coined *perturbation-optimization* (PO) since it draws perturbed versions of the mean vectors involved in the hierarchical model before using them to define the linear system to be solved [79].

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 3.4 Perturbation-optimization sampler

1: Draw $\mathbf{z}' \sim \mathcal{N}(\mathbf{0}_d, \mathbf{Q})$. ▷ with local perturbation as in [79].
2: Set $\boldsymbol{\eta} = \mathbf{b} + \mathbf{z}'$.
3: Solve $\mathbf{Q}\boldsymbol{\theta} = \boldsymbol{\eta}$ w.r.t. $\boldsymbol{\theta}$. ▷ with the CG solver as used, for instance, in [47].
4: **return** $\boldsymbol{\theta}$.

Algorithmic Efficiency. If $K \in \mathbb{N}^*$ iterations of an appropriate linear solver (e.g., the CG method) are used for step 3 in Algorithm 3.4, the global computational and storage complexities of this algorithm are of order $\mathcal{O}(Kd^2)$ and $\Theta(d)$. Regarding sampling accuracy, while Algorithm 3.4 in theory is an exact approach, the K-truncation procedure implies an approximate sampling scheme [77]. A solution to correct this bias has been proposed in [37] which builds upon a reversible-jump approach [43].

3.3.2. Optimization with Perturbation. Alternatively, (3.4) can also be seen as a perturbed version of the linear system $\mathbf{Q}\boldsymbol{\theta} = \mathbf{b}$. Thus, some works have focused on modified versions of well-known linear solvers such as the CG solver [81, 91, 95]. Actually, only one additional line of code providing a univariate Gaussian sampling step (perturbation) is required to turn the classical CG solver into a CG sampler [81, 95]; see step 8 in Algorithm 3.5. This perturbation step sequentially builds a Gaussian vector with a covariance matrix that is the best k-rank approximation of $\mathbf{Q}^{-1}$ in the Krylov subspace $\mathcal{K}_k(\mathbf{Q}, \mathbf{r}^{(0)})$ [81]. Then a perturbation vector $\mathbf{y}^{(K_{\mathrm{CG}})}$ is simulated before addition to $\boldsymbol{\mu}$ so that finally $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{y}^{(K_{\mathrm{CG}})}$.

Algorithm 3.5 Conjugate gradient sampler

Input: Threshold $\epsilon > 0$, fixed initialization $\boldsymbol{\omega}^{(0)}$, and random vector $\mathbf{c} \in \mathbb{R}^d$.
1: Set $k = 1$, $\mathbf{r}^{(0)} = \mathbf{c} - \mathbf{Q}\boldsymbol{\omega}^{(0)}$, $\mathbf{h}^{(0)} = \mathbf{r}^{(0)}$, $d^{(0)} = \mathbf{h}^{(0)\top}\mathbf{Q}\mathbf{h}^{(0)}$, and $\mathbf{y}^{(0)} = \boldsymbol{\omega}^{(0)}$.
2: **while** $\left\|\mathbf{r}^{(k)}\right\| \geq \epsilon$ **do**
3: Set $\gamma^{(k-1)} = \dfrac{\mathbf{r}^{(k-1)\top}\mathbf{r}^{(k-1)}}{d^{(k-1)}}$.
4: Set $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \gamma^{(k-1)}\mathbf{Q}\mathbf{h}^{(k-1)}$.
5: Set $\eta^{(k)} = -\dfrac{\mathbf{r}^{(k)\top}\mathbf{r}^{(k)}}{\mathbf{r}^{(k-1)T}\mathbf{r}^{(k-1)}}$.
6: Set $\mathbf{h}^{(k)} = \mathbf{r}^{(k)} - \eta^{(k)}\mathbf{h}^{(k-1)}$.
7: Set $d^{(k)} = \mathbf{h}^{(k)\top}\mathbf{Q}\mathbf{h}^{(k)}$.
8: Set $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \dfrac{z}{\sqrt{d^{(k-1)}}}\mathbf{h}^{(k-1)}$, where $z \sim \mathcal{N}(0,1)$.
9: $k = k + 1$.
10: **end while**
11: Set $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{y}^{(K_{\mathrm{CG}})}$, where K_{CG} is the number of CG iterations.
12: **return** $\boldsymbol{\theta}$.

Algorithmic Efficiency. From a computational point of view, the CG sampler inherits the benefits of the CG solver: only matrix-vector products involving $\mathbf{Q}$ and the storage of two d-dimensional vectors are needed, and one exact sample from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ is obtained after at most $K_{\mathrm{CG}} = d$ iterations. This yields an approximate computational cost of $\mathcal{O}(K_{\mathrm{CG}}d^2)$ flops and a storage requirement of $\Theta(d)$, where K_{CG} is the number of CG iterations [47]. The CG sampler belongs to the family of Krylov-based samplers (e.g., Lanczos). As such, it suffers from the same numerical problem due to

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

finite machine precision and the K_{CG}-truncation procedure. In addition, the covariance of the generated samples depends on the distribution of the eigenvalues of the matrix $\mathbf{Q}$. Actually, if these eigenvalues are not well spread out, Algorithm 3.5 stops after $K_{\mathrm{CG}} < d$ iterations, which yields an approximate sample with the best K_{CG}-rank approximation of $\mathbf{Q}^{-1}$ as the actual covariance matrix. In order to correct this approximation, reorthogonalization schemes can be employed but could become as computationally prohibitive as Cholesky sampling when d is large [95]. These sources of approximation are detailed in [81]. A generalization of Algorithm 3.5 has been considered in [30], where a random set of K' mutually conjugate directions $\{\mathbf{h}^{(k)}\}_{k\in[K']}$ is considered at each iteration of a Gibbs sampler.

4. Sampling Algorithms Based on MCMC. The previous section presented existing Gaussian sampling approaches by directly adapting ideas and techniques from numerical linear algebra such as matrix decompositions and matrix approximations. In this section, we will present another family of sampling approaches, namely, MCMC approaches, which build a discrete-time Markov chain $(\boldsymbol{\theta}^{(t)})_{t\in\mathbb{N}}$ with $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ (or a close approximation of $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$) as its invariant distribution [87]. In what follows, we state that an MCMC approach is exact if the associated sampler admits an invariant distribution that coincides with $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. Contrary to the approaches reviewed in section 3, which produce i.i.d. samples from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ or a close approximation to it, MCMC approaches produce correlated samples that are asymptotically distributed according to their invariant distribution. Hence, at a first glance, it seems natural to think that MCMC samplers are less trustworthy since the number of iterations required until convergence is very difficult to assess in practice [49]. Interestingly, we will show numerically in section 6 that MCMC methods might perform better than i.i.d. samplers in some cases and thus might be serious contenders for the most efficient Gaussian sampling algorithms; see also [32]. Furthermore, we will also show that most of these MCMC approaches can be unified via a stochastic version of the PPA [90]. This framework will be presented in section 5.

4.1. Matrix Splitting. We begin the review of MCMC samplers by detailing so-called *matrix-splitting* (MS) approaches that build on the decomposition $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ of the precision matrix. As we shall see, both exact and approximate MS samplers have been proposed in the existing literature. These methods embed one of the simplest MCMC methods, namely, the componentwise Gibbs sampler [36]. Similarly to Algorithm 3.1 for samplers in section 3, it can be viewed as one of the simplest and most straightforward approaches to sampling from a target Gaussian distribution.

4.1.1. Exact Matrix Splitting. Given the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ with density π in (2.1), an attractive and simple option is to sequentially draw one component of $\boldsymbol{\theta}$ given the others. This is the well-known componentwise Gibbs sampler; see Algorithm 4.1 [35, 36, 92]. The main advantage of Algorithm 4.1 is its simplicity and the low cost per *sweep* (i.e., internal iteration) of $\mathcal{O}(d^2)$ flops, which is comparable with Cholesky applied to Toeplitz covariance matrices [102]. More generally, one can also consider random sweeps over the d components of $\boldsymbol{\theta}$ or blockwise strategies which update simultaneously several components of $\boldsymbol{\theta}$. The analysis of these strategies and their respective convergence rates are detailed in [88].

In [1, 10, 42], the authors showed by rewriting Algorithm 4.1 using a matrix formulation that it actually provides a stochastic version of the Gauss–Seidel linear solver that relies on the decomposition $\mathbf{Q} = \mathbf{L} + \mathbf{D} + \mathbf{L}^{\top}$, where $\mathbf{L}$ and $\mathbf{D}$ are the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 4.1 Componentwise Gibbs sampler

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: **for** $i \in [d]$ **do**
4: Draw $z \sim \mathcal{N}(0, 1)$.
5: Set $\theta_i^{(t)} = \frac{[\mathbf{Q}\boldsymbol{\mu}]_i}{Q_{ii}} + \frac{z}{\sqrt{Q_{ii}}} - \frac{1}{Q_{ii}} \left(\sum_{j>i} Q_{ij}\theta_j^{(t-1)} + \sum_{j<i} Q_{ij}\theta_j^{(t)} \right)$.
6: **end for**
7: Set $t = t + 1$.
8: **end while**
9: **return** $\boldsymbol{\theta}^{(T)}$.

strictly lower triangular and diagonal parts of $\mathbf{Q}$, respectively. Indeed, each iteration solves the linear system

$$(\mathbf{L} + \mathbf{D})\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \mathbf{D}^{1/2}\mathbf{z} - \mathbf{L}^\top \boldsymbol{\theta}^{(t-1)} , \tag{4.1}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$. By setting $\mathbf{M} = \mathbf{L} + \mathbf{D}$ and $\mathbf{N} = -\mathbf{L}^\top$ so that $\mathbf{Q} = \mathbf{M} - \mathbf{N}$, the updating rule (4.1) can be written as solving the usual Gauss–Seidel linear system

$$\mathbf{M}\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \tilde{\mathbf{z}} + \mathbf{N}\boldsymbol{\theta}^{(t-1)} , \tag{4.2}$$

where $\mathbf{N} = -\mathbf{L}^\top$ is strictly upper triangular and $\tilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{D})$ is easy to sample.

Interestingly, (4.2) is a perturbed instance of MS schemes which are a class of linear iterative solvers based on the splitting of $\mathbf{Q}$ into $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ [41, 93]. Capitalizing on this one-to-one equivalence between samplers and linear solvers, the authors in [32] extended Algorithm 4.1 to other MCMC samplers based on different matrix splittings $\mathbf{Q} = \mathbf{M} - \mathbf{N}$. These are reported in Table 2 and yield Algorithm 4.2. The acronym SOR stands for *successive overrelaxation*.

Table 2 *Examples of MS schemes for* $\mathbf{Q}$ *which can be used in Algorithm* 4.2. *The matrices* $\mathbf{D}$ *and* $\mathbf{L}$ *denote the diagonal and strictly lower triangular parts of* $\mathbf{Q}$, *respectively. The vector* $\tilde{\mathbf{z}}$ *is the one appearing in step* 3 *of Algorithm* 4.2 *and* ω *is a positive scalar.*

Sampler	$\mathbf{M}$	$\mathbf{N}$	$\mathrm{cov}(\tilde{\mathbf{z}}) = \mathbf{M}^\top + \mathbf{N}$	Convergence
Richardson	$\mathbf{I}_d/\omega$	$\mathbf{I}_d/\omega - \mathbf{Q}$	$2\mathbf{I}_d/\omega - \mathbf{Q}$	$0 < \omega < 2/\left\|\mathbf{Q}\right\|$
Jacobi	$\mathbf{D}$	$\mathbf{D} - \mathbf{Q}$	$2\mathbf{D} - \mathbf{Q}$	$\lvert Q_{ii}\rvert > \sum_{j\neq i} \lvert Q_{ij}\rvert \ \forall i \in [d]$
Gauss–Seidel	$\mathbf{D} + \mathbf{L}$	$-\mathbf{L}^\top$	$\mathbf{D}$	always
SOR	$\mathbf{D}/\omega + \mathbf{L}$	$\frac{1-\omega}{\omega}\mathbf{D} - \mathbf{L}^\top$	$\frac{2-\omega}{\omega}\mathbf{D}$	$0 < \omega < 2$

Algorithmic Efficiency. Similarly to linear solvers, Algorithm 4.2 is guaranteed to converge toward the correct distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ if $\rho\left(\mathbf{M}^{-1}\mathbf{N}\right) < 1$, where $\rho(\cdot)$ is the spectral radius of a matrix. In practice, Algorithm 4.2 is stopped after T iterations and the error between the distribution of $\boldsymbol{\theta}^{(T)}$ and $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ can be assessed quantitatively; see [32]. The computational efficiency of Algorithm 4.2 is directly related to the complexity of solving the linear systems $\mathbf{M}\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \tilde{\mathbf{z}} + \mathbf{N}\boldsymbol{\theta}^{(t-1)}$, similar

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Algorithm 4.2 MCMC sampler based on exact matrix splitting

Input: Number T of iterations, initialization $\boldsymbol{\theta}^{(0)}$, and splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\tilde{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{M}^\top + \mathbf{N})$.
4: Solve $\mathbf{M}\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \tilde{\mathbf{z}} + \mathbf{N}\boldsymbol{\theta}^{(t-1)}$ w.r.t. $\boldsymbol{\theta}^{(t)}$.
5: Set $t = t + 1$.
6: **end while**
7: **return** $\boldsymbol{\theta}^{(T)}$.

to (4.2), and the difficulty of sampling $\tilde{\mathbf{z}}$ with covariance $\mathbf{M}^\top + \mathbf{N}$. As pointed out in [32], the simpler $\mathbf{M}$, the denser $\mathbf{M}^\top + \mathbf{N}$ and the more difficult the sampling of $\tilde{\mathbf{z}}$. For instance, Jacobi and Richardson schemes yield a simple diagonal linear system requiring $\mathcal{O}(d)$ flops, but one has to sample from a Gaussian distribution with an arbitrary covariance matrix; see step 3 of Algorithm 4.2. Iterative samplers requiring at least K steps such as those reviewed in section 3 can be used. This yields a computational burden of $\mathcal{O}(KTd^2)$ flops for step 3 and as such Jacobi and Richardson samplers admit a computational cost of $\mathcal{O}(KTd^2)$ and a storage requirement of $\Theta(d)$. On the other hand, both Gauss–Seidel and SOR schemes are associated to a simple sampling step which can be performed in $\mathcal{O}(d)$ flops with Algorithm 2.2, but one has to solve a lower triangular system which can be done in $\mathcal{O}(d^2)$ flops via forward substitution. In order to mitigate the trade-off between steps 3 and 4, approximate MS approaches have been proposed recently [9, 54]; see subsection 4.1.2.

Polynomial Accelerated Gibbs Samplers. When the splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ is symmetric, that is, both $\mathbf{M}$ and $\mathbf{N}$ are symmetric matrices, the rate of convergence of Algorithm 4.2 can be improved by using polynomial preconditioners [32]. For ease of presentation, we will first explain how such a preconditioning accelerates linear solvers based on MS, before building upon the one-to-one equivalence between linear solvers and Gibbs samplers to show how Algorithm 4.2 can be accelerated. Given a linear system $\mathbf{Q}\boldsymbol{\theta} = \mathbf{v}$ for $\mathbf{v} \in \mathbb{R}^d$, linear solvers based on the MS, $\mathbf{Q} = \mathbf{M} - \mathbf{N}$, consider the recursion, for any $t \in \mathbb{N}$ and $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$, $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathbf{M}^{-1}(\mathbf{v} - \mathbf{Q}\boldsymbol{\theta}^{(t)})$. The error at iteration t defined by $\mathbf{e}^{(t+1)} = \boldsymbol{\theta}^{(t+1)} - \mathbf{Q}^{-1}\mathbf{v}$ can be shown to be equal to $(\mathbf{I}_d - \mathbf{M}^{-1}\mathbf{Q})^t\mathbf{e}^{(0)}$ [41]. Since this error is a tth-order polynomial of $\mathbf{M}^{-1}\mathbf{Q}$, it is then natural to wonder whether one can find another tth order polynomial P_t that achieves a lower error, that is, $\rho(\mathsf{P}_t(\mathbf{M}^{-1}\mathbf{Q})) < \rho((\mathbf{I}_d - \mathbf{M}^{-1}\mathbf{Q})^t)$. This can be accomplished by considering the second-order iterative scheme defined, for any $t \in \mathbb{N}$, by [8]

$$\boldsymbol{\theta}^{(t+1)} = \alpha_t\boldsymbol{\theta}^{(t)} + (1 - \alpha_t)\boldsymbol{\theta}^{(t-1)} + \beta_t\mathbf{M}^{-1}(\mathbf{v} - \mathbf{Q}\boldsymbol{\theta}^{(t)}) \,,$$

where $(\alpha_t, \beta_t)_{t\in\mathbb{N}}$ is a set of acceleration parameters. This iterative method yields an error at step t given by $\mathbf{e}^{(t+1)} = \mathsf{P}_t(\mathbf{M}^{-1}\mathbf{Q})\mathbf{e}^{(0)}$, where P_t is a scaled Chebyshev polynomial; see (3.1). Optimal values for $(\alpha_t, \beta_t)_{t\in\mathbb{N}}$ are given by [8]

$$\alpha_t = \tau_1\beta_t \quad \text{and} \quad \beta_t = \left(\tau_1 - \tau_2^2\beta_{t-1}\right)^{-1} \,,$$

$\tau_1 = [\lambda_{\min}(\mathbf{M}^{-1}\mathbf{Q}) + \lambda_{\max}(\mathbf{M}^{-1}\mathbf{Q})]/2$ and $\tau_2 = [\lambda_{\max}(\mathbf{M}^{-1}\mathbf{Q}) - \lambda_{\min}(\mathbf{M}^{-1}\mathbf{Q})]/4$. Note that these optimal choices suppose that the minimal and maximal eigenvalues of $\mathbf{M}^{-1}\mathbf{Q}$ are real valued and available. The first requirement is satisfied, for instance,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

if the splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ is symmetric, while the second one is met by using the CG algorithm as explained in [32]. In the literature [32, 88], a classical symmetric splitting scheme that has been considered is derived from the SOR splitting and as such is called *symmetric SOR* (SSOR). Denote by $\mathbf{M}_{\mathrm{SOR}}$ and $\mathbf{N}_{\mathrm{SOR}}$ the matrices involved in the SOR splitting such that $\mathbf{Q} = \mathbf{M}_{\mathrm{SOR}} - \mathbf{N}_{\mathrm{SOR}}$; see row 4 of Table 2. Then, for any $0 < \omega < 2$, the SSOR splitting is defined by $\mathbf{Q} = \mathbf{M}_{\mathrm{SSOR}} - \mathbf{N}_{\mathrm{SSOR}}$ with

$$\mathbf{M}_{\mathrm{SSOR}} = \frac{\omega}{2-\omega}\mathbf{M}_{\mathrm{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\mathrm{SOR}}^{\top} \quad \text{and} \quad \mathbf{N}_{\mathrm{SSOR}} = \frac{\omega}{2-\omega}\mathbf{N}_{\mathrm{SOR}}\mathbf{D}^{-1}\mathbf{N}_{\mathrm{SOR}}^{\top} .$$

By resorting to the one-to-one equivalence between linear solvers and Gibbs samplers, [31, 32] showed that the acceleration above via Chebyshev polynomials can be applied to Gibbs samplers based on a symmetric splitting. In this context, the main challenge when dealing with accelerated Gibbs samplers compared to accelerated linear solvers is the calibration of the noise covariance to ensure that the invariant distribution coincides with $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. For the sake of completeness, the pseudocode associated to an accelerated version of Algorithm 4.2 based on the SSOR splitting is detailed in Algorithm 4.3. Associated convergence results and numerical studies associated to this algorithm can be found in [31, 32].

