Bernoulli **23**(4B), 2017, 3711–3743
DOI: 10.3150/16-BEJ863

Accelerated Gibbs sampling of normal distributions using matrix splittings and polynomials

COLIN FOX[1] and ALBERT PARKER[2]

[1]*Department of Physics, University of Otago, Dunedin, New Zealand. E-mail: fox@physics.otago.ac.nz*
[2]*Center for Biofilm Engineering, Department of Mathematical Sciences, Montana State University, Bozeman, MT, USA. E-mail: parker@math.montana.edu*

Standard Gibbs sampling applied to a multivariate normal distribution with a specified precision matrix is equivalent in fundamental ways to the Gauss–Seidel iterative solution of linear equations in the precision matrix. Specifically, the iteration operators, the conditions under which convergence occurs, and geometric convergence factors (and rates) are identical. These results hold for arbitrary matrix splittings from classical iterative methods in numerical linear algebra giving easy access to mature results in that field, including existing convergence results for antithetic-variable Gibbs sampling, REGS sampling, and generalizations. Hence, efficient deterministic stationary relaxation schemes lead to efficient generalizations of Gibbs sampling. The technique of polynomial acceleration that significantly improves the convergence rate of an iterative solver derived from a *symmetric* matrix splitting may be applied to accelerate the equivalent generalized Gibbs sampler. Identicality of error polynomials guarantees convergence of the inhomogeneous Markov chain, while equality of convergence factors ensures that the optimal solver leads to the optimal sampler. Numerical examples are presented, including a Chebyshev accelerated SSOR Gibbs sampler applied to a stylized demonstration of low-level Bayesian image reconstruction in a large 3-dimensional linear inverse problem.

Keywords: Bayesian inference; Gaussian Markov random field; Gibbs sampling; matrix splitting; multivariate normal distribution; non-stationary stochastic iteration; polynomial acceleration

1. Introduction

The Metropolis–Hastings algorithm for MCMC was introduced to main-stream statistics around 1990 (Robert and Casella [48]), though prior to that the Gibbs sampler provided a coherent approach to investigating distributions with Markov random field structure (Turčin [60], Grenander [32], Geman and Geman [25], Gelfand and Smith [23], Besag and Green [11], Sokal [58]). The Gibbs sampler may be thought of as a particular Metropolis–Hastings algorithm that uses the conditional distributions as proposal distributions, with acceptance probability always equal to 1 (Geyer [26]).

In statistics, the Gibbs sampler is popular because of ease of implementation (see, e.g., Roberts and Sahu [51]), when conditional distributions are available in the sense that samples may be drawn from the full conditionals. However, the Gibbs sampler is not often presented as an efficient algorithm, particularly for massive models. In this work, we show that generalized and

1350-7265

accelerated Gibbs samplers are contenders for the fastest sampling algorithms for normal target distributions, because they are equivalent to the fastest algorithms for solving systems of linear equations.

Almost all current MCMC algorithms, including Gibbs samplers, simulate a *fixed* transition kernel that induces a *homogeneous* Markov chain that converges *geometrically* in distribution to the desired target distribution. In this aspect, modern variants of the Metropolis–Hastings algorithm are unchanged from the Metropolis algorithm as first implemented in the 1950s. The adaptive Metropolis algorithm of Haario *et al.* [34] (see also Roberts and Rosenthal [50]) is an exception, though it converges to a geometrically convergent Metropolis–Hastings algorithm that bounds convergence behaviour.

We focus on the application of Gibbs sampling to drawing samples from a multivariate normal distribution with a given covariance or precision matrix. Our concern is to develop generalized Gibbs samplers with optimal geometric, or better than geometric, distributional convergence by drawing on ideas in numerical computation, particularly the mature field of computational linear algebra. We apply the matrix-splitting formalism to show that fixed-scan Gibbs sampling from a multivariate normal is equivalent in fundamental ways to the stationary linear iterative solvers applied to systems of equations in the precision matrix.

Stationary iterative solvers are now considered to be very slow precisely because of their geometric rate of convergence, and are no longer used for large systems. However, they remain a basic building block in the most efficient linear solvers. By establishing equivalence of error polynomials, we provide a route whereby acceleration techniques from numerical linear algebra may be applied to Gibbs sampling from normal distributions. The fastest solvers employ non-stationary iterations, hence the equivalent generalized Gibbs sampler induces an inhomogeneous Markov chain. Explicit calculation of the error polynomial guarantees convergence, while control of the error polynomial gives optimal performance.

The adoption of the matrix splitting formalism gives the following practical benefits in the context of fixed-scan Gibbs sampling from normal targets:

1. a one-to-one equivalence between generalized Gibbs samplers and classical linear iterative solvers;
2. rates of convergence and error polynomials for the Markov chain induced by a generalized Gibbs sampler;
3. acceleration of the Gibbs sampler to induce an inhomogeneous Markov chain that achieves the optimal error polynomial, and hence has optimal convergence rate for expectations and in distribution;
4. numerical estimates of convergence rate of the (accelerated) Gibbs sampler in a single chain and a priori estimates of number of iterations to convergence;
5. access to preconditioning, whereby the sampling problem is transformed into an equivalent problem for which the accelerated Gibbs sampler has improved convergence rate.

Some *direct* linear solvers have already been adapted to sampling from multivariate normal distributions, with Rue [52] demonstrating the use of solvers based on Cholesky factorization to allow computationally efficient sampling. This paper extends the connection to the *iterative* linear solvers. Since iterative methods are the most efficient for massive linear systems, the associated samplers will be the most efficient