Algorithm 4.3 Chebyshev accelerated SSOR sampler

Input: SSOR tuning parameter $0 < \omega < 2$, extreme eigenvalues $\lambda_{\min}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q})$, and $\lambda_{\max}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q})$ of $\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q}$, number T of iterations, initialization $\mathbf{w}^{(0)}$, diagonal $\mathbf{D}$ of $\mathbf{Q}$, and SOR splitting $\mathbf{Q} = \mathbf{M}_{\mathrm{SOR}} - \mathbf{N}_{\mathrm{SOR}}$.

Set $\mathbf{D}_\omega = (2/\omega - 1)\mathbf{D}$.
Set $\sqrt{\delta} = (\lambda_{\max}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q}) - \lambda_{\min}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q}))/4$.
Set $\tau = 2/(\lambda_{\max}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q}) + \lambda_{\min}(\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{Q}))$.
Set $\beta = 2\tau$, $\alpha = 1$, $e = 2/\alpha - 1$, $c = (2/\tau - 1)e$, and $\kappa = \tau$.
Set $t = 1$.
while $t \leq T$ **do**
    Draw $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
    Solve $\mathbf{M}_{\mathrm{SOR}}\mathbf{x}_1 = \mathbf{M}_{\mathrm{SOR}}\mathbf{w}^{(t-1)} + \sqrt{e}\mathbf{D}_\omega^{1/2}\mathbf{z}_1 - \mathbf{Q}\mathbf{w}^{(t-1)}$ w.r.t. $\mathbf{x}_1$.
    Draw $\mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$.
    Solve $\mathbf{M}_{\mathrm{SOR}}^{\top}\mathbf{x}_2 = \mathbf{M}_{\mathrm{SOR}}^{\top}(\mathbf{x}_1 - \mathbf{w}^{(t-1)}) + \sqrt{c}\mathbf{D}_\omega^{1/2}\mathbf{z}_2 - \mathbf{Q}\mathbf{x}_1$ w.r.t. $\mathbf{x}_2$.
    if $t = 1$, **then**
        Set $\mathbf{w}^{(t)} = \alpha(\mathbf{w}^{(t-1)} + \tau\mathbf{x}_2)$.
        Set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\mu} + \mathbf{w}^{(t)}$.
    else
        Set $\mathbf{w}^{(t)} = \alpha(\mathbf{w}^{(t-1)} - \mathbf{w}^{(t-2)} + \tau\mathbf{x}_2) + \mathbf{w}^{(t-2)}$.
        Set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\mu} + \mathbf{w}^{(t)}$.
    end if
    Set $\beta = \frac{1}{1/\tau - \beta\delta}$, $\alpha = \frac{\beta}{\tau}$, $e = \frac{2\kappa(1-\alpha)}{\beta} + 1$, $c = \frac{2}{\tau} - 1 + (e-1)(\frac{1}{\tau} + \frac{1}{\kappa} - 1)$, and $\kappa = \beta + (1-\alpha)\kappa$.
    Set $t = t + 1$.
end while
return $\boldsymbol{\theta}^{(T)}$.

Algorithmic Efficiency. Similar to Algorithm 4.2, Algorithm 4.3 is exact in the sense that it admits $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ as an invariant distribution. The work [32] gave guidelines to choosing the truncation parameter T such that the error between the distribution

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

of $\boldsymbol{\theta}^{(T)}$ and $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ is sufficiently small. Regarding computation and storage, since triangular linear systems can be solved in $\mathcal{O}(d^2)$ flops by either back or forward substitution, Algorithm 4.3 admits a computational cost of $\mathcal{O}(Td^2)$ and a storage requirement of $\Theta(d)$.

4.1.2. Approximate Matrix Splitting. Motivated by efficiency and parallel computations, the authors in [9] and [54] proposed to relax exact MS and introduced two MCMC samplers whose invariant distributions are approximations of $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. First, in order to solve efficiently the linear system $\mathbf{M}\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \tilde{\mathbf{z}} + \mathbf{N}\boldsymbol{\theta}^{(t-1)}$ involved in step 4 of Algorithm 4.2, these approximate approaches consider MS schemes with diagonal matrices $\mathbf{M}$. For exact samplers, e.g., Richardson and Jacobi, we saw in the previous paragraph that such a convenient structure for $\mathbf{M}$ implies that the drawing of the Gaussian vector $\tilde{\mathbf{z}}$ becomes more demanding. To bypass this issue, approximate samplers draw Gaussian vectors $\tilde{\mathbf{z}}'$ with simpler covariance matrices $\tilde{\mathbf{M}}$ instead of $\mathbf{M}^\top + \mathbf{N}$. Again, attractive choices for $\tilde{\mathbf{M}}$ are diagonal matrices, since the associated sampling task then boils down to Algorithm 2.2. This yields Algorithm 4.4, which is highly amenable to parallelization since both the covariance matrix $\tilde{\mathbf{M}}$ of $\tilde{\mathbf{z}}'$ and the matrix $\mathbf{M}$ involved in the linear system to be solved are diagonal. Table 3 gathers the respective expressions of $\mathbf{M}$, $\mathbf{N}$, and $\tilde{\mathbf{M}}$ for the two approaches introduced in [54] and [9] and coined "Hogwild sampler" and "clone MCMC," respectively.

Algorithm 4.4 MCMC sampler based on approximate matrix splitting

Input: Number T of iterations, initialization $\boldsymbol{\theta}^{(0)}$, and splitting $\mathbf{Q} = \mathbf{M} - \mathbf{N}$.

1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\tilde{\mathbf{z}}' \sim \mathcal{N}(\mathbf{0}_d, \tilde{\mathbf{M}})$. ▷ $\tilde{\mathbf{M}} = \mathbf{D}$ or $2\,(\mathbf{D} + 2\omega\mathbf{I}_d)$; see Table 3.
4: Solve $\mathbf{M}\boldsymbol{\theta}^{(t)} = \mathbf{Q}\boldsymbol{\mu} + \tilde{\mathbf{z}}' + \mathbf{N}\boldsymbol{\theta}^{(t-1)}$.
5: Set $t = t + 1$.
6: **end while**
7: **return** $\boldsymbol{\theta}^{(T)}$.

Algorithmic Efficiency. Regarding sampling accuracy, the Hogwild sampler and clone MCMC define a Markov chain whose invariant distribution is Gaussian with the correct mean $\boldsymbol{\mu}$ but with precision matrix $\widetilde{\mathbf{Q}}_{\mathrm{MS}}$, where

$$\widetilde{\mathbf{Q}}_{\mathrm{MS}} = \begin{cases} \mathbf{Q}\left(\mathbf{I}_d - \mathbf{D}^{-1}(\mathbf{L} + \mathbf{L}^\top)\right) & \text{for the Hogwild sampler,} \\ \mathbf{Q}\left(\mathbf{I}_d - \frac{1}{2}(\mathbf{D} + 2\omega^{-1}\mathbf{I}_d)^{-1}\mathbf{Q}\right) & \text{for clone MCMC.} \end{cases}$$

Contrary to the Hogwild sampler, clone MCMC is able to sample exactly from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ in the asymptotic scenario $\omega \to 0$, since in this case $\widetilde{\mathbf{Q}}_{\mathrm{MS}} \to \mathbf{Q}$. While retaining a memory requirement of $\Theta(d)$, the induced approximation yields a highly parallelizable sampler. Indeed, compared to Algorithm 4.2, the computational complexities associated to step 3 and the solving of the triangular system in step 4 are decreased by an order of magnitude to $\mathcal{O}(d)$. Note that the overall computational complexity of step 4 is still $\mathcal{O}(d^2)$ because of the matrix-vector product $\mathbf{N}\boldsymbol{\theta}^{(t-1)}$.

4.2. Data Augmentation. Since the precision matrix $\mathbf{Q}$ has been assumed to be arbitrary, the MS schemes $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ in Table 2 were not motivated by its structure but rather by the computational efficiency of the associated samplers. Hence, inspired by efficient linear solvers, relevant choices for $\mathbf{M}$ and $\mathbf{N}$ given in Tables 2 and 3 have

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Table 3 *MS schemes for* $\mathbf{Q}$ *that can be used in Algorithm* 4.4. *The matrices* $\mathbf{D}$ *and* $\mathbf{L}$ *denote the diagonal and strictly lower triangular parts of* $\mathbf{Q}$, *respectively. The vector* $\tilde{\mathbf{z}}'$ *is the one appearing in step* 3 *of Algorithm* 4.4 *and* $\omega > 0$ *is a tuning parameter controlling the bias of those methods. Sufficient conditions to guarantee* $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$ *are given in* [9, 54].

Sampler	$\mathbf{M}$	$\mathbf{N}$	$\mathrm{cov}(\tilde{\mathbf{z}}') = \tilde{\mathbf{M}}$
Hogwild with blocks of size 1 [54]	$\mathbf{D}$	$-\mathbf{L} - \mathbf{L}^\top$	$\mathbf{D}$
Clone MCMC [9]	$\mathbf{D} + 2\omega\mathbf{I}_d$	$2\omega\mathbf{I}_d - \mathbf{L} - \mathbf{L}^\top$	$2\,(\mathbf{D} + 2\omega\mathbf{I}_d)$

been considered. Another line of search explores schemes specifically dedicated to precision matrices $\mathbf{Q}$ of the form

$$\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2 \,, \tag{4.3}$$

where, contrary to the MS schemes discussed in the previous section, the two matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are not chosen by the user but result directly from the statistical model under consideration. In particular, such situations arise when deriving hierarchical Bayesian models (see, e.g., [50, 78, 92]). By capitalizing on possible specific structures of $\{\mathbf{Q}_i\}_{i\in[2]}$, it may be desirable to separate $\mathbf{Q}_1$ and $\mathbf{Q}_2$ into two different hopefully simpler steps of a Gibbs sampler. To this purpose, this section discusses *data augmentation* (DA) approaches which introduce one (or several) auxiliary variables $\mathbf{u} \in \mathbb{R}^k$ such that the joint distribution of the couple $(\boldsymbol{\theta}, \mathbf{u})$ yields simple conditional distributions, and thus sampling steps within a Gibbs sampler [9, 66, 67, 104]. Then a straightforward marginalization of the auxiliary variable $\mathbf{u}$ permits us to retrieve the target distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, either exactly or in an asymptotic regime depending on the nature of the DA scheme. Both exact and approximate DA methods have been proposed.

4.2.1. Exact Data Augmentation. This section reviews some exact DA approaches to obtaining samples from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. The term *exact* here means that the joint distribution of $(\boldsymbol{\theta}, \mathbf{u})$ admits a density $\pi(\boldsymbol{\theta}, \mathbf{u})$ that satisfies almost surely

$$\int_{\mathbb{R}^k} \pi(\boldsymbol{\theta}, \mathbf{u}) = \pi(\boldsymbol{\theta}) \tag{4.4}$$

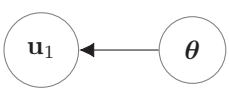
and yields proper marginal distributions. Figure 4 describes the directed acyclic graphs (DAGs) associated with two hierarchical models proposed in [66, 67] to decouple $\mathbf{Q}_1$ from $\mathbf{Q}_2$ by involving auxiliary variables. In what follows, we detail the motivations behind these two DA schemes. Among the two matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ involved in the composite precision matrix $\mathbf{Q}$, without loss of generality, we assume that $\mathbf{Q}_2$ presents a particular and simpler structure (e.g., diagonal or circulant) than $\mathbf{Q}_1$. We want now to benefit from this structure by leveraging the efficient sampling schemes previously discussed in subsection 2.2 and well suited to handling a Gaussian distribution with a precision matrix only involving $\mathbf{Q}_2$. This is the aim of the first DA model called EDA, which introduces the joint distribution with p.d.f.

$$\pi(\boldsymbol{\theta}, \mathbf{u}_1) \propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{\theta} - \boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{\theta} - \boldsymbol{\mu}) + (\mathbf{u}_1 - \boldsymbol{\theta})^\top \mathbf{R}(\mathbf{u}_1 - \boldsymbol{\theta})\right]\right) \,, \tag{4.5}$$

with $\mathbf{R} = \omega^{-1}\mathbf{I}_d - \mathbf{Q}_1$ and $0 < \omega < \|\mathbf{Q}_1\|^{-1}$, where $\|\cdot\|$ is the spectral norm. The resulting Gibbs sampler (see Algorithm 4.5) relies on two conditional Gaussian sampling steps whose associated conditional distributions are detailed in Table 4. This

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

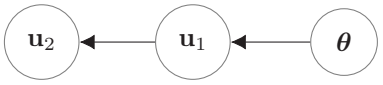
Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms



$$\boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{Q}^{-1}\right)$$

$$\mathbf{u}_1 \mid \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\theta}, \mathbf{R}^{-1}\right),\ \mathbf{R} = \omega^{-1}\mathbf{I}_d - \mathbf{Q}_1$$

(a) Hierarchical EDA model



$$\boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{Q}^{-1}\right)$$

$$\mathbf{u}_1 \mid \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\theta}, \mathbf{R}^{-1}\right)$$

$$\mathbf{u}_2 \mid \mathbf{u}_1 \sim \mathcal{N}\left(\mathbf{G}_1\mathbf{u}_1, \boldsymbol{\Lambda}_1^{-1}\right)$$

(b) Hierarchical GEDA model

Fig. 4 *Hierarchical models proposed in* [66, 67]*, where* ω *is such that* $0 < \omega < \|\mathbf{Q}_1\|^{-1}$.

Table 4 *Conditional probability distributions of* $\boldsymbol{\theta} \mid \mathbf{u}_1$, $\mathbf{u}_1 \mid \boldsymbol{\theta}, \mathbf{u}_2$, *and* $\mathbf{u}_2 \mid \mathbf{u}_1$ *for the exact DA schemes detailed in subsection* 4.2.1. *The parameter* ω *is such that* $0 < \omega < \|\mathbf{Q}_1\|^{-1}$. *For simplicity, the conditioning is notationally omitted.*

Sampler	$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1})$	$\mathbf{u}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_1}, \mathbf{Q}_{\mathbf{u}_1}^{-1})$	$\mathbf{u}_2 \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_2}, \mathbf{Q}_{\mathbf{u}_2}^{-1})$
EDA	$\mathbf{Q}_{\boldsymbol{\theta}} = \omega^{-1}\mathbf{I}_d + \mathbf{Q}_2$	$\mathbf{Q}_{\mathbf{u}_1} = \mathbf{R}$	-
	$\boldsymbol{\mu}_{\boldsymbol{\theta}} = \mathbf{Q}_{\boldsymbol{\theta}}^{-1}\left(\mathbf{R}\mathbf{u}_1 + \mathbf{Q}\boldsymbol{\mu}\right)$	$\boldsymbol{\mu}_{\mathbf{u}_1} = \boldsymbol{\theta}$	-
GEDA	$\mathbf{Q}_{\boldsymbol{\theta}} = \omega^{-1}\mathbf{I}_d + \mathbf{Q}_2$	$\mathbf{Q}_{\mathbf{u}_1} = \omega^{-1}\mathbf{I}_d$	$\mathbf{Q}_{\mathbf{u}_2} = \boldsymbol{\Lambda}_1$
	$\boldsymbol{\mu}_{\boldsymbol{\theta}} = \mathbf{Q}_{\boldsymbol{\theta}}^{-1}(\mathbf{R}\mathbf{u}_1 + \mathbf{Q}\boldsymbol{\mu})$	$\boldsymbol{\mu}_{\mathbf{u}_1} = \boldsymbol{\theta} - \omega(\mathbf{Q}_1\boldsymbol{\theta} - \mathbf{G}_1^\top\boldsymbol{\Lambda}_1^{-1}\mathbf{u}_2)$	$\boldsymbol{\mu}_{\mathbf{u}_2} = \mathbf{G}_1\mathbf{u}_1$

Algorithm 4.5 Gibbs sampler based on exact data augmentation (G)EDA

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$, $\mathbf{u}_1^{(0)}$.

Set $t = 1$.
while $t \leq T$ **do**
  Draw $\mathbf{u}_2^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_2}, \mathbf{Q}_{\mathbf{u}_2}^{-1})$. ▷ Only if GEDA is considered.
  Draw $\mathbf{u}_1^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}_1}, \mathbf{Q}_{\mathbf{u}_1}^{-1})$.
  Draw $\boldsymbol{\theta}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1})$.
  Set $t = t + 1$.
end while
return $\boldsymbol{\theta}^{(T)}$.

scheme has the great advantage of decoupling the two precision matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ since they are not simultaneously involved in either of the two steps. In particular, introducing the auxiliary variable $\mathbf{u}_1$ permits us to remove the dependence in $\mathbf{Q}_1$ when defining the precision matrix of the conditional distribution of $\boldsymbol{\theta}$. While efficient sampling from this conditional is possible, we have to ensure that sampling the auxiliary variable $\mathbf{u}_1$ can be achieved with a reasonable computational cost. Again, if $\mathbf{Q}_1$ presents a nice structure, the specific approaches reviewed in subsection 2.2 can be employed. If this is not the case, the authors of [66, 67] proposed a generalization of EDA, called GEDA, to simplify the whole Gibbs sampling procedure when $\mathbf{Q}$ arises

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

from a hierarchical Bayesian model. In such models, $\mathbf{Q}_1$, as a fortiori $\mathbf{Q}_2$, naturally admits an explicit decomposition written as $\mathbf{Q}_1 = \mathbf{G}_1^\top \mathbf{\Lambda}_1 \mathbf{G}_1$, where $\mathbf{\Lambda}_1$ is a positive-definite (and very often diagonal) matrix. By building on this explicit decomposition, GEDA introduces an additional auxiliary variable $\mathbf{u}_2$ such that the augmented p.d.f. is

$$\begin{aligned}\pi(\boldsymbol{\theta}, \mathbf{u}_1, \mathbf{u}_2) \propto & \exp\left(-\frac{1}{2}\left[(\boldsymbol{\theta}-\boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{\theta}-\boldsymbol{\mu}) + (\mathbf{u}_1-\boldsymbol{\theta})^\top \mathbf{R}(\mathbf{u}_1-\boldsymbol{\theta})\right]\right) \\ & \times \exp\left(-\frac{1}{2}(\mathbf{u}_2 - \mathbf{G}_1\mathbf{u}_1)^\top \mathbf{\Lambda}_1 (\mathbf{u}_2 - \mathbf{G}_1\mathbf{u}_1)\right) . \end{aligned} \tag{4.6}$$

The associated joint distribution yields conditional Gaussian distributions with diagonal covariance matrices for both $\mathbf{u}_1$ and $\mathbf{u}_2$ that can be sampled efficiently with Algorithm 2.2; see Table 4.

Algorithmic Efficiency. First, both EDA and GEDA admit $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ as invariant distribution and hence are exact. Regarding EDA, since the conditional distribution of $\mathbf{u}_1 \mid \boldsymbol{\theta}$ might admit an arbitrary precision matrix in the worst-case scenario, its computational and storage complexities are $\mathcal{O}(KTd^2)$ and $\Theta(d)$, where K is a truncation parameter associated to one of the algorithms reviewed in section 3. On the other hand, GEDA benefits from an additional DA which yields the reduced computational and storage requirements of $\mathcal{O}(Td^2)$ and $\Theta(d)$.

4.2.2. Approximate Data Augmentation. An approximate DA scheme inspired by variable-splitting approaches in optimization [2, 3, 15] was proposed in [104]. This framework, also called asymptotically exact DA (AXDA) [105], was initially introduced to deal with any target distribution, not limited to Gaussian distributions; it therefore a fortiori applies to them as well. An auxiliary variable $\mathbf{u} \in \mathbb{R}^d$ is introduced such that the joint p.d.f. of $(\boldsymbol{\theta}, \mathbf{u})$ is

$$\pi(\boldsymbol{\theta}, \mathbf{u}) \propto \exp\left(-\frac{1}{2}\left[(\boldsymbol{\theta}-\boldsymbol{\mu})^\top \mathbf{Q}_2(\boldsymbol{\theta}-\boldsymbol{\mu}) + (\mathbf{u}-\boldsymbol{\mu})^\top \mathbf{Q}_1(\mathbf{u}-\boldsymbol{\mu}) + \frac{\|\mathbf{u}-\boldsymbol{\theta}\|^2}{\omega}\right]\right) , \tag{4.7}$$

where $\omega > 0$. The main idea behind (4.7) is to replicate the variable of interest $\boldsymbol{\theta}$ in order to sample via a Gibbs sampling strategy two different random variables $\mathbf{u}$ and $\boldsymbol{\theta}$ with covariance matrices involving, separately, $\mathbf{Q}_1$ and $\mathbf{Q}_2$. This algorithm, coined the split Gibbs sampler (SGS), is detailed in Algorithm 4.6 and sequentially draws from the conditional distributions

$$\mathbf{u} \mid \boldsymbol{\theta} \sim \mathcal{N}\left((\omega^{-1}\mathbf{I}_d + \mathbf{Q}_1)^{-1}(\omega^{-1}\boldsymbol{\theta} + \mathbf{Q}_1\boldsymbol{\mu}), (\omega^{-1}\mathbf{I}_d + \mathbf{Q}_1)^{-1}\right) , \tag{4.8}$$

$$\boldsymbol{\theta} \mid \mathbf{u} \sim \mathcal{N}\left((\omega^{-1}\mathbf{I}_d + \mathbf{Q}_2)^{-1}(\omega^{-1}\mathbf{u} + \mathbf{Q}_2\boldsymbol{\mu}), (\omega^{-1}\mathbf{I}_d + \mathbf{Q}_2)^{-1}\right) . \tag{4.9}$$

Again, this approach has the great advantage of decoupling the two precision matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ defining $\mathbf{Q}$ since they are not simultaneously involved in either of the two steps of the Gibbs sampler. In [66], the authors showed that exact DA schemes (i.e., EDA and GEDA) generally outperform AXDA as far as Gaussian sampling is concerned. This was expected since the AXDA framework proposed is not specifically designed for Gaussian targets but for a wide family of distributions.

Algorithmic Efficiency. The sampling efficiency of Algorithm 4.6 depends upon the parameter ω which controls the strength of the coupling between $\mathbf{u}$ and $\boldsymbol{\theta}$ as well as the bias-variance trade-off of this method; it yields exact sampling when $\omega \to 0$. Indeed,

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

the marginal distribution of $\boldsymbol{\theta}$ under the joint distribution with density defined in (4.7) is a Gaussian with the correct mean $\boldsymbol{\mu}$ but with an approximate precision matrix $\widetilde{\mathbf{Q}}_{\mathrm{DA}}$ that admits the closed-form expression

$$\widetilde{\mathbf{Q}}_{\mathrm{DA}} = \mathbf{Q}_2 + \left(\mathbf{Q}_1^{-1} + \omega \mathbf{I}_d\right)^{-1} . \tag{4.10}$$

In the worst-case scenario where $\mathbf{Q}_1$ is arbitrary, sampling from the conditional distribution (4.8) can be performed with an iterative algorithm running K iterations such as those reviewed in section 3. Hence Algorithm 4.6 admits the same computational and storage complexities as EDA (see Algorithm 4.5), that is, $\mathcal{O}(KTd^2)$ and $\Theta(d)$.

Algorithm 4.6 Gibbs sampler based on approximate data augmentation

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$.
1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\mathbf{u}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{u}}, \mathbf{Q}_{\mathbf{u}}^{-1})$ in (4.8).
4: Draw $\boldsymbol{\theta}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1})$ in (4.9).
5: Set $t = t + 1$.
6: **end while**
7: **return** $\boldsymbol{\theta}^{(T)}$.

5. A Unifying Revisit of Gibbs Samplers via a Stochastic Version of the PPA.

Sections 3 and 4 showed that numerous approaches have been proposed to sample from a possibly high-dimensional Gaussian distribution with density (2.1). This section proposes to unify these approaches within a general Gaussian simulation framework which is actually a stochastic counterpart of the celebrated PPA in optimization [90]. This viewpoint will shed new light on the connections among the reviewed simulation-based algorithms, and in particular among Gibbs samplers.

5.1. A Unifying Proposal Distribution.

The approaches described in section 4 use surrogate probability distributions (e.g., conditional or approximate distributions) to make Gaussian sampling easier. In what follows, we show that most of these surrogate distributions can be put under a common umbrella by considering the density

$$\kappa(\boldsymbol{\theta}, \mathbf{u}) \propto \pi(\boldsymbol{\theta}) \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{u})^\top \mathbf{R}(\boldsymbol{\theta} - \mathbf{u})\right) , \tag{5.1}$$

where $\mathbf{u} \in \mathbb{R}^d$ is an additional (auxiliary) variable and $\mathbf{R} \in \mathbb{R}^{d \times d}$ is a symmetric matrix acting as a preconditioner such that κ defines a proper density on an appropriate state space. More precisely, in what follows, depending on the definition of the variable $\mathbf{u}$, the probability density κ in (5.1) will refer to either a joint p.d.f. $\pi(\boldsymbol{\theta}, \mathbf{u})$ or a conditional probability density $\pi(\boldsymbol{\theta} \mid \mathbf{u})$. Contrary to the MCMC samplers detailed in section 4, the methods described in section 3 do not use explicit surrogate distributions to simplify the sampling procedure. Instead, they directly perturb deterministic approaches from numerical linear algebra without explicitly defining a simpler surrogate distribution at each iteration. This feature can be encoded with the choice $\mathbf{R} \to \mathbf{0}_{d \times d}$ so that these methods can be described by this unifying model as well. Then the main motivation for using the surrogate density κ is to *precondition* the initial p.d.f. π to end up with simpler sampling steps as in section 4.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

5.2. Revisiting MCMC Sampling Approaches. This section builds on the probability kernel density (5.1) to revisit, unify, and extend the exact and approximate approaches reviewed in section 4. We emphasize that exact approaches indeed target the distribution of interest $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, while approximate approaches only target an approximation of $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$.

5.2.1. From Exact Data Augmentation to Exact Matrix Splitting. We assume here that the variable $\mathbf{u}$ refers to an auxiliary variable such that the joint distribution of the couple $(\boldsymbol{\theta}, \mathbf{u})$ has a density given by $\pi(\boldsymbol{\theta}, \mathbf{u}) \triangleq \kappa(\boldsymbol{\theta}, \mathbf{u})$. In addition, here we restrict $\mathbf{R}$ to be positive definite. It follows that

$$\int_{\mathbb{R}^d} \pi(\boldsymbol{\theta}, \mathbf{u}) \mathrm{d}\mathbf{u} = Z^{-1} \pi(\boldsymbol{\theta}) \int_{\mathbb{R}^d} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \mathbf{u})^\top \mathbf{R}(\boldsymbol{\theta} - \mathbf{u})\right) \mathrm{d}\mathbf{u} = \pi(\boldsymbol{\theta}) \tag{5.2}$$

holds almost surely with $Z = \det(\mathbf{R})^{-1/2}(2\pi)^{d/2}$. Hence, the joint density (5.1) yields an exact DA scheme whatever the choice of the positive-definite matrix $\mathbf{R}$. We will show that the exact DA approaches described in Algorithm 4.5 precisely fit the proposed generic framework with a specific choice for the preconditioning matrix $\mathbf{R}$. We will then extend this class of exact DA approaches and show a one-to-one equivalence between Gibbs samplers based on exact MS (see subsection 4.1.1) and those based on exact DA (see subsection 4.2.1).

To this end, we start by making the change of variable $\mathbf{v} = \mathbf{R}\mathbf{u}$. Combined with the joint probability density (5.1), this yields the following two conditional probability distributions:

$$\mathbf{v} \mid \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{R}\boldsymbol{\theta}, \mathbf{R}\right) , \tag{5.3}$$

$$\boldsymbol{\theta} \mid \mathbf{v} \sim \mathcal{N}\left((\mathbf{Q} + \mathbf{R})^{-1}(\mathbf{v} + \mathbf{Q}\boldsymbol{\mu}), (\mathbf{Q} + \mathbf{R})^{-1}\right) . \tag{5.4}$$

As emphasized in subsection 5.1, the aim of introducing the preconditioning matrix $\mathbf{R}$ is to yield simpler sampling steps. In the general case where $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$, with $\mathbf{Q}_1$ and $\mathbf{Q}_2$ two matrices that cannot be easily handled jointly (e.g., because they are not diagonalized in the same basis), an attractive option is $\mathbf{R} = \omega^{-1}\mathbf{I}_d - \mathbf{Q}_1$. Indeed, this choice ensures that $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are separated and are not simultaneously involved in either of the two conditional sampling steps. Note that this choice yields the EDA scheme already discussed in subsection 4.2.1; see Table 4. Now we relate this exact DA scheme to an exact MS scheme. By rewriting the Gibbs sampling steps associated with the conditional distributions (5.3) and (5.4) as an autoregressive process of order 1 w.r.t. $\boldsymbol{\theta}$ [14], it follows that an equivalent sampling strategy is

$$\tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{Q}\boldsymbol{\mu}, 2\mathbf{R} + \mathbf{Q}\right) , \tag{5.5}$$

$$\boldsymbol{\theta}^{(t)} = (\mathbf{Q} + \mathbf{R})^{-1}\left(\tilde{\mathbf{z}} + \mathbf{R}\boldsymbol{\theta}^{(t-1)}\right) . \tag{5.6}$$

Defining $\mathbf{M} = \mathbf{Q} + \mathbf{R}$ and $\mathbf{N} = \mathbf{R}$, or equivalently $\mathbf{Q} = \mathbf{M} - \mathbf{N}$, this yields

$$\tilde{\mathbf{z}} \sim \mathcal{N}\left(\mathbf{Q}\boldsymbol{\mu}, \mathbf{M}^\top + \mathbf{N}\right) , \tag{5.7}$$

$$\boldsymbol{\theta}^{(t)} = \mathbf{M}^{-1}\left(\tilde{\mathbf{z}} + \mathbf{N}\boldsymbol{\theta}^{(t-1)}\right) , \tag{5.8}$$

which boils down to the Gibbs sampler based on exact MS discussed in subsection 4.1.1 (see Algorithm 4.2).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Table 5 *Equivalence relations between exact DA and exact MS approaches. The matrices* $\mathbf{Q}_1$ *and* $\mathbf{Q}_2$ *are such that* $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$*. The matrix* $\mathbf{D}_1$ *denotes the diagonal part of* $\mathbf{Q}_1$*, and* $\omega > 0$ *is a positive scalar ensuring the positive definiteness of* $\mathbf{R}$*. Bold acronyms refer to novel samplers derived from the proposed unifying framework.*

$\mathbf{R} = \mathrm{cov}(\mathbf{v}\|\boldsymbol{\theta})$	$(\mathbf{Q}+\mathbf{R})^{-1} = \mathrm{cov}(\boldsymbol{\theta}\|\mathbf{v})$	$\mathbf{M}^\top + \mathbf{N} = \mathrm{cov}(\tilde{\mathbf{z}})$	DA sampler	MS sampler
$\frac{\mathbf{I}_d}{\omega} - \mathbf{Q}_1$	$\left(\frac{\mathbf{I}_d}{\omega} + \mathbf{Q}_2\right)^{-1}$	$\frac{2\mathbf{I}_d}{\omega} + \mathbf{Q}_2 - \mathbf{Q}_1$	EDA [66]	Richardson [32]
$\frac{\mathbf{D}_1}{\omega} - \mathbf{Q}_1$	$\left(\frac{\mathbf{D}_1}{\omega} + \mathbf{Q}_2\right)^{-1}$	$\frac{2\mathbf{D}_1}{\omega} + \mathbf{Q}_2 - \mathbf{Q}_1$	**EDAJ**	Jacobi [32]

To illustrate the relevance of this rewriting when considering the case of two matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ that cannot be efficiently handled in the same basis, Table 5 presents two possible choices of $\mathbf{R}$ which relate two MS strategies to their DA counterparts. First, one particular choice of $\mathbf{R}$ (row 1 of Table 5) shows directly that the Richardson MS sampler can be rewritten as the EDA sampler. More precisely, the autoregressive process of order 1 w.r.t. $\boldsymbol{\theta}$ defined by EDA yields a variant of the Richardson sampler. This finding relates two different approaches proposed by authors from distinct communities (numerical algebra and signal processing). Second, the proposed unifying framework also permits us to go beyond existing approaches by proposing a novel exact DA approach via a specific choice for the precision matrix $\mathbf{R}$ driven by an existing MS method. Indeed, following the same rewriting trick with another particular choice of $\mathbf{R}$ (row 2 of Table 5), an exact DA scheme can be easily derived from the Jacobi MS approach. To our knowledge, this novel DA method, referred to as EDAJ in the table, has not been documented in the existing literature.

Finally, the table reports two particular choices of $\mathbf{R}$ which lead to revisiting existing MS and/or DA methods. It is worth noting that other relevant choices might be possible, which would allow one to derive new exact DA and MS methods or to draw further analogies between existing approaches. Note also that Table 5 shows the main benefit of an exact DA scheme over its MS counterpart thanks to the decoupling between $\mathbf{Q}_1$ and $\mathbf{Q}_2$ into two separate simulation steps. This feature can be directly observed by comparing the two first columns of Table 5 with the third column.

5.2.2. From Approximate Matrix Splitting to Approximate Data Augmentation. We now build on the proposed unifying proposal (5.1) to extend the class of samplers based on approximate MS and reviewed in subsection 4.1.2. With some abuse of notation, the variable $\mathbf{u}$ in (5.1) now refers to an iterate associated to $\boldsymbol{\theta}$. More precisely, let us define $\mathbf{u} = \boldsymbol{\theta}^{(t-1)}$ to be the current iterate within an MCMC algorithm and κ to be

$$\kappa(\boldsymbol{\theta}, \mathbf{u}) \triangleq p\left(\boldsymbol{\theta}|\mathbf{u} = \boldsymbol{\theta}^{(t-1)}\right) \propto \pi(\boldsymbol{\theta}) \exp\left(-\frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right)^\top \mathbf{R}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right)\right). \tag{5.9}$$

Readers familiar with MCMC algorithms will recognize in (5.9) a proposal density that can be used within Metropolis–Hastings schemes [87]. However, unlike the usual random-walk algorithm which considers the Gaussian proposal distribution $\mathcal{N}(\boldsymbol{\theta}^{(t-1)}, \lambda \mathbf{I}_d)$ with $\lambda > 0$, the originality of (5.9) is to define the proposal by combining the Gaussian target density π with a term that is equal to a Gaussian kernel density when $\mathbf{R}$ is positive definite. If we always accept the proposed sample obtained from (5.9) without any correction, that is, $\boldsymbol{\theta}^{(t)} = \widetilde{\boldsymbol{\theta}} \sim P(\cdot \mid \mathbf{u} = \boldsymbol{\theta}^{(t-1)})$ with density

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

(5.9), this directly implies that the associated Markov chain converges in distribution toward a Gaussian random variable with distribution $\mathcal{N}(\boldsymbol{\mu}, \widetilde{\mathbf{Q}}^{-1})$ with the correct mean $\boldsymbol{\mu}$ but with precision matrix

$$\widetilde{\mathbf{Q}} = \mathbf{Q}\left(\mathbf{I}_d + (\mathbf{R}+\mathbf{Q})^{-1}\mathbf{R}\right) . \tag{5.10}$$

This algorithm is detailed in Algorithm 5.1. Note again that one can obtain samples from the initial target distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ by replacing step 4 with an acceptance/rejection step; see [87] for details.

Algorithm 5.1 MCMC sampler based on (5.9)

Input: Number T of iterations and initialization $\boldsymbol{\theta}^{(0)}$.
1: Set $t = 1$.
2: **while** $t \leq T$ **do**
3: Draw $\widetilde{\boldsymbol{\theta}} \sim P\left(\cdot \mid \mathbf{u} = \boldsymbol{\theta}^{(t-1)}\right)$ in (5.9).
4: Set $\boldsymbol{\theta}^{(t)} = \widetilde{\boldsymbol{\theta}}$.
5: Set $t = t + 1$.
6: **end while**
7: **return** $\boldsymbol{\theta}^{(T)}$.

Moreover, the instance (5.9) of (5.1) paves the way to an extended class of samplers based on approximate MS. More precisely, the draw of a proposed sample $\widetilde{\boldsymbol{\theta}}$ from (5.9) can be replaced by the following two-step sampling procedure:

$$\tilde{\mathbf{z}}' \sim \mathcal{N}\left(\mathbf{Q}\boldsymbol{\mu}, \mathbf{R}+\mathbf{Q}\right) , \tag{5.11}$$

$$\boldsymbol{\theta}^{(t)} = (\mathbf{Q}+\mathbf{R})^{-1}\left(\tilde{\mathbf{z}}' + \mathbf{R}\boldsymbol{\theta}^{(t-1)}\right) . \tag{5.12}$$

The MS form with $\mathbf{M} = \mathbf{Q} + \mathbf{R}$, $\mathbf{N} = \mathbf{R}$ is

$$\tilde{\mathbf{z}}' \sim \mathcal{N}\left(\mathbf{Q}\boldsymbol{\mu}, \mathbf{M}\right) , \tag{5.13}$$

$$\boldsymbol{\theta}^{(t)} = \mathbf{M}^{-1}\left(\tilde{\mathbf{z}}' + \mathbf{N}\boldsymbol{\theta}^{(t-1)}\right) . \tag{5.14}$$

This recursion defines an extended class of approximate MS-based samplers and encompasses the Hogwild sampler reviewed in subsection 4.1.2 by taking $\mathbf{R} = -\mathbf{L} - \mathbf{L}^\top$. In addition to the existing Hogwild approach, Table 6 lists two new approximate MS approaches resulting from specific choices of the preconditioning matrix $\mathbf{R}$. They are coined *approximate* Richardson and Jacobi samplers since the expressions for $\mathbf{M}$ and $\mathbf{N}$ are very similar to those associated to their exact counterparts; see Table 2. For those two samplers, note that the approximate precision matrix $\widetilde{\mathbf{Q}}$ tends toward $2\mathbf{Q}$ in the asymptotic regime $\omega \to 0$. Indeed, for the approximate Jacobi sampler, we have

$$\begin{aligned}\widetilde{\mathbf{Q}} &= \mathbf{Q}\left(\mathbf{I}_d + \omega\left(\frac{\mathbf{I}_d}{\omega} - \mathbf{Q}\right)\right) \\ &= \mathbf{Q}\left(2\mathbf{I}_d - \omega\mathbf{Q}\right) \\ &\underset{\omega\to 0}{\to} 2\mathbf{Q} .\end{aligned}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Table 6 *Extended class of Gibbs samplers based on approximate MS with* $\mathbf{Q} = \mathbf{M} - \mathbf{N}$ *with* $\mathbf{N} = \mathbf{R}$ *and approximate DA. The matrices* $\mathbf{D}$ *and* $\mathbf{L}$ *denote the diagonal and strictly lower triangular parts of* $\mathbf{Q}$*, respectively.* ω *is a positive scalar. Bold names and acronyms refer to novel samplers derived from the proposed unifying framework.*

$\frac{1}{2}\mathbf{M} = \text{cov}(\mathbf{v}'\|\boldsymbol{\theta})$	$\frac{1}{2}\mathbf{M}^{-1} = \text{cov}(\boldsymbol{\theta}\|\mathbf{v}')$	$\mathbf{M} = \text{cov}(\tilde{\mathbf{z}}')$	MS sampler	DA sampler
$\frac{1}{2}\mathbf{D}$	$\frac{1}{2}\mathbf{D}^{-1}$	$\mathbf{D}$	Hogwild [54]	**ADAH**
$\frac{\mathbf{I}_d}{2\omega}$	$\frac{\omega\mathbf{I}_d}{2}$	$\frac{\mathbf{I}_d}{\omega}$	**approx. Richardson**	**ADAR**
$\frac{\mathbf{D}}{2\omega}$	$\frac{\omega\mathbf{D}^{-1}}{2}$	$\frac{\mathbf{D}}{\omega}$	**approx. Jacobi**	**ADAJ**

In order to retrieve the original precision matrix $\mathbf{Q}$ when $\omega \to 0$, [9] proposed an approximate DA strategy that can be related to the work of [105].

In subsection 5.2.1, we showed that exact DA approaches can be rewritten to recover exact MS approaches. In what follows, we will take the opposite path to show that approximate MS approaches admit approximate DA counterparts that are highly amenable to distributed and parallel computations. Using the fact that $\mathbf{z}' = \mathbf{Q}\boldsymbol{\mu} + \mathbf{z}_1 + (\mathbf{Q}+\mathbf{R})\mathbf{z}_2$, where $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}_d, \frac{1}{2}(\mathbf{R}+\mathbf{Q}))$ and $\mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}_d, \frac{1}{2}(\mathbf{R}+\mathbf{Q})^{-1})$, the recursion (5.14) can be written equivalently as

$$\widetilde{\boldsymbol{\theta}} = (\mathbf{Q}+\mathbf{R})^{-1}\left(\mathbf{Q}\boldsymbol{\mu} + \mathbf{R}\boldsymbol{\theta}^{(t-1)} + \mathbf{z}_1\right) + \mathbf{z}_2 \; .$$

By introducing an auxiliary variable $\mathbf{v}'$ defined by $\mathbf{v}' = \mathbf{R}\boldsymbol{\theta}^{(t-1)} + \mathbf{z}_1$, the resulting two-step Gibbs sampling relies on the conditional sampling steps

$$\begin{aligned}
&\mathbf{v}' \mid \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{R}\boldsymbol{\theta}, \frac{1}{2}(\mathbf{R}+\mathbf{Q})\right) \; , \\
&\boldsymbol{\theta} \mid \mathbf{v}' \sim \mathcal{N}\left((\mathbf{Q}+\mathbf{R})^{-1}(\mathbf{v}'+\mathbf{Q}\boldsymbol{\mu}), \frac{1}{2}(\mathbf{R}+\mathbf{Q})^{-1}\right) \; ,
\end{aligned}$$

and targets the joint distribution with density $\pi(\boldsymbol{\theta}, \mathbf{v}')$. Compared to the exact DA approaches reviewed in subsection 4.2.1, the sampling difficulty associated to each conditional sampling step is the same and only driven by the structure of the matrix $\mathbf{M} = \mathbf{R} + \mathbf{Q}$. In particular, this matrix becomes diagonal for the three specific choices listed in Table 6. These choices lead to three new sampling schemes that we name ADAH, ADAR, and ADAJ since they represent the DA counterparts of the approximate MS samplers discussed above. Interestingly, these DA schemes naturally emerge here without assuming any explicit decomposition $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ or including an additional auxiliary variable (as in GEDA). Finally, as previously highlighted, when compared to their exact counterparts, these DA schemes have the great advantage of leading to Gibbs samplers suited for parallel computations, hence simplifying the sampling procedure.

5.3. Gibbs Samplers as Stochastic Versions of the PPA. This section aims to draw new connections between optimization and the sampling approaches discussed in this paper. In particular, we will focus on the proximal point algorithm (PPA) [90]. After briefly presenting this optimization method, we will show that the Gibbs

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

samplers based on the proposal (5.9) can be interestingly interpreted as stochastic counterparts of the PPA. Let us assume here that $\mathbf{R}$ is positive semidefinite and define the *weighted* norm w.r.t. $\mathbf{R}$ for all $\boldsymbol{\theta} \in \mathbb{R}^d$ by

$$\|\boldsymbol{\theta}\|_{\mathbf{R}}^2 \triangleq \boldsymbol{\theta}^\top \mathbf{R} \boldsymbol{\theta} \,. \tag{5.15}$$

The Proximal Point Algorithm. The PPA [90] is an important and widely used method to find zeros of a maximal monotone operator K, that is, to solve problems of the following form:

$$\text{Find } \boldsymbol{\theta}^\star \in \mathcal{H} \text{ such that } \mathbf{0}_d \in \mathsf{K}(\boldsymbol{\theta}^\star) \,, \tag{5.16}$$

where $\mathcal{H}$ is a real Hilbert space. For simplicity, we will take here $\mathcal{H} = \mathbb{R}^d$ equipped with the usual Euclidean norm and focus on the particular case $\mathsf{K} = \partial f$, where f is a lower semicontinuous (l.s.c.), proper, coercive, and convex function and ∂ denotes the subdifferential operator; see Appendix B. In this case, the PPA is equivalent to the proximal minimization algorithm [68, 69] which aims at solving the following minimization problem:

$$\text{Find } \boldsymbol{\theta}^\star \in \mathbb{R}^d \text{ such that } \boldsymbol{\theta}^\star = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^d} f(\boldsymbol{\theta}) \,. \tag{5.17}$$

This algorithm is called the proximal point algorithm in reference to the work by Moreau [76]. For readability reasons, we refer the reader to Appendix B for details about this algorithm for a general operator K and to the comprehensive overview in [90] for more information.

Algorithm 5.2 Proximal point algorithm (PPA)

1: Choose an initial value $\boldsymbol{\theta}^{(0)}$, a positive semidefinite matrix $\mathbf{R}$, and a maximal number of iterations T.
2: Set $t = 1$.
3: **while** $t \leq T$ **do**
4: $\quad \boldsymbol{\theta}^{(t)} = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^d} f(\boldsymbol{\theta}) + \frac{1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)} \right\|_{\mathbf{R}}^2 .$
5: **end while**
6: **return** $\boldsymbol{\theta}^{(T)}$.

The PPA is detailed in Algorithm 5.2. Note that instead of directly minimizing the objective function f, the PPA adds a quadratic penalty term depending on the previous iterate $\boldsymbol{\theta}^{(t-1)}$ and then solves an approximation of the initial optimization problem at each iteration. This idea of successive approximations is exactly the deterministic counterpart of (5.9), which proposes a new sample based on successive approximations of the target density π via a Gaussian kernel with precision matrix $\mathbf{R}$. In fact, searching for the maximum a posteriori estimator under the proposal distribution $P(\cdot \mid \boldsymbol{\theta}^{(t-1)})$ with density $p(\cdot \mid \boldsymbol{\theta}^{(t-1)})$ in (5.9) boils down to solving

$$\operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^d} \underbrace{-\log \pi(\boldsymbol{\theta})}_{f(\boldsymbol{\theta})} + \frac{1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)} \right\|_{\mathbf{R}}^2 \,, \tag{5.18}$$

which coincides with step 4 in Algorithm 5.2 by taking $f = -\log \pi$. This emphasizes the tight connection between optimization and simulation that we have highlighted in previous sections.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

The PPA, the ADMM, and the Approximate Richardson Gibbs Sampler. An important motivation of the PPA is related to the *preconditioning* idea used in the unifying model proposed in (5.1). Indeed, the PPA has been extensively used within the alternating direction method of multipliers (ADMM) [15, 33, 40] as a preconditioner in order to avoid high-dimensional inversions [17, 19, 28, 58, 110]. The ADMM [15] is an optimization approach that solves the minimization problem in (5.17) when $g(\boldsymbol{\theta}) = g_1(\mathbf{A}\boldsymbol{\theta}) + g_2(\boldsymbol{\theta})$, $\mathbf{A} \in \mathbb{R}^{k\times d}$, via the iterative scheme

$$\mathbf{z}^{(t)} = \underset{\mathbf{z}\in\mathbb{R}^k}{\arg\min}\; g_1(\mathbf{z}) + \frac{1}{2\rho}\left\|\mathbf{z} - \mathbf{A}\boldsymbol{\theta}^{(t-1)} - \mathbf{u}^{(t-1)}\right\|^2, \tag{5.19}$$

$$\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\arg\min}\; g_2(\boldsymbol{\theta}) + \frac{1}{2\rho}\left\|\mathbf{A}\boldsymbol{\theta} - \mathbf{z}^{(t)} + \mathbf{u}^{(t-1)}\right\|^2, \tag{5.20}$$

$$\mathbf{u}^{(t)} = \mathbf{u}^{(t-1)} + \mathbf{A}\boldsymbol{\theta}^{(t)} - \mathbf{z}^{(t)}, \tag{5.21}$$

where $\mathbf{z} \in \mathbb{R}^k$ is a splitting variable, $\mathbf{u} \in \mathbb{R}^k$ is a scaled dual variable, and ρ is a positive penalty parameter. Without loss of generality,[4] let us assume that g_2 is a quadratic function, that is, for any $\boldsymbol{\theta} \in \mathbb{R}^d$, $g_2(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^\top(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})/2$. Even in this simple case, one can notice that the $\boldsymbol{\theta}$-update (5.20) involves a matrix $\mathbf{A}$ operating directly on $\boldsymbol{\theta}$, preluding an expensive inversion of a high-dimensional matrix associated to $\mathbf{A}$. To deal with such an issue, Algorithm 5.2 is considered to solve approximately the minimization problem in (5.20). The PPA applied to the minimization problem (5.20) reads

$$\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\arg\min}\; \frac{1}{2}(\boldsymbol{\theta}-\bar{\boldsymbol{\theta}})^\top(\boldsymbol{\theta}-\bar{\boldsymbol{\theta}}) + \frac{1}{2\rho}\left\|\mathbf{A}\boldsymbol{\theta} - \mathbf{z}^{(t)} + \mathbf{u}^{(t-1)}\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right\|_{\mathbf{R}}^2. \tag{5.22}$$

In order to draw some connections with (5.9), we set $\mathbf{Q} = \rho^{-1}\mathbf{A}^\top\mathbf{A} + \mathbf{I}_d$ and $\boldsymbol{\mu} = \mathbf{Q}^{-1}[\mathbf{A}^\top(\mathbf{u}^{(t-1)} - \mathbf{z}^{(t)})/\rho + \bar{\boldsymbol{\theta}}]$ and rewrite (5.22) as

$$\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\arg\min}\; \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu})^\top\mathbf{Q}(\boldsymbol{\theta} - \boldsymbol{\mu}) + \frac{1}{2}\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(t-1)}\right\|_{\mathbf{R}}^2. \tag{5.23}$$

Note that $(1/2)(\boldsymbol{\theta} - \boldsymbol{\mu})^\top\mathbf{Q}(\boldsymbol{\theta} - \boldsymbol{\mu})$ is the potential function associated to π in (2.1) and, as such, (5.23) can be seen as the deterministic counterpart of (5.9). By defining $\mathbf{R} = \omega^{-1}\mathbf{I}_d - \mathbf{Q}$, where $0 < \omega \leq \rho\left\|\mathbf{A}\right\|^{-2}$ ensures that $\mathbf{R}$ is positive semidefinite, the $\boldsymbol{\theta}$-update in (5.22) becomes (see Appendix B)

$$\boldsymbol{\theta}^{(t)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\arg\min}\frac{1}{2\omega}\left\|\boldsymbol{\theta} - \omega\left(\mathbf{R}\boldsymbol{\theta}^{(t-1)} + \mathbf{Q}\boldsymbol{\mu}\right)\right\|^2. \tag{5.24}$$

Note that (5.24) boils down to solving $\omega^{-1}\boldsymbol{\theta} = \mathbf{R}\boldsymbol{\theta}^{(t-1)} + \mathbf{Q}\boldsymbol{\mu}$, which is exactly the deterministic counterpart of the approximate Richardson Gibbs sampler in Table 6. This highlights further the tight links between the proposed unifying framework and the use of the PPA in the optimization literature. It also paves the way to novel sampling methods inspired by optimization approaches which are not necessarily dedicated to Gaussian sampling; this goes beyond the scope of the present paper.

[4] If g_2 admits a nonquadratic form, an additional splitting variable can be introduced and the following comments still hold.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

6. A Comparison of Gaussian Sampling Methods with Numerical Simulations. This section aims to provide readers with a detailed comparison of the Gaussian sampling techniques discussed in sections 3 and 4. In particular, we summarize the benefits and bottlenecks associated with each method, illustrate some of them on numerical applications, and propose an up-to-date selection of the most efficient algorithms for archetypal Gaussian sampling tasks. General guidelines resulting from this review and numerical experiments are gathered in Figure 12.

6.1. Summary, Comparison, and Discussion of Existing Approaches. Table 7 summarizes the main features of the sampling techniques reviewed above. In particular, for each approach, this table recalls its exactness (or not), its computational and storage costs, the most expensive sampling step to compute, the possible linear system to solve, and the presence of tuning parameters. It aims to synthesize the main pros and cons of each class of samplers. Regarding MCMC approaches, the computational cost associated to the required sampling step is not taken into account in the column "Comp. cost" since it depends upon the structure of $\mathbf{Q}$. Instead, the column "Sampling" indicates the type of sampling step required by the sampling approach.

Rather than conducting a one-to-one comparison between samplers, we choose to focus on selected important questions raised by the taxonomy reported in this table. Concerning more technical or in-depth comparisons between specific approaches, we refer the interested reader to the appropriate references; see, for instance, those highlighted in sections 3 and 4. These questions of interest lead to scenarios that motivate the dedicated numerical experiments conducted in subsection 6.2, then subsection 6.3 gathers guidelines to choosing an appropriate sampler for a given sampling task. The questions we focus on are as follows.

Question 1. In which scenarios do iterative approaches become interesting compared to factorization approaches?

In section 3, one sees that square root, CG, and PO approaches bypass the computational and storage costs of factorization thanks to an iterative process of K cheaper steps, with $K \in \mathbb{N}_*$. A natural question is, in which scenarios does the total cost of K iterations remain efficient when compared to factorization methods? Table 7 tells us that high-dimensional scenarios ($d \gg 1$) favor iterative methods as soon as memory requirements of order $\Theta(d^2)$ become prohibitive. If this storage is not an issue, iterative samplers become interesting only if the number of iterations K is such that $K \ll (d + T - 1)/T$. This inequality is verified only in cases where a small number of samples T is required ($T \ll d$), which in turn imposes $K \ll d$. Note that this condition remains similar when a Gaussian sampling step is embedded within a Gibbs sampler with a varying covariance or precision matrix (see Example 2.1): the condition on K is $K \ll d$, whatever the number T of samples, since it is no longer possible to precompute the factorization of $\mathbf{Q}$.

Question 2. In which scenarios should we prefer an iterative sampler from section 3 or an MCMC sampler from section 4?

Table 7 shows that the iterative samplers reviewed in section 3 have to perform K iterations to generate one sample. In contrast, most MCMC samplers generate one sample per iteration. However, these samples are distributed according to the target distribution (or an approximation of it) only in an asymptotic regime, i.e., when $T \to \infty$ (in practice, after a burn-in period). If one considers a burn-in period of length T_{bi} whose samples are discarded, MCMC samplers are interesting w.r.t.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Table 7 *Taxonomy of existing methods to sample from an arbitrary d-dimensional Gaussian distribution $\Pi \triangleq \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$; see (2.1). $K \in \mathbb{N}_*$ and $T_{\mathrm{bi}} \in \mathbb{N}_*$ are the number of iterations of a given iterative sampler (e.g., the CG one) and the number of burn-in iterations for MCMC algorithms, respectively. "Target Π" refers to approaches that target the right distribution Π under infinite precision arithmetic. Note that some i.i.d. samplers using a truncation parameter K might target Π for specific choices of K (e.g., the classical CG sampler is exact for $K = d$). "Finite time sampling" refers here to approaches which need a truncation procedure to deliver a sample in finite time. The matrix $\mathbf{A}$ is a symmetric and positive-definite matrix associated to $\mathbf{Q}$; see section 4. The sampling methods highlighted in bold stand for novel approaches derived from the proposed unifying framework. AMS stands for "approximate matrix splitting" (see Table 6), EDA for "exact data augmentation" (see Table 4), and ADA for "approximate data augmentation" (see Table 6).*

Method	Instance	Target Π	Finite time sampling	Comp. cost	Storage cost	Sampling	Linear system	No tuning
Factorization	Cholesky [91, 94]	✓	✓	$\mathcal{O}(d^2(d+T))$	$\Theta(d^2)$	$\mathcal{N}(0,1)$	triangular	✓
	Square root [24]	✓	✓	$\mathcal{O}(d^2(d+T))$	$\Theta(d^2)$	$\mathcal{N}(0,1)$	full	✓
$\mathbf{Q}^{1/2}$ approx.	Chebyshev [24, 82]	✗	✓	$\mathcal{O}(d^2KT)$	$\Theta(d)$	$\mathcal{N}(0,1)$	✗	✓
	Lanczos [7, 98]	✗	✓	$\mathcal{O}(K(K+d^2)T)$	$\Theta(K(K+d))$	$\mathcal{N}(0,1)$	✗	✓
CG	Classical [81]	✗	✓	$\mathcal{O}(d^2KT)$	$\Theta(d)$	$\mathcal{N}(0,1)$	✗	✓
	Gradient scan [30]	✓	✗	$\mathcal{O}(d^2K(T+T_{\mathrm{bi}}))$	$\Theta(Kd)$	$\mathcal{N}(0,1)$	✗	✓
PO	Truncated [77, 79]	✗	✓	$\mathcal{O}(d^2KT)$	$\Theta(d)$	$\mathcal{N}(0,1)$	full	✓
	Reversible jump [37]	✓	✗	$\mathcal{O}(d^2K(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	full	✓
MS	Richardson [32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(\mathbf{0}_d, \mathbf{A})$	diagonal	✗
	Jacobi [32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(\mathbf{0}_d, \mathbf{A})$	diagonal	✓
	Gauss–Seidel [32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	triangular	✓
	SOR [32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	triangular	✗
	SSOR [32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	triangular	✗
	Cheby-SSOR [31, 32]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	triangular	✗
	Hogwild [54]	✗	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	diagonal	✓
	Clone MCMC [9]	✗	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	diagonal	✗
	Unifying AMS	✗	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	diagonal	✓
DA	EDA [66]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(\mathbf{0}_d, \mathbf{A})$	✗	✓
	GEDA [66]	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	✗	✓
	SGS [104]	✗	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(\mathbf{0}_d, \mathbf{A})$	✗	✗
	Unifying EDA	✓	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(\mathbf{0}_d, \mathbf{A})$	✗	✓
	Unifying ADA	✗	✗	$\mathcal{O}(d^2(T+T_{\mathrm{bi}}))$	$\Theta(d)$	$\mathcal{N}(0,1)$	✗	✗

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

iterative ones only if $T + T_{\text{bi}} \ll KT$. Since most often $K \ll T_{\text{bi}}$, this condition favors MCMC methods when a large number $T \gtrsim T_{\text{bi}}$ of samples is desired. When a small number $T \lesssim T_{\text{bi}}/K$ of samples is desired, iterative methods are preferred. In intermediate situations, the choice depends on the precise number of required samples T, mixing properties of the MCMC sampler, and the number of iterations K of the alternative iterative algorithm.

Question 3. When is it efficient to use a decomposition $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ of the precision matrix in comparison with other approaches?

Sections 3 and 4 have shown that some sampling methods, such as Algorithms 3.4 and 4.5, exploit a decomposition of the form $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ to simplify the sampling task. Regarding the PO approaches, the main benefit lies in the cheap computation of the vector $\mathbf{z}' \sim \mathcal{N}(\mathbf{0}_d, \mathbf{Q})$ before solving the linear system $\mathbf{Q}\boldsymbol{\theta} = \mathbf{z}'$; see [79] for more details. On the other hand, MCMC samplers based on exact DA yield simpler sampling steps a priori and do not require us to solve any high-dimensional linear system. However, the Achilles' heel of MCMC methods is that they only produce samples of interest in the asymptotic regime where the number of iterations tends toward infinity. For a fixed number of MCMC iterations, dependent samples are obtained and their quality depends highly upon the mixing properties of the MCMC sampler. Numerical experiments in subsection 6.2 include further discussion on this point.

6.2. Numerical Illustrations. This section illustrates the main differences between the methods reviewed in sections 3 and 4. The main purpose is not to give an exhaustive one-to-one comparison among all approaches listed in Table 7. Instead, these methods are compared in light of three experimental scenarios that address the questions raised in subsection 6.1. More specific numerical simulations can be found in the cited works and references therein. Since the main challenges of Gaussian sampling are related to the properties of the precision matrix $\mathbf{Q}$, or the covariance $\boldsymbol{\Sigma}$ (see subsection 2.2), the mean vector $\boldsymbol{\mu}$ is set to $\mathbf{0}_d$ and only centered distributions are considered. For the first two scenarios associated with Questions 1 and 2, the unbiased estimate of the empirical covariance matrix will be used to assess the convergence in distribution of the samples generated by each algorithm:

$$\widehat{\boldsymbol{\Sigma}}_T = \frac{1}{T-1} \sum_{t=1}^{T} (\boldsymbol{\theta}^{(t)} - \bar{\boldsymbol{\theta}})(\boldsymbol{\theta}^{(t)} - \bar{\boldsymbol{\theta}})^\top , \tag{6.1}$$

where $\bar{\boldsymbol{\theta}} = T^{-1} \sum_{t=1}^{T} \boldsymbol{\theta}^{(t)}$ is the empirical mean. Note that other metrics (such as the empirical precision matrix) could have been used to assess whether these samples are distributed according to a sufficiently close approximation of the target distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. Among available metrics, we chose the one that has been the most used in the literature, that is, (6.1) [9, 32, 37].

For the scenario 3 associated with the corresponding final question, the considered high-dimensional setting will preclude the computation of exact and empirical covariance matrices. Instead, MCMC samplers will be compared in terms of the computational efficiency and quality of the generated chains (see subsection 6.2.3 for details).

The experimental setting is as follows. To ensure fair comparisons, all algorithms are implemented on equal grounds, with the same quality of optimization. The programming language used is Python and all loops have been carefully re-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

placed by matrix-vector products as far as possible. Simulations are run on a computer equipped with an Intel Xeon 3.70 GHz processor with 16.0 GB of RAM, under Windows 7. Among the infinite set of possible examples, we choose examples of Gaussian sampling problems that often appear in applications and that have been previously considered in the literature, so that they provide good tutorials to answer the question raised by each scenario. The code to reproduce all the figures of this section is available in a Jupyter notebook format available online at https://github.com/mvono/PyGauss/tree/master/notebooks.

6.2.1. Scenario 1. This first set of experiments addresses Question 1 concerning iterative versus factorization approaches. We consider a sampling problem also tackled in [81] to demonstrate the performance of Algorithm 3.5 based on the conjugate gradient. For the sake of clarity, we divide this scenario into two subexperiments.

Comparison between Factorization and Iterative Approaches. In this first subexperiment, we compare so-called factorization approaches with iterative approaches on two Gaussian sampling problems. We consider first the problem of sampling from $\mathcal{N}(\mathbf{0}_d, \mathbf{Q}^{-1})$, where the covariance matrix $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$ is chosen as a squared exponential matrix that is commonly used in applications involving Gaussian processes [48, 63, 85, 96, 103, 108]. Its coefficients are defined by

$$\Sigma_{ij} = 2\exp\left(-\frac{(s_i - s_j)^2}{2a^2}\right) + \epsilon\delta_{ij} \quad \forall i, j \in [d]\ , \tag{6.2}$$

where $\{s_i\}_{i\in[d]}$ are evenly spaced scalars on $[-3, 3]$, $\epsilon > 0$, and the Kronecker symbol $\delta_{ij} = 1$ if $i = j$ and zero otherwise. In (6.2), the parameters a and ϵ have been set to $a = 1.5$ and $\epsilon = 10^{-6}$, which yields a distribution of the eigenvalues of $\boldsymbol{\Sigma}$ such that the large ones are well separated while the small ones are clustered together near 10^{-6}; see Figure 5 (first row). We compare the Cholesky sampler (Algorithm 3.1), the approximate inverse square root sampler using Chebyshev polynomials (Algorithm 3.2), and the CG-based sampler (Algorithm 3.5). The sampler using Chebyshev polynomials needs $K_{\text{cheby}} = 23$ iterations on average, while the CG iterations are stopped once loss of conjugacy occurs, following the guidelines of [81], that is, at $K_{\text{kryl}} = 8$. In all experiments, $T = 10^5$ samples are simulated in dimensions ranging from 1 to several thousands.

Figure 5 shows the results associated with these three direct samplers in dimension $d = 100$. The generated samples indeed follow a target Gaussian distribution admitting a covariance matrix close to $\boldsymbol{\Sigma}$. This is attested to by the small residuals observed between the estimated and the true covariances. Based on this criterion, all approaches successfully sample from $\mathcal{N}(\mathbf{0}_d, \boldsymbol{\Sigma})$. This is emphasized by the spectrum of the estimated matrices $\widehat{\boldsymbol{\Sigma}}_T$, which coincides with the spectrum of $\boldsymbol{\Sigma}$ for large eigenvalues. This observation ensures that most of the covariance information has been captured. However, note that only the Cholesky method is able to recover accurately all the eigenvalues, including the smallest ones.

Figure 6 compares the direct samplers above in terms of central processing unit (CPU) time. To generate only one sample ($T = 1$), as expected, one can observe that the Cholesky sampler is faster than the two iterative samplers in small dimension d and becomes computationally demanding as d grows beyond a few hundred. Indeed, for small d the cost implied by several iterations (K_{cheby} or K_{kryl}) within each iterative sampler dominates the cost of the factorization in Algorithm 3.1, while the contrary holds for large values of d. Since the Cholesky factorization is performed only once,

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

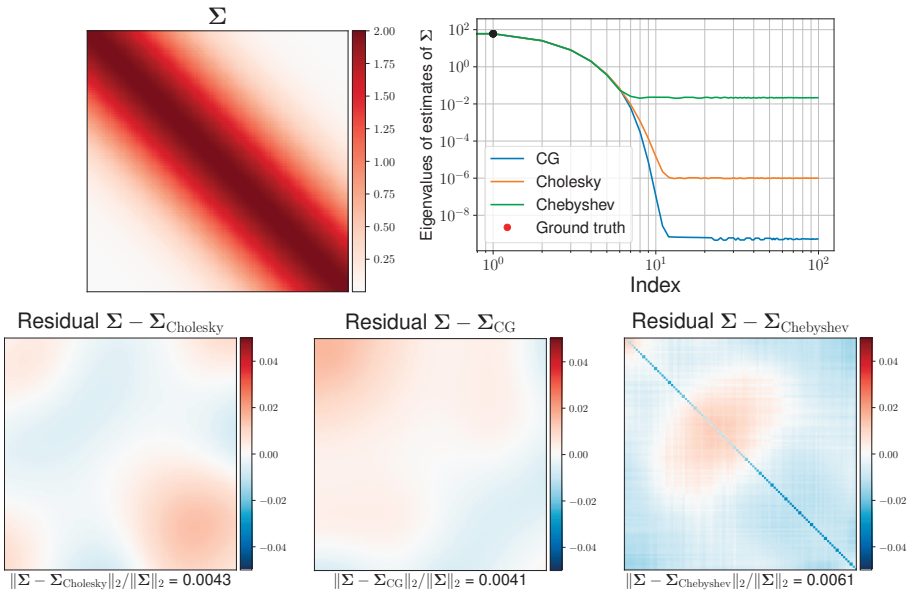


Fig. 5 *Scenario* 1. *Results of the three considered samplers for the sampling from* $\mathcal{N}(\mathbf{0}_d, \boldsymbol{\Sigma})$ *in dimension* $d = 100$ *with* $\boldsymbol{\Sigma}$ *detailed in* (6.2).

the Cholesky sampler becomes more attractive than the other two approaches as the sample size T increases. However, as already pointed out in subsection 2.3, it is worth noting that precomputing the Cholesky factor is no longer possible once the Gaussian sampling task involves a matrix $\boldsymbol{\Sigma}$ which changes at each iteration of a Gibbs sampler, e.g., when considering a hierarchical Bayesian model with unknown hyperparameters (see Example 2.1). We also point out that a comparison among direct samplers reviewed in section 3 and their related versions was conducted in [7] in terms of CPU and GPU times. In agreement with the findings reported here, this comparison essentially showed that the choice of the sampler in small dimensions is not particularly important, while iterative direct samplers become interesting in high-dimensional scenarios where Cholesky factorization becomes impossible.

We complement our analysis by focusing on another sampling problem which now considers the matrix defined in (6.2) as a precision matrix instead of a covariance matrix: we now want to generate samples from $\mathcal{N}(\mathbf{0}_d, \widetilde{\boldsymbol{\Sigma}})$ with $\widetilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^{-1}$. This sampling problem is expected to be more difficult since the largest eigenvalues of $\widetilde{\boldsymbol{\Sigma}}$ are now clustered; see Figure 7 (first row). Figure 7 (second row) shows that the CG and Chebyshev samplers fail to capture covariance information as accurately as the Cholesky sampler. The residuals between the estimated and the true covariances remain important on the diagonal: variances are inaccurately underestimated. This observation is in line with [81], which showed that the CG sampler is not suitable for the sampling from a Gaussian distribution whose covariance matrix has many clustered large eigenvalues, probably as a consequence of the bad conditioning of the matrix. As far as the Chebyshev sampler is concerned, this failure was expected since the interval $[\lambda_l, \lambda_u]$ on which the function $x \mapsto x^{-1/2}$ has to be well approximated becomes very large with an extent of about 10^6. Of course, the relative error between

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

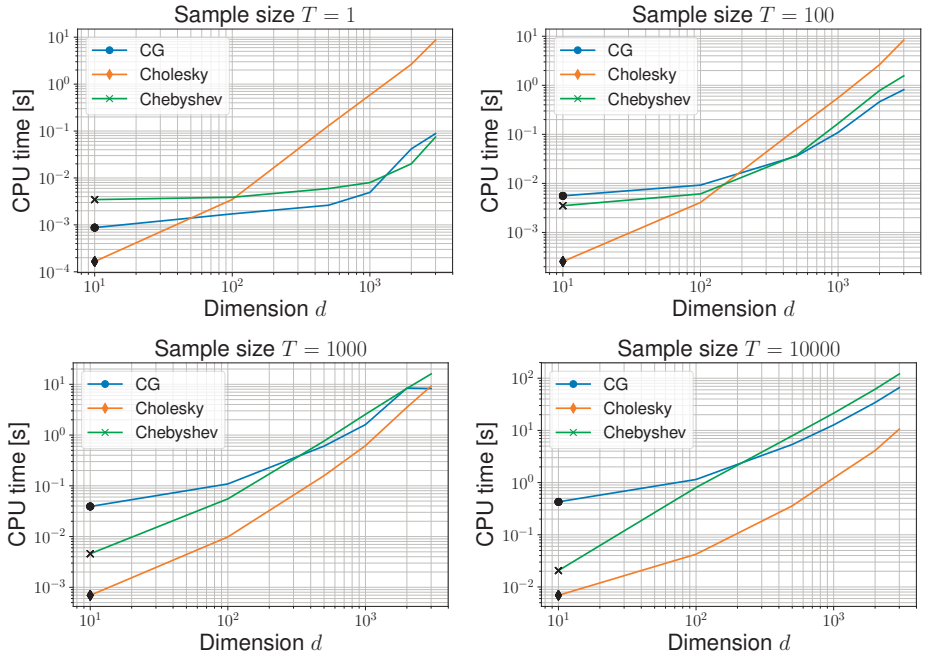


Fig. 6 *Scenario* 1. *Comparison among the three considered direct samplers in terms of CPU time for the sampling from* $\mathcal{N}(\mathbf{0}_d, \boldsymbol{\Sigma})$ *with* $\boldsymbol{\Sigma}$ *detailed in* (6.2).

$\widetilde{\boldsymbol{\Sigma}}$ and its estimate can be decreased by sufficiently increasing the number of iterations K_{cheby}, but this is possible only at a prohibitive computational cost.

On the choice of the metric to monitor convergence. We saw in Figures 5 and 7 that the covariance estimation error was small if the large values of the covariance matrix were captured and large in the opposite scenario. Note that if the precision estimation error was chosen as a metric, we would have observed similar numerical results: if the largest eigenvalues of the precision matrix were not captured, the precision estimation error would have been large. Regarding the CG sampler, Fox and Parker in [31], for instance, highlighted this fact and illustrated it numerically (see equations (3.1) and (3.2) in that paper).

Comparison between Chebyshev- and CG-Based Samplers. In order to discriminate the two iterative direct samplers in Algorithms 3.2 and 3.5, we consider a toy example in dimension $d = 15$. The covariance matrix $\boldsymbol{\Sigma}$ is chosen to be diagonal with diagonal elements drawn randomly from the discrete set $[\![1, 5]\!]$. As shown in Figure 8, $\boldsymbol{\Sigma}$ has repeated and large eigenvalues. Because of this, the CG sampler stops at $K_{\text{kryl}} = 5$ (the number of distinct eigenvalues of $\boldsymbol{\Sigma}$) and fails to sample accurately from $\mathcal{N}(\mathbf{0}_d, \boldsymbol{\Sigma})$, while the sampler based on Chebyshev polynomials yields samples of interest. Hence, although the CG sampler is an attractive iterative option, its accuracy is known to be highly dependent on the distribution of the eigenvalues of $\boldsymbol{\Sigma}$ which is in general unknown in high-dimensional settings. Preconditioning techniques to spread out these eigenvalues can be used but might fail to reduce the error significantly, as shown in [81].

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

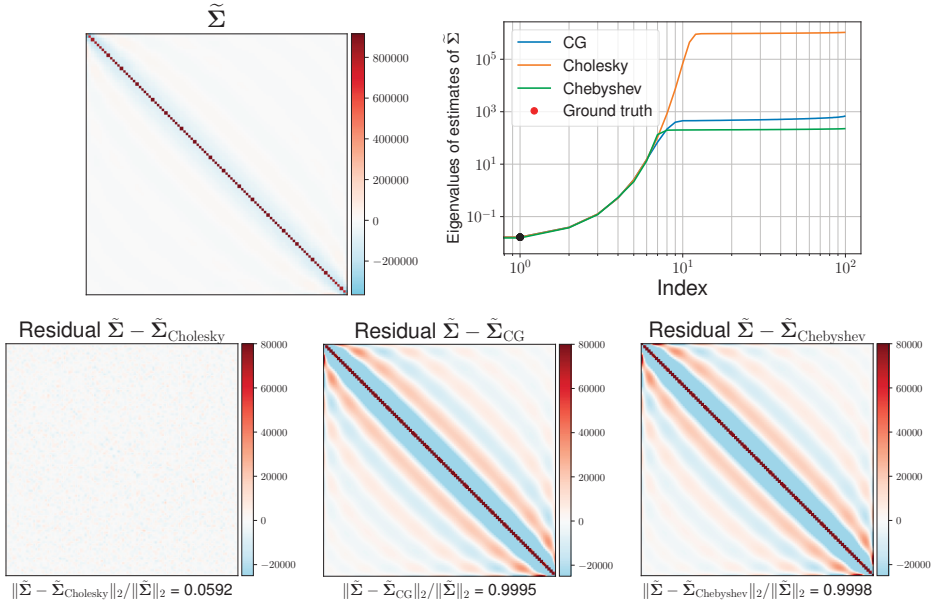


Fig. 7 *Scenario* 1. *Results of the three considered samplers for the sampling from* $\mathcal{N}(\mathbf{0}_d, \widetilde{\boldsymbol{\Sigma}})$ *in dimension* $d = 100$.

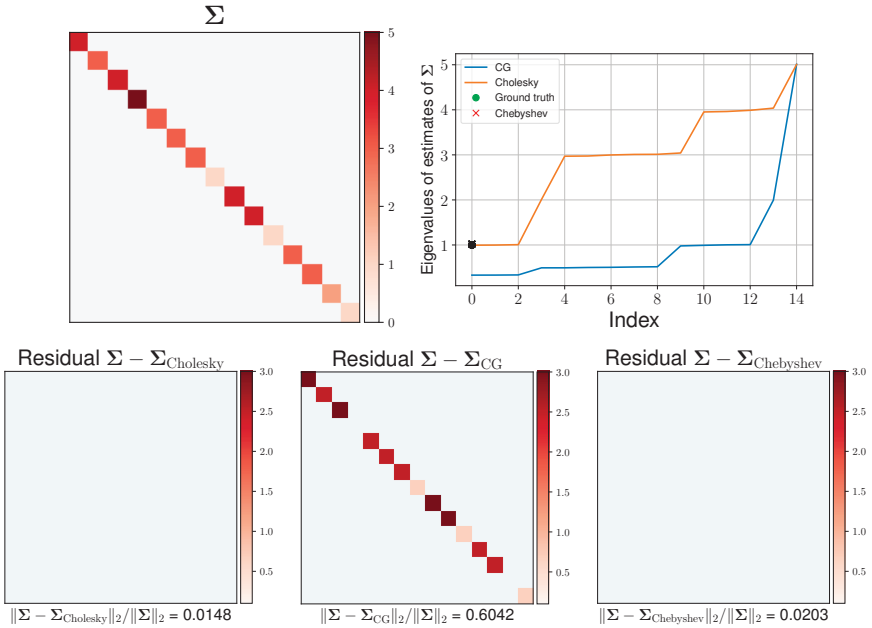


Fig. 8 *Scenario* 1. *Results of the three considered direct samplers for the sampling from* $\mathcal{N}(\mathbf{0}_d, \boldsymbol{\Sigma})$ *with* $\boldsymbol{\Sigma}$ *diagonal in dimension* $d = 15$.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

6.2.2. Scenario 2. Now we turn to Question 2 and compare iterative and MCMC samplers. In this scenario, we consider a precision matrix $\mathbf{Q}$ which is commonly used to build Gaussian Markov random fields (GMRFs) [92]. Before defining this matrix, we introduce some notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected two-dimensional graph (see Figure 9), where $\mathcal{V}$ is the set of d nodes in the graph and $\mathcal{E}$ represents the edges between the nodes. We say that nodes i and j are neighbors and write $i \sim j$ if there is an edge connecting these two nodes. The number of neighbors of node i is denoted n_i and is also called the *degree*. Using this notation, we set $\mathbf{Q}$ to be a second-order locally linear precision matrix [48, 92] associated to the two-dimensional lattice shown in Figure 9,

$$Q_{ij} = \epsilon\delta_{ij} + \begin{cases} \phi n_i & \text{if } i = j \ , \\ -\phi & \text{if } i \sim j \ , \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in [d] \ , \tag{6.3}$$

where we set $\epsilon = 1$ (in fact, $\epsilon > 0$ suffices) and $\phi > 0$ to ensure that $\mathbf{Q}$ is a nonsingular matrix yielding a nonintrinsic Gaussian density w.r.t. the d-dimensional Lebesgue measure; see subsection 2.2. Note that this precision matrix is band-limited with bandwidth of the order $\mathcal{O}(\sqrt{d})$ [92], preluding the possible embedding of Algorithm 2.3 within the samplers considered in this scenario. Related instances of this precision matrix have also been considered in [32, 52, 81] in order to show the benefits of both direct and MCMC samplers. In what follows, we consider the sampling from $\mathcal{N}(\mathbf{0}_d, \mathbf{Q}^{-1})$ for three different scalar parameters $\phi \in \{0.1, 1, 10\}$ leading to three covariance matrices $\mathbf{Q}^{-1}$ with different correlation structures; see Figure 9. This will be of interest since it is known that the efficiency of Gibbs samplers is highly dependent on the correlation between the components of the Gaussian vector $\boldsymbol{\theta} \in \mathbb{R}^d$ [87, 92].

For this scenario, we set $d = 100$ in order to provide complete diagnostics for evaluating the accuracy of the samples generated by each algorithm. We implement the four MCMC samplers based on exact MS (see Table 2) without considering a burn-in period (i.e., $T_{\text{bi}} = 0$). These MCMC algorithms are compared with the direct samplers based on Cholesky factorization and Chebyshev polynomials; see section 3. Since the matrix $\mathbf{Q}$ is strictly diagonally dominant, that is, $|Q_{ii}| > \sum_{j \neq i} |Q_{ij}|$ for all $i \in [d]$, the convergence of the MCMC sampler based on Jacobi splitting is ensured [32, 41]. Based on this convergence property, we can use an optimal value for the parameter ω appearing in the MCMC sampler based on successive overrelaxation (SOR) splitting; see Appendix C.

Figure 10 shows the relative error between the estimated covariance matrix and the true one w.r.t. the number of samples generated with i.i.d. samplers from section 3 and MCMC samplers from section 4. Regarding MCMC samplers, no burn-in has been considered here in order to emphasize that these algorithms do not yield i.i.d. samples from the first iteration compared to the samplers reviewed in section 3. This behavior is particularly noticeable when $\phi = 10$, where one can observe that Gauss–Seidel, Jacobi, and Richardson samplers need far more samples than Chebyshev and Cholesky samplers to reach the same precision in terms of covariance estimation. Interestingly, this claim does not hold for "accelerated" Gibbs samplers such as the SOR (accelerated version of the Gauss–Seidel sampler) and the Chebyshev accelerated SSOR samplers. Indeed, for $\phi = 1$, one can note that the latter sampler is as efficient as i.i.d. samplers. On the other hand, when $\phi = 10$, these two accelerated Gibbs samplers manage to achieve lower relative covariance estimation error than the Chebyshev sampler when the number of iterations increases. This behavior is due to the fact that the Chebyshev

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

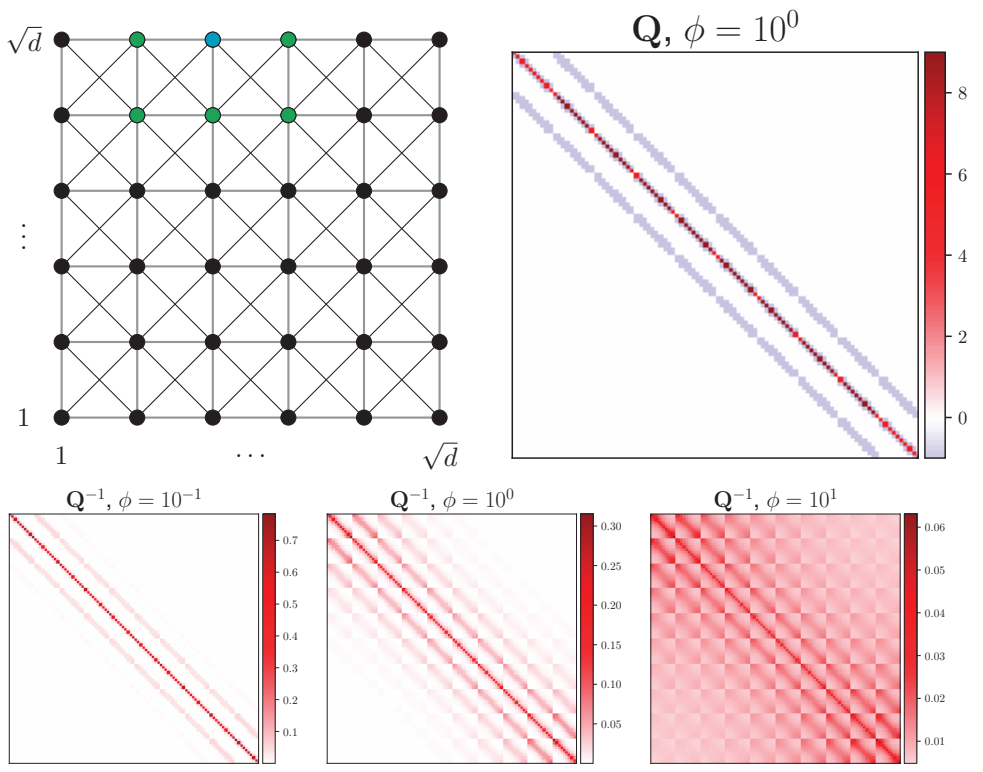


Fig. 9 *Scenario* 2. *Illustrations of the considered Gaussian sampling problem: (top left)* 2*D lattice* ($d = 36$) *associated to the precision matrix* $\mathbf{Q}$ *in* (6.3)*; (top right)* $\mathbf{Q}$ *depicted for* $\phi = 1$*; (bottom)* $\mathbf{Q}^{-1} = \boldsymbol{\Sigma}$ *for* $\phi \in \{0.1, 1, 10\}$. *All the results are shown for* $d = 100$. *On the* 2*D lattice, the green nodes are the neighbors of the blue node while the coordinates of the lattice correspond to the coordinates of* $\boldsymbol{\theta} \in \mathbb{R}^d$ *with nonzero correlations, that is,* $1 \leq i, j \leq \sqrt{d}$.

sampler involves a truncation procedure and as such provides approximate samples from $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ compared to exact MCMC schemes, which produce asymptotically exact samples. These numerical findings match observations made by [32], which experimentally showed that accelerated Gibbs approaches can be considered as serious contenders for the fastest samplers in high-dimensional settings compared to i.i.d. methods.

Table 8 complements these numerical findings by reporting the spectral radius of the iteration operator $\mathbf{M}^{-1}\mathbf{N}$ associated to each MCMC sampler. This radius is particularly important since it is directly related to the convergence factor of the corresponding MCMC sampler [32]. In order to provide quantitative insights into the relative performance of each sampler, Table 8 also shows the number of samples T and corresponding CPU time such that the relative error between the covariance matrix $\boldsymbol{\Sigma}$ and its estimate $\widehat{\boldsymbol{\Sigma}}_T$ computed from T samples generated by each algorithm is lower than 5×10^{-2}, i.e., a relative error of 5%. Thanks to the small bandwidth ($b = 11$) of $\mathbf{Q}$, the covariance matrix $\mathbf{M}^\top + \mathbf{N}$ of the vector $\tilde{\mathbf{z}}$ appearing in step 3 of Algorithm 4.2 is also band-limited with $b = 11$ for both Jacobi and Richardson splitting. Hence Algorithm 2.3 specifically dedicated to the band matrix can be used within Algorithm 4.2 for these two splitting strategies. Although the convergence

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

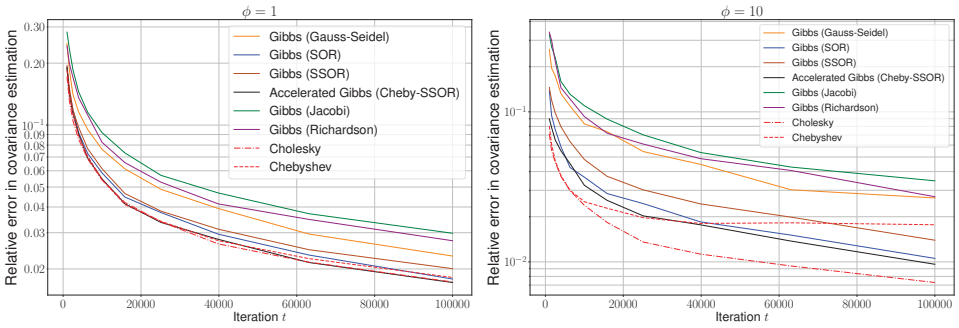


Fig. 10 *Scenario* 2. *Relative error associated to the estimation of the covariance matrix* $\mathbf{Q}^{-1}$ *defined by* $\|\mathbf{Q}^{-1} - \mathrm{var}(\boldsymbol{\theta}^{(1:t)})\|_2 / \|\mathbf{Q}^{-1}\|_2$ *w.r.t. the number of iterations* t *in Algorithm* 4.2, *with* $d = 100$ *(left:* $\phi = 1$, *right:* $\phi = 10$*). We also highlight the relative error obtained with an increasing number of samples generated independently from Cholesky and Chebyshev samplers. The results have been averaged over* 30 *independent runs. The standard deviations are not shown for readability reasons.*

of these samplers is slower than that of the Gauss–Seidel sampler, their CPU times become roughly equivalent. Note that this computational gain is problem-dependent and cannot be ensured in general. Cholesky factorization appears to be much faster in all cases when the same constant covariance is used for many samples. The next scenario will consider high-dimensional scenarios where Cholesky factorization is no longer possible.

6.2.3. Scenario 3. Finally, we deal with Question 3 to assess the benefits of samplers that take advantage of the decomposition structure $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ of the precision matrix. As motivated in subsection 6.1, we will focus here on the exact DA approaches detailed in subsection 4.2 and compare the latter to iterative samplers which produce uncorrelated samples, such as those reviewed in section 3.

To this end, we consider Gaussian sampling problems in high dimensions $d \in [10^4, 10^6]$ for which Cholesky factorization is both computationally and memory prohibitive when a standard computer is used. This sampling problem commonly appears in image processing [38, 67, 78, 104] and arises from the linear inverse problem, usually called *deconvolution* or *deblurring* in image processing:

$$\mathbf{y} = \mathbf{S}\boldsymbol{\theta} + \boldsymbol{\varepsilon}\,, \tag{6.4}$$

where $\mathbf{y} \in \mathbb{R}^d$ refers to a blurred and noisy observation, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the unknown original image rearranged in lexicographic order, and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{\Gamma})$ with $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_d)$ is a synthetic structured noise such that $\gamma_i \sim 0.7\delta_{\kappa_1} + 0.3\delta_{\kappa_2}$ for all $i \in [d]$. In what follows, we set $\kappa_1 = 13$ and $\kappa_2 = 40$. The matrix $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a circulant convolution matrix associated to the space-invariant box blurring kernel $\frac{1}{9}\mathbf{1}_{3\times 3}$, where $\mathbf{1}_{p\times p}$ is the $p \times p$-matrix filled with ones. We adopt a smoothing conjugate prior distribution on $\boldsymbol{\theta}$ [61, 74, 75], introduced in subsection 2.2 and Figure 2, written as $\mathcal{N}(\mathbf{0}_d, (\frac{\xi_0}{d}\mathbf{1}_{d\times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta})^{-1})$, where $\boldsymbol{\Delta}$ is the discrete two-dimensional Laplacian operator; $\xi_0 = 1$ ensures that this prior is nonintrinsic while $\xi_1 = 1$ controls the smoothing. Bayes' rule then yields the Gaussian posterior distribution

$$\boldsymbol{\theta} \mid \mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{Q}^{-1}\right), \tag{6.5}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Table 8 *Scenario 2. Comparison between Cholesky-, Chebyshev-, and MS-based Gibbs samplers for $d = 100$. The samplers have been run until the relative error between the covariance matrix $\mathbf{Q}^{-1}$ and its estimate is lower than 5×10^{-2}. For Richardson, SOR, SSOR, and Cheby-SSOR samplers, the tuning parameter ω is the optimal one; see Appendix C. The results have been averaged over 30 independent runs.*

Sampler		ϕ	ω	$\rho(\mathbf{M}^{-1}\mathbf{N})$	T	CPU time [s]
Cholesky		0.1	-	-	6.3×10^4	0.29
		1	-	-	1.3×10^4	0.06
		10	-	-	2.9×10^3	0.01
Chebyshev ($K = 21$)		0.1	-	-	6.4×10^4	2.24
		1	-	-	1.3×10^4	0.44
		10	-	-	2.5×10^3	0.19
MCMC MS-based samplers	Richardson	0.1	0.6328	0.3672	6.7×10^4	5.44
		1	0.1470	0.8530	3.8×10^4	3.03
		10	0.0169	0.9831	4×10^4	3.31
	Jacobi	0.1	-	0.4235	6.8×10^4	5.72
		1	-	0.8749	3.9×10^4	3.24
		10	-	0.9856	4.6×10^4	3.69
	Gauss–Seidel	0.1	-	0.1998	6.5×10^4	8.48
		1	-	0.7677	2.5×10^4	3.34
		10	-	0.9715	2.5×10^4	3.32
	SOR	0.1	1.0494	0.1189	6.4×10^4	8.40
		1	1.3474	0.4726	1.6×10^4	1.31
		10	1.7110	0.7852	5.4×10^3	0.71
	SSOR	0.1	0.9644	0.0936	6.4×10^4	19.65
		1	1.3331	0.4503	1.6×10^4	4.91
		10	1.7101	0.9013	9.3×10^3	2.86
	Cheby-SSOR	0.1	0.9644	0.0246	6.3×10^4	9.17
		1	1.3331	0.1485	1.3×10^4	1.89
		10	1.7101	0.5213	4.5×10^3	0.65

where

$$\mathbf{Q} = \mathbf{S}^\top \boldsymbol{\Delta}^{-1}\mathbf{S} + \frac{\xi_0}{d}\mathbf{1}_{d\times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta} \,, \tag{6.6}$$

$$\boldsymbol{\mu} = \mathbf{Q}^{-1}\mathbf{S}^\top \boldsymbol{\Delta}^{-1}\mathbf{y} \,. \tag{6.7}$$

Sampling from (6.5) is challenging since the size of the precision matrix forbids its computation. Moreover, the presence of the matrix $\boldsymbol{\Gamma}$ rules out the diagonalization of $\mathbf{Q}$ in the Fourier basis and therefore the direct use of Algorithm 2.4. In addition, resorting to MCMC samplers based on exact MS to sample from (6.5) raises several difficulties. First, both Richardson- and Jacobi-based samplers involve a sampling step with an unstructured covariance matrix; see Table 2. This step can be performed with one of the direct samplers reviewed in section 3 but this implies an additional computational cost. On the other hand, although Gauss–Seidel and SOR-based MCMC samplers involve a simple sampling step, they require access to the lower triangular part of (6.6). In this high-dimensional scenario, the precision matrix cannot be easily computed on a standard desktop computer and this lower triangular part must be found with surrogate approaches. One possibility is to compute each nonzero coef-

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

ficient of this triangular matrix following the matrix-vector products $\mathbf{e}_i^\top \mathbf{Q}\mathbf{e}_j$ for all $i, j \in [d]$ such that $j \leq i$, where we recall that $\mathbf{e}_i$ is the ith canonical vector of $\mathbb{R}^d$. These quantities can be precomputed when $\mathbf{Q}$ remains constant along the T iterations but, again, this becomes computationally prohibitive when $\mathbf{Q}$ depends on unknown hyperparameters to be estimated within a Gibbs sampler.

Nonetheless, since the precision matrix (6.6) can be decomposed as $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ with $\mathbf{Q}_1 = \mathbf{S}^\top \boldsymbol{\Gamma}^{-1}\mathbf{S}$ and $\mathbf{Q}_2 = \frac{\xi_0}{d}\mathbf{1}_{d\times d} + \xi_1 \boldsymbol{\Delta}^\top \boldsymbol{\Delta}$, we can apply Algorithm 4.5 to sample efficiently from (6.5). This algorithm is particularly interesting in this example since the three sampling steps involve two diagonal matrices and one circulant precision matrix, respectively. For the two first ones, one can use Algorithm 2.2, while Algorithm 2.4 can be used to sample from the last one.

In what follows, we compare Algorithm 4.5 with the CG direct sampler defined by Algorithm 3.5. Since we consider high-dimensional scenarios, the covariance estimate in (6.1) cannot be used to assess the convergence of these samplers. Instead, we compare the respective efficiency of the considered samplers by computing the effective sample size ratio per second (ESSR). For an MCMC sampler, the ESSR gives an estimate of the equivalent number of i.i.d. samples that can be drawn in one second; see [55, 62]. It is defined as

$$\text{ESSR}(\vartheta) = \frac{1}{T_1}\frac{\text{ESS}(\vartheta)}{T} = \frac{1}{T_1\left(1 + 2\sum_{t=1}^{\infty}\rho_t(\vartheta)\right)}, \tag{6.8}$$

where T_1 is the CPU time in seconds required to draw one sample and $\rho_t(\vartheta)$ is the lag-t autocorrelation of a scalar parameter ϑ. A variant of the ESSR has, for instance, been used in [37] in order to measure the efficiency of an MCMC variant of the PO algorithm (Algorithm 3.4). For a direct sampler providing i.i.d. draws, the ESSR (6.8) simplifies to $1/T_1$ and represents the number of samples obtained in one second. In both cases, the larger the ESSR, the more computationally efficient is the sampler.

Figure 11 shows the ESSR associated to the two considered algorithms for $d \in [10^4, 10^6]$. The latter has been computed by choosing the "slowest" component of $\boldsymbol{\theta}$ as the scalar summary ϑ, that is, the one with the largest variance. As in the statistical software STAN [18], we have truncated the infinite sum in (6.8) at the first negative ρ_t. One can note that for the various high-dimensional problems considered here, the GEDA sampler exhibits good mixing properties which, combined with its low computational cost per iteration, yields a larger ESSR than the direct CG sampler. Hence, in this specific case, building on both the decomposition $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ of the precision matrix and an efficient MCMC sampler is highly beneficial compared to using $\mathbf{Q}$ directly in Algorithm 3.5. Obviously, this gain in computational efficiency w.r.t. direct samplers is not guaranteed in general since GEDA relies on an appropriate decomposition $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$.

6.3. Guidelines to Choosing the Appropriate Gaussian Sampling Approach. In this section, we provide the reader with some insights into how to choose the most appropriate sampler for a given Gaussian simulation task when vanilla Cholesky sampling cannot be envisioned. These guidelines are formulated as simple binary questions in a decision tree (see Figure 12) to determine the class of samplers that is of potential interest. We choose to start[5] from the existence of a natural decompo-

[5]Note that alternative decision trees could be built by considering other issues as the primary decision level.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

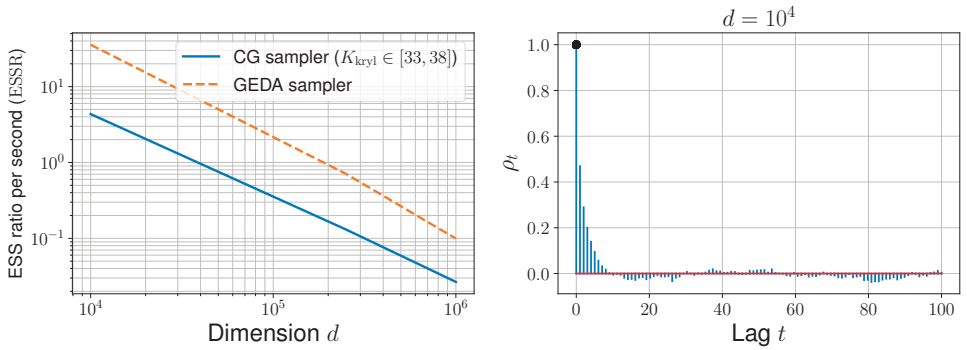


Fig. 11 *Scenario 3. (left) ESS ratio per second (ESSR); (right) autocorrelation function ρ_t shown for $d = 10^4$. For both figures, the slowest component of $\boldsymbol{\theta}$ is used as the scalar summary ϑ.*

sition $\mathbf{Q} = \mathbf{Q}_1 + \mathbf{Q}_2$ since some existing approaches are specifically dedicated to this scenario. Then we discriminate existing sampling approaches based on several criteria which have been discussed throughout this review such as the prescribed accuracy, the number of desired samples, or the eigenvalue spectrum of the precision matrix $\mathbf{Q}$. Regarding sampling accuracy, we highlight sampling approaches that are expected or not expected to yield samples with arbitrary accuracy more efficiently than vanilla Cholesky sampling; see Table 1. MCMC approaches introduced in section 4 and the iterative i.i.d. samplers of section 3 are distinguished depending on the number of desired samples T and their relative efficiency measured via the burn-in period T_{bi} for MCMC samplers and via a truncation parameter $K \in \mathbb{N}^*$ for i.i.d. samplers. The latter guidelines follow from remarks highlighted in sections 3 and 4 and from the numerical results in subsection 6.2.

As already mentioned in subsection 2.3, we emphasize that this review only aims to highlight the main approaches dedicated to high-dimensional Gaussian sampling, which arises in many different contexts. Therefore, it remains difficult to enunciate here precise rules for each context. Thus, the guidelines in Figure 12 give general principles to guide the practitioner toward an appropriate class of sampling approaches that are reviewed and complemented by additional references provided in this paper.

7. Conclusion. Given the ubiquity of the Gaussian distribution and the huge number of related contributions, this paper aimed to consolidate an up-to-date review of the main approaches dedicated to high-dimensional Gaussian sampling. To this end, we first presented the Gaussian sampling problem at stake as well as its specific and previously reviewed instances. Then we pointed out the main difficulties when the associated covariance matrix is not structured and the dimension of the problem increases. We reviewed two main classes of approaches from the literature, namely, approaches derived from numerical linear algebra and those based on MCMC sampling. In order to help practitioners in choosing the most appropriate algorithm for a given sampling task, we compared the reviewed methods by highlighting and illustrating their respective pros and cons. Ultimately, we have provided general insights into how to select one of the most appropriate samplers using a decision tree; see Figure 12. In addition, we have also unified most of the reviewed MCMC approaches under a common umbrella by building upon a stochastic counterpart of the celebrated proximal point algorithm that is well known in optimization. This enabled us to shed a new light on existing sampling approaches and draw further links between them. To

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

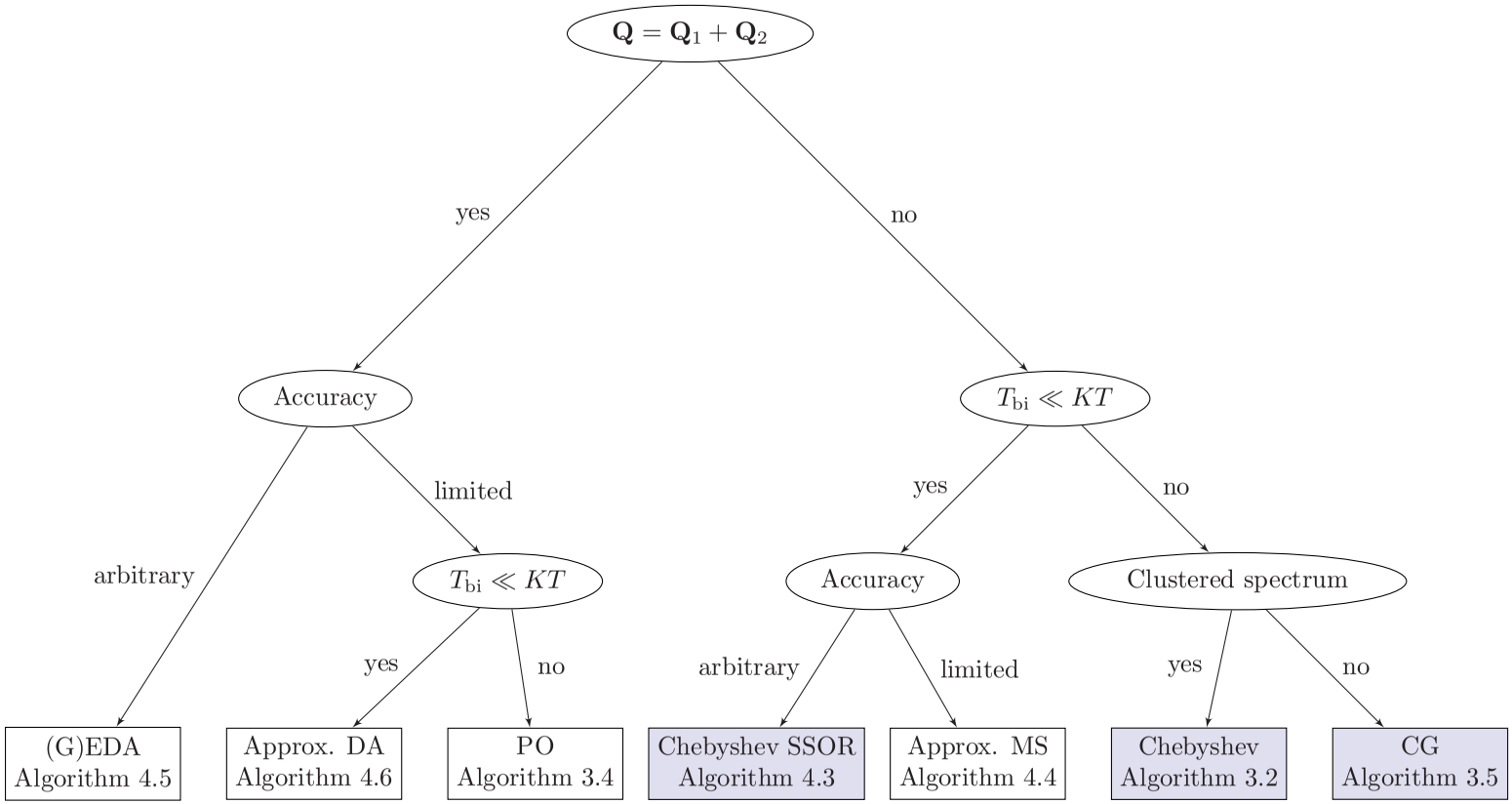


Fig. 12 *General guidelines to choosing the most appropriate sampling approach based on those reviewed in this paper. When computation of order* $\Theta(d^3)$ *and storage of* $\Theta(d^2)$ *elements are not prohibitive, the sampler of choice is obviously Algorithm* 3.1. *The accuracy of a sampler is referred to as* arbitrary *when it is expected to obtain samples with arbitrary accuracy more efficiently than vanilla Cholesky sampling; the accuracy is said to be* limited *when this is not the case. The integer parameters* T_{bi}, T, *and* K, *respectively, refer to the burn-in period of MCMC approaches, the number of desired samples from* $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, *and the truncation parameter associated to samplers of section* 3. *The shaded nodes highlight that nonasymptotic convergence bounds were available when this review was written.*

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

promote reproducibility, this article is complemented by a companion package written in Python named PyGauss[6] that implements all the reviewed approaches.

Appendix A. Guide to Notation. The following table lists and defines all the notation used in this paper.

$\mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian probability distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.
$\mathbf{M}^\top$	Transpose of matrix $\mathbf{M}$.
$f(d) = \mathcal{O}(d)$	Order of the function f when $d \to \infty$ up to constant factors.
$f(d) = \Theta(d)$	There exist $C_1, C_2 \in \mathbb{R}$ such that $C_1 d \leq f(d) \leq C_2 d$ when $d \to \infty$.
$\det(\mathbf{M})$	Determinant of the matrix $\mathbf{M}$.
$\triangleq$	By definition.
$\mathbf{0}_d$	Null vector on $\mathbb{R}^d$.
$\mathbf{0}_{d\times d}$	Null matrix on $\mathbb{R}^{d\times d}$.
$\mathbf{I}_d$	Identity matrix on $\mathbb{R}^{d\times d}$.
$\|\cdot\|$	The L^2 norm.
$\mathrm{diag}(\mathbf{v})$	The $d \times d$ diagonal matrix with diagonal elements $\mathbf{v} = (v_1, \ldots, v_d)^\top$.
$\mathbf{Q} = \mathbf{M} - \mathbf{N}$	Matrix-splitting decomposition of the precision matrix $\mathbf{Q}$.

Appendix B. Details and Proofs Associated to Subsection 5.1. First, we briefly recall some useful definitions associated to monotone operators. For more information on the theory of monotone operators in Hilbert spaces, we refer the interested reader to the book [11].

General Definitions.

DEFINITION B.1 (operator). *Let the notation $2^{\mathbb{R}^d}$ represent the family of all subsets of $\mathbb{R}^d$. An operator or multivalued function $\mathsf{K}\colon \mathbb{R}^d \to 2^{\mathbb{R}^d}$ maps every point in $\mathbb{R}^d$ to a subset of $\mathbb{R}^d$.*

DEFINITION B.2 (graph). *Let $\mathsf{K} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$. The graph of K is defined by*

$$\mathrm{gra}(\mathsf{K}) = \{(\boldsymbol{\theta}, \mathbf{u}) \in \mathbb{R}^d \times \mathbb{R}^d \mid \mathbf{u} \in \mathsf{K}(\boldsymbol{\theta})\}. \tag{B.1}$$

DEFINITION B.3 (monotone operator). *Let $\mathsf{K} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$. K is said to be monotone if*

$$\forall (\boldsymbol{\theta}, \mathbf{u}) \in \mathrm{gra}(\mathsf{K}) \quad \textit{and} \quad \forall (\mathbf{y}, \mathbf{p}) \in \mathrm{gra}(\mathsf{K}), \langle \boldsymbol{\theta} - \mathbf{y}, \mathbf{u} - \mathbf{p} \rangle \geq 0. \tag{B.2}$$

DEFINITION B.4 (maximal monotone operator). *Let $\mathsf{K} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ be monotone. Then K is maximal monotone if there exists no monotone operator $\mathsf{P} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$*

[6] http://github.com/mvono/PyGauss.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

such that $\mathrm{gra}(\mathsf{P})$ *properly contains* $\mathrm{gra}(\mathsf{K})$*, i.e., for every* $(\boldsymbol{\theta}, \mathbf{u}) \in \mathbb{R}^d \times \mathbb{R}^d$*,*

$$(\boldsymbol{\theta}, \mathbf{u}) \in \mathrm{gra}(\mathsf{K}) \Leftrightarrow \forall (\mathbf{y}, \mathbf{p}) \in \mathrm{gra}(\mathsf{K}), \langle \boldsymbol{\theta} - \mathbf{y}, \mathbf{u} - \mathbf{p} \rangle \geq 0. \tag{B.3}$$

DEFINITION B.5 (nonexpansiveness). *Let* $\mathsf{K} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$*. Then* K *is nonexpansive if it is Lipschitz continuous with constant* 1*, i.e., for every* $(\boldsymbol{\theta}, \mathbf{y}) \in \mathbb{R}^d \times \mathbb{R}^d$*,*

$$\left\| \mathsf{K}(\mathbf{y}) - \mathsf{K}(\boldsymbol{\theta}) \right\| \leq \left\| \mathbf{y} - \boldsymbol{\theta} \right\|, \tag{B.4}$$

where $\|\cdot\|$ *is the standard Euclidean norm.*

DEFINITION B.6 (domain). *Let* $\mathsf{K} : \mathbb{R}^d \to 2^{\mathbb{R}^d}$*. The domain of* K *is defined by*

$$\mathrm{dom}(\mathsf{K}) = \{\boldsymbol{\theta} \in \mathbb{R}^d \mid \mathsf{K}(\boldsymbol{\theta}) \neq \varnothing\}. \tag{B.5}$$

The PPA. For $\lambda > 0$, define the Moreau–Yosida resolvent operator associated to K as the operator L defined by

$$\mathsf{L} = (\mathrm{Id} + \lambda \mathsf{K})^{-1}, \tag{B.6}$$

where Id is the identity operator. The monotonicity of K implies that L is nonexpansive and its maximal monotonicity yields $\mathrm{dom}(\mathsf{L}) = \mathbb{R}^d$ [72], where the notation "dom" means the domain of the operator L. Therefore, solving the problem (5.16) is equivalent to solving the following fixed point problem for all $\boldsymbol{\theta} \in \mathbb{R}^d$:

$$\boldsymbol{\theta} = \mathsf{L}(\boldsymbol{\theta}). \tag{B.7}$$

This result suggests that finding the zeros of K can be achieved by building a sequence of iterates $\{\boldsymbol{\theta}^{(t)}\}_{t \in \mathbb{N}}$ such that for $t \in \mathbb{N}$,

$$\boldsymbol{\theta}^{(t+1)} = (\mathrm{Id} + \lambda \mathsf{K})^{-1}(\boldsymbol{\theta}^{(t)}). \tag{B.8}$$

This iteration corresponds to the PPA with an arbitrary monotone operator K.

Proof of (5.23). Applying the PPA with $\mathbf{R} = \mathbf{W} - \rho^{-1}\mathbf{A}^\top\mathbf{A}$ to (5.20) leads to

$$\begin{aligned}
\boldsymbol{\theta}^{(t+1)} &= \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min}\ g_2(\boldsymbol{\theta}) + \frac{1}{2\rho} \left\| \mathbf{A}\boldsymbol{\theta} - \mathbf{z}^{(t+1)} + \mathbf{u}^{(t)} \right\|^2 + \frac{1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^{(t)} \right\|_{\mathbf{R}}^2 \\
&= \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min}\ g_2(\boldsymbol{\theta}) + \frac{1}{2\rho} \left\| \mathbf{A}\boldsymbol{\theta} - \mathbf{z}^{(t+1)} + \mathbf{u}^{(t)} \right\|^2 + \frac{1}{2} \left\langle \mathbf{R}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(t)}\right), \boldsymbol{\theta} - \boldsymbol{\theta}^{(t)} \right\rangle \\
&= \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min}\ g_2(\boldsymbol{\theta}) + \frac{1}{2} \left(\boldsymbol{\theta}^\top \left[\frac{1}{\rho}\mathbf{A}^\top\mathbf{A} + \mathbf{R} \right] \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \left[\frac{1}{\rho}\mathbf{A}^\top \left\{ \mathbf{z}^{(t+1)} - \mathbf{u}^{(t)} \right\} + \mathbf{R}^\top \boldsymbol{\theta}^{(t)} \right] \right) \\
&= \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min}\ g_2(\boldsymbol{\theta}) + \frac{1}{2} \left(\boldsymbol{\theta}^\top \mathbf{W}\boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \left[\mathbf{W}\boldsymbol{\theta}^{(t)} + \frac{1}{\rho}\mathbf{A}^\top \left\{ \mathbf{z}^{(t+1)} - \mathbf{u}^{(t)} - \mathbf{A}\boldsymbol{\theta}^{(t)} \right\} \right] \right) \\
&= \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\arg\min}\ g_2(\boldsymbol{\theta}) + \frac{1}{2} \left\| \boldsymbol{\theta} - \left(\boldsymbol{\theta}^{(t)} + \frac{1}{\rho}\mathbf{W}^{-1}\mathbf{A}^\top \left[\mathbf{z}^{(t+1)} - \mathbf{u}^{(t)} - \mathbf{A}\boldsymbol{\theta}^{(t)} \right] \right) \right\|_{\mathbf{W}}^2 .
\end{aligned}$$

Appendix C. Details Associated to Subsection 6.2.2. The optimal value of the tuning parameter ω for the two MS schemes SOR and Richardson is given by

$$\omega_{\mathrm{SOR}}^* = \frac{2}{1 + \sqrt{1 - \rho\left(\mathbf{I}_d - \mathbf{D}^{-1}\mathbf{Q}\right)^2}}, \tag{C.1}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

where $\mathbf{D}$ is the diagonal part of $\mathbf{Q}$. Regarding the MCMC sampler based on Richardson splitting, we used the optimal value

$$\omega^*_{\text{Richardson}} = \frac{2}{\lambda_{\min}(\mathbf{Q}) + \lambda_{\max}(\mathbf{Q})}\,, \tag{C.2}$$

where $\lambda_{\min}(\mathbf{Q})$ and $\lambda_{\max}(\mathbf{Q})$ are the minimum and maximum eigenvalues of $\mathbf{Q}$, respectively. Finally, for the samplers based on SSOR splitting including Algorithm 4.3, we used the optimal relaxation parameter

$$\omega^*_{\text{SSOR}} = \frac{2}{1 + \sqrt{2\big(1 - \rho\left(\mathbf{D}^{-1}(\mathbf{L} + \mathbf{L}^\top)\right)\big)}}\,, \tag{C.3}$$

where $\mathbf{L}$ is the strictly lower triangular part of $\mathbf{Q}$.

Acknowledgments. The authors would like to thank Dr. Jérôme Idier (LS2N, France) and Prof. Jean-Christophe Pesquet (CentraleSupélec, France) for relevant feedback on an earlier version of this paper. They are also grateful to the editor and two anonymous reviewers whose comments helped to significantly improve the quality of the paper.

REFERENCES

[1] S. L. Adler, *Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions*, Phys. Rev. D, 23 (1981), pp. 2901–2904. (Cited on p. 20)

[2] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, *Fast image recovery using variable splitting and constrained optimization*, IEEE Trans. Image Process., 19 (2010), pp. 2345–2356. (Cited on p. 27)

[3] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo, *An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems*, IEEE Trans. Image Process., 20 (2011), pp. 681–695. (Cited on p. 27)

[4] E. Allen, J. Baglama, and S. Boyd, *Numerical approximation of the product of the square root of a matrix with a vector*, Linear Algebra Appl., 310 (2000), pp. 167–181, https://doi.org/10.1016/S0024-3795(00)00068-9. (Cited on p. 17)

[5] Y. Altmann, S. McLaughlin, and N. Dobigeon, *Sampling from a multivariate Gaussian distribution truncated on a simplex: A review*, in Proc. IEEE-SP Workshop Stat. and Signal Process. (SSP), Gold Coast, Australia, 2014, pp. 113–116, https://doi.org/10.1109/SSP.2014.6884588. (Cited on p. 12)

[6] S. A. Armstrong, J. E. Staunton, L. B. Silverman, R. Pieters, M. L. den Boer, M. D. Minden, S. E. Sallan, E. S. Lander, T. R. Golub, and S. J. Korsmeyer, *MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia*, Nat. Genet., 30 (2002), pp. 41–47, https://doi.org/10.1038/ng765. (Cited on pp. 13, 14)

[7] E. Aune, J. Eidsvik, and Y. Pokern, *Iterative numerical methods for sampling from high dimensional Gaussian distributions*, Stat. Comput., 23 (2013), pp. 501–521, https://doi.org/10.1007/s11222-012-9326-8. (Cited on pp. 16, 17, 36, 39)

[8] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, 1994, https://doi.org/10.1017/CBO9780511624100. (Cited on p. 22)

[9] A.-C. Barbos, F. Caron, J.-F. Giovannelli, and A. Doucet, *Clone MCMC: Parallel high-dimensional Gaussian Gibbs sampling*, in Adv. in Neural Information Process. Systems, 2017, pp. 5020–5028. (Cited on pp. 22, 24, 25, 32, 36, 37)

[10] P. Barone and A. Frigessi, *Improving stochastic relaxation for Gaussian random fields*, Probab. Eng. Inform. Sci., 4 (1990), pp. 369–389. (Cited on p. 20)

[11] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 2nd ed., Springer, 2017. (Cited on p. 49)

[12] J. Besag and C. Kooperberg, *On conditional and intrinsic autoregressions*, Biometrika, 82 (1995), pp. 733–746, https://doi.org/10.2307/2337341. (Cited on pp. 5, 12)

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag, 2006. (Cited on p. 12)

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

[14] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 3rd ed., Prentice Hall, 1994. (Cited on p. 29)
[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn., 3 (2011), pp. 1–122. (Cited on pp. 27, 34)
[16] S. Brahim-Belhouari and A. Bermak, *Gaussian process for nonstationary time series prediction*, Comput. Statist. Data Anal., 47 (2004), pp. 705–712, https://doi.org/10.1016/j.csda.2004.02.006. (Cited on p. 5)
[17] K. Bredies and H. Sun, *A proximal point analysis of the preconditioned alternating direction method of multipliers*, J. Optim. Theory Appl., 173 (2017), pp. 878–907, https://doi.org/10.1007/s10957-017-1112-5. (Cited on p. 34)
[18] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, *Stan: A probabilistic programming language*, J. Stat. Softw., 76 (2017), pp. 1–32, https://doi.org/10.18637/jss.v076.i01. (Cited on p. 46)
[19] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, J. Math. Imaging Vision, 40 (2011), pp. 120–145, https://doi.org/10.1007/s10851-010-0251-1. (Cited on p. 34)
[20] A.-L. Cholesky, *Sur la résolution numérique des systèmes d'équations linéaires*, Bull. Soc. Amis Bibl. École Polytech., 39 (1910), pp. 81–95. (Cited on p. 14)
[21] E. Chow and Y. Saad, *Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions*, SIAM J. Sci. Comput., 36 (2014), pp. A588–A608, https://doi.org/10.1137/130920587. (Cited on pp. 16, 17)
[22] E. S. Coakley and V. Rokhlin, *A fast divide-and-conquer algorithm for computing the spectra of real symmetric tridiagonal matrices*, Appl. Comput. Harmonic Anal., 34 (2013), pp. 379–414, https://doi.org/10.1016/j.acha.2012.06.003. (Cited on p. 16)
[23] N. A. C. Cressie, *Statistics for Spatial Data*, 2nd ed., Wiley, 1993. (Cited on p. 5)
[24] M. W. Davis, *Generating large stochastic simulations—the matrix polynomial approximation method*, Math. Geosci., 19 (1987), pp. 99–107, https://doi.org/10.1007/BF00898190. (Cited on pp. 15, 16, 36)
[25] O. Demir-Kavuk, H. Riedesel, and E.-W. Knapp, *Exploring classification strategies with the CoEPrA* 2006 *contest*, Bioinformatics, 26 (2010), pp. 603–609, https://doi.org/10.1093/bioinformatics/btq021. (Cited on pp. 13, 14)
[26] C. R. Dietrich and G. N. Newsam, *Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix*, SIAM J. Sci. Comput., 18 (1997), pp. 1088–1107, https://doi.org/10.1137/S1064827592240555. (Cited on p. 11)
[27] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid Monte Carlo*, Phys. Lett. B, 195 (1987), pp. 216–222. (Cited on p. 5)
[28] E. Esser, X. Zhang, and T. F. Chan, *A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science*, SIAM J. Imaging Sci., 3 (2010), pp. 1015–1046, https://doi.org/10.1137/09076934X. (Cited on p. 34)
[29] L. Fahrmeir and S. Lang, *Bayesian semiparametric regression analysis of multicategorical time-space data*, Ann. Inst. Statist. Math., 53 (2001), pp. 11–30, https://doi.org/10.1023/A:1017904118167. (Cited on p. 5)
[30] O. Féron, F. Orieux, and J. F. Giovannelli, *Gradient scan Gibbs sampler: An efficient algorithm for high-dimensional Gaussian distributions*, IEEE J. Sel. Topics Signal Process., 10 (2016), pp. 343–352, https://doi.org/10.1109/JSTSP.2015.2510961. (Cited on pp. 20, 36)
[31] C. Fox and A. Parker, *Convergence in variance of Chebyshev accelerated Gibbs samplers*, SIAM J. Sci. Comput., 36 (2014), pp. A124–A147, https://doi.org/10.1137/120900940. (Cited on pp. 23, 36, 40)
[32] C. Fox and A. Parker, *Accelerated Gibbs sampling of normal distributions using matrix splittings and polynomials*, Bernoulli, 23 (2017), pp. 3711–3743, https://doi.org/10.3150/16-BEJ863. (Cited on pp. 7, 20, 21, 22, 23, 30, 36, 37, 42, 43)
[33] D. Gabay and B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl., 2 (1976), pp. 17–40, https://doi.org/10.1016/0898-1221(76)90003-1. (Cited on p. 34)
[34] B. Galerne, Y. Gousseau, and J. Morel, *Random phase textures: Theory and synthesis*, IEEE Trans. Image Process., 20 (2011), pp. 257–267, https://doi.org/10.1109/TIP.2010.2052822. (Cited on p. 5)
[35] A. Gelman, C. Robert, N. Chopin, and J. Rousseau, *Bayesian Data Analysis*, Chapman & Hall, 1995. (Cited on p. 20)

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[36] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell., 6 (1984), pp. 721–741. (Cited on pp. 5, 13, 20)

[37] C. Gilavert, S. Moussaoui, and J. Idier, *Efficient Gaussian sampling for solving large-scale inverse problems using MCMC*, IEEE Trans. Signal Process., 63 (2015), pp. 70–80, https://doi.org/10.1109/TSP.2014.2367457. (Cited on pp. 5, 19, 36, 37, 46)

[38] J.-F. Giovannelli and J. Idier, *Regularization and Bayesian Methods for Inverse Problems in Signal and Image Processing*, 1st ed., Wiley-IEEE Press, 2015. (Cited on p. 44)

[39] P. Giudici and P. J. Green, *Decomposable graphical Gaussian model determination*, Biometrika, 86 (1999), pp. 785–801, https://doi.org/10.1093/biomet/86.4.785. (Cited on p. 5)

[40] R. Glowinski and A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, RAIRO Anal. Numer., 9 (1975), pp. 41–76, http://www.numdam.org/item/M2AN_1975__9_2_41_0. (Cited on p. 34)

[41] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, 1989. (Cited on pp. 6, 10, 14, 16, 21, 22, 42)

[42] J. Goodman and A. D. Sokal, *Multigrid Monte Carlo method. Conceptual foundations*, Phys. Rev. D, 40 (1989), pp. 2035–2071, https://doi.org/10.1103/PhysRevD.40.2035. (Cited on p. 20)

[43] P. J. Green, *Reversible jump Markov chain Monte Carlo computation and Bayesian model determination*, Biometrika, 82 (1995), pp. 711–732, https://doi.org/10.1093/biomet/82.4.711. (Cited on p. 19)

[44] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191, https://doi.org/10.1137/S0895479892241287. (Cited on p. 16)

[45] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing $A^\alpha, \log(a)$, and related matrix functions by contour integrals*, SIAM J. Numer. Anal., 46 (2008), pp. 2505–2523, https://doi.org/10.1137/070700607. (Cited on p. 17)

[46] J. Havil, *Gamma: Exploring Euler's Constant*, Princeton University Press, 2003. (Cited on p. 5)

[47] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Inst. Stand., 49 (1952), pp. 409–436. (Cited on p. 19)

[48] D. Higdon, *A primer on space-time modeling from a Bayesian perspective*, in Statistical Methods for Spatio-Temporal Systems, B. Finkenstadt, L. Held, and V. lsham, eds., Chapman & Hall/CRC, 2007, pp. 217–279. (Cited on pp. 38, 42)

[49] J. P. Hobert and G. L. Jones, *Honest exploration of intractable probability distributions via Markov chain Monte Carlo*, Stat. Sci., 16 (2001), pp. 312–334, https://doi.org/10.1214/ss/1015346317. (Cited on p. 20)

[50] J. Idier, ed., *Bayesian Approach to Inverse Problems*, Wiley, 2008, https://doi.org/10.1002/9780470611197. (Cited on pp. 10, 25)

[51] M. Ilic, T. Pettitt, and I. Turner, *Bayesian computations and efficient algorithms for computing functions of large sparse matrices*, ANZIAM J., 45 (2004), pp. 504–518, https://eprints.qut.edu.au/22511/. (Cited on p. 16)

[52] M. Ilic, I. W. Turner, and D. P. Simpson, *A restarted Lanczos approximation to functions of a symmetric matrix*, IMA J. Numer. Anal., 30 (2009), pp. 1044–1061, https://doi.org/10.1093/imanum/drp003. (Cited on pp. 16, 17, 42)

[53] H. Jeffreys, *An invariant form for the prior probability in estimation problems*, Proc. R. Soc. A, 186 (1946), pp. 453–461. (Cited on p. 13)

[54] M. Johnson, J. Saunderson, and A. Willsky, *Analyzing Hogwild parallel Gaussian Gibbs sampling*, in Adv. in Neural Information Process. Systems, 2013, pp. 2715–2723. (Cited on pp. 7, 22, 24, 25, 32, 36)

[55] R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal, *Markov chain Monte Carlo in practice: A roundtable discussion*, Amer. Statist., 52 (1998), pp. 93–100. (Cited on p. 46)

[56] J. Kittler and J. Föglein, *Contextual classification of multispectral pixel data*, Image Vis. Comput., 2 (1984), pp. 13–29, https://doi.org/10.1016/0262-8856(84)90040-4. (Cited on p. 5)

[57] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324, https://doi.org/10.1109/5.726791. (Cited on pp. 13, 14)

[58] M. Li, D. Sun, and K.-C. Toh, *A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization*, SIAM J. Optim., 26 (2016), pp. 922–950, https://doi.org/10.1137/140999025. (Cited on p. 34)

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[59] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed., Springer, 2009. (Cited on p. 11)
[60] Y. Li and S. K. Ghosh, *Efficient sampling methods for truncated multivariate normal and Student-t distributions subject to linear inequality constraints*, J. Stat. Theory Pract., 9 (2015), pp. 712–732, https://doi.org/10.1080/15598608.2014.996690. (Cited on p. 12)
[61] A. C. Likas and N. P. Galatsanos, *A variational approach for Bayesian blind image deconvolution*, IEEE Trans. Signal Process., 52 (2004), pp. 2222–2233. (Cited on p. 44)
[62] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, 2001. (Cited on p. 46)
[63] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003. (Cited on p. 38)
[64] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed., Academic Press, 2008. (Cited on p. 16)
[65] A. Mantoglou and J. L. Wilson, *The turning bands method for simulation of random fields using line generation by a spectral method*, Water Resour. Res., 18 (1982), pp. 1379–1394, https://doi.org/10.1029/WR018i005p01379. (Cited on p. 11)
[66] Y. Marnissi, D. Abboud, E. Chouzenoux, J.-C. Pesquet, M. El-Badaoui, and A. Benazza-Benyahia, *A data augmentation approach for sampling Gaussian models in high dimension*, in Proc. European Signal Process. Conf. (EUSIPCO), Coruna, Spain, 2019, pp. 1–5. (Cited on pp. 7, 25, 26, 27, 30, 36)
[67] Y. Marnissi, E. Chouzenoux, A. Benazza-Benyahia, and J.-C. Pesquet, *An auxiliary variable method for Markov chain Monte Carlo algorithms in high dimension*, Entropy, 20 (2018), art. 110, https://doi.org/10.3390/e20020110. (Cited on pp. 5, 7, 25, 26, 44)
[68] B. Martinet, *Brève communication. Régularisation d'inéquations variationnelles par approximations successives*, ESAIM Math. Model. Numer. Anal., 4 (1970), pp. 154–158, http://www.numdam.org/item/M2AN_1970__4_3_154_0. (Cited on p. 33)
[69] B. Martinet, *Determination approaché d'un point fixe dune application pseudocontractante. Cas de l'application prox*, C. R. Math. Acad. Sci. Paris, 274 (1972), pp. 163–165. (Cited on p. 33)
[70] J. Mason and D. Handscomb, *Chebyshev Polynomials*, CRC Press, 2002. (Cited on p. 16)
[71] J. M. Mejía and I. Rodríguez-Iturbe, *On the synthesis of random field sampling from the spectrum: An application to the generation of hydrologic spatial processes*, Water Resour. Res., 10 (1974), pp. 705–711, https://doi.org/10.1029/WR010i004p00705. (Cited on p. 11)
[72] G. J. Minty, *Monotone (nonlinear) operators in Hilbert space*, Duke Math. J., 29 (1962), pp. 341–346, https://doi.org/10.1215/S0012-7094-62-02933-2. (Cited on p. 50)
[73] A. de Moivre, *The Doctrine of Chances: Or, a Method of Calculating the Probability of Events in Play*, 1st ed., W. Pearson, 1718. (Cited on p. 5)
[74] R. Molina, J. Mateos, and A. K. Katsaggelos, *Blind deconvolution using a variational approach to parameter, image, and blur estimation*, IEEE Trans. Image Process., 15 (2006), pp. 3715–3727. (Cited on p. 44)
[75] R. Molina and B. D. Ripley, *Using spatial models as priors in astronomical image analysis*, J. Appl. Stat., 16 (1989), pp. 193–206. (Cited on p. 44)
[76] J. J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bull. de la Soc. Math. de France, 93 (1965), pp. 273–299, https://doi.org/10.24033/bsmf.1625. (Cited on p. 33)
[77] F. Orieux, O. Feron, and J. F. Giovannelli, *Sampling high-dimensional Gaussian distributions for general linear inverse problems*, IEEE Signal Process. Lett., 19 (2012), pp. 251–254, https://doi.org/10.1109/LSP.2012.2189104. (Cited on pp. 19, 36)
[78] F. Orieux, J.-F. Giovannelli, and T. Rodet, *Bayesian estimation of regularization and point spread function parameters for Wiener–Hunt deconvolution*, J. Opt. Soc. Am. A, 27 (2010), pp. 1593–1607, https://doi.org/10.1364/JOSAA.27.001593. (Cited on pp. 11, 25, 44)
[79] G. Papandreou and A. L. Yuille, *Gaussian sampling by local perturbations*, in Adv. in Neural Information Process. Systems, 2010, pp. 1858–1866. (Cited on pp. 18, 19, 36, 37)
[80] T. Park and G. Casella, *The Bayesian Lasso*, J. Amer. Stat. Assoc., 103 (2008), pp. 681–686, https://doi.org/10.1198/016214508000000337. (Cited on p. 5)
[81] A. Parker and C. Fox, *Sampling Gaussian distributions in Krylov spaces with conjugate gradients*, SIAM J. Sci. Comput., 34 (2012), pp. B312–B334, https://doi.org/10.1137/110831404. (Cited on pp. 12, 19, 20, 36, 38, 39, 40, 42)
[82] M. Pereira and N. Desassis, *Efficient simulation of Gaussian Markov random fields by Chebyshev polynomial approximation*, Spat. Stat., 31 (2019), art. 100359, https://doi.org/10.1016/j.spasta.2019.100359. (Cited on pp. 16, 36)

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[83] N. G. Polson, J. G. Scott, and J. Windle, *Bayesian inference for logistic models using Pólya-Gamma latent variables*, J. Amer. Stat. Assoc., 108 (2013), pp. 1339–1349, https://doi.org/10.1080/01621459.2013.829001. (Cited on p. 5)

[84] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, 2007. (Cited on p. 16)

[85] C. E. Rasmussen, *Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals*, in Bayesian Statistics 7, Oxford University Press, 2003, pp. 651–660. (Cited on p. 38)

[86] C. P. Robert, *The Bayesian Choice: A Decision-Theoretic Motivation*, Springer, 2001. (Cited on pp. 13, 18)

[87] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*, Springer, 2004. (Cited on pp. 13, 20, 30, 31, 42)

[88] G. O. Roberts and S. K. Sahu, *Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler*, J. Roy. Stat. Soc. Ser. B, 59 (1997), pp. 291–317, https://doi.org/10.1111/1467-9868.00070. (Cited on pp. 20, 23)

[89] G. O. Roberts and R. L. Tweedie, *Exponential convergence of Langevin distributions and their discrete approximations*, Bernoulli, 2 (1996), pp. 341–363, https://doi.org/10.2307/3318418. (Cited on p. 5)

[90] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898, https://doi.org/10.1137/0314056. (Cited on pp. 7, 20, 28, 32, 33)

[91] H. Rue, *Fast sampling of Gaussian Markov random fields*, J. Roy. Stat. Soc. Ser. B, 63 (2001), pp. 325–338, https://doi.org/10.1111/1467-9868.00288. (Cited on pp. 6, 10, 15, 19, 36)

[92] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, Chapman & Hall/CRC, 2005. (Cited on pp. 5, 6, 10, 11, 12, 15, 20, 25, 42)

[93] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003, https://doi.org/10.1137/1.9780898718003. (Cited on p. 21)

[94] E. M. Scheuer and D. S. Stoller, *On the generation of normal random vectors*, Technometrics, 4 (1962), pp. 278–281, https://doi.org/10.2307/1266625. (Cited on p. 36)

[95] M. K. Schneider and A. S. Willsky, *A Krylov subspace method for covariance approximation and simulation of random processes and fields*, Multidimens. Syst. Signal Process., 14 (2003), pp. 295–318, https://doi.org/10.1023/A:1023530718764. (Cited on pp. 19, 20)

[96] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell., 22 (2000), pp. 888–905, https://doi.org/10.1109/34.868688. (Cited on p. 38)

[97] M. Shinozuka and C.-M. Jan, *Digital simulation of random processes and its applications*, J. Sound Vib., 25 (1972), pp. 111–128, https://doi.org/10.1016/0022-460X(72)90600-1. (Cited on p. 11)

[98] D. P. Simpson, I. W. Turner, and A. N. Pettitt, *Fast Sampling from a Gaussian Markov Random Field Using Krylov Subspace Approaches*, 2008, https://eprints.qut.edu.au/14376/. (Cited on pp. 16, 17, 36)

[99] D. P. Simpson, I. W. Turner, C. M. Strickland, and A. N. Pettitt, *Scalable Iterative Methods for Sampling from Massive Gaussian Random Vectors*, preprint, https://arxiv.org/abs/1312.1476, 2013. (Cited on p. 16)

[100] G. W. Stewart, *Matrix Algorithms. Volume 2: Eigensystems*, SIAM, 2001, https://doi.org/10.1137/1.9780898718058. (Cited on p. 17)

[101] D. B. Thomas, W. Luk, P. H. Leong, and J. D. Villasenor, *Gaussian random number generators*, ACM Comput. Surv., 39 (2007), art. 11, https://doi.org/10.1145/1287620.1287622. (Cited on p. 9)

[102] W. F. Trench, *An algorithm for the inversion of finite Toeplitz matrices*, SIAM J. Appl. Math., 12 (1964), pp. 512–522, https://doi.org/10.1137/0112045. (Cited on p. 20)

[103] D. W. Vasco, L. R. Johnson, and O. Marques, *Global Earth structure: Inference and assessment*, Geophys. J. Int., 137 (1999), pp. 381–407, https://doi.org/10.1046/j.1365-246X.1999.00823.x. (Cited on p. 38)

[104] M. Vono, N. Dobigeon, and P. Chainais, *Split-and-augmented Gibbs sampler: Application to large-scale inference problems*, IEEE Trans. Signal Process., 67 (2019), pp. 1648–1661, https://doi.org/10.1109/TSP.2019.2894825. (Cited on pp. 25, 27, 36, 44)

[105] M. Vono, N. Dobigeon, and P. Chainais, *Asymptotically exact data augmentation: Models, properties, and algorithms*, J. Comput. Graph. Statist., 30 (2021), pp. 335–348 (Cited on pp. 27, 32)

[106] M. Welling and Y. W. Teh, *Bayesian learning via stochastic gradient Langevin dynamics*, in Proc. Int. Conf. Machine Learning (ICML), 2011, pp. 681–688. (Cited on p. 5)

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[107] S. Wilhelm and B. Manjunath, *tmvtnorm: A package for the truncated multivariate normal distribution*, The R Journal, 2 (2010), pp. 25–29, https://doi.org/10.32614/RJ-2010-005. (Cited on p. 12)

[108] C. K. I. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, MIT Press, 2006. (Cited on p. 38)

[109] A. T. A. Wood and G. Chan, *Simulation of stationary Gaussian processes in* $[0,1]^d$, J. Comput. Graph. Stat., 3 (1994), pp. 409–432, https://doi.org/10.1080/10618600.1994.10474655. (Cited on p. 11)

[110] X. Zhang, M. Burger, and S. Osher, *A unified primal-dual algorithm framework based on Bregman iteration*, J. Sci. Comput., 46 (2011), pp. 20–46, https://doi.org/10.1007/s10915-010-9408-8. (Cited on p. 34)

Downloaded 02/03/22 to 193.51.236.101 Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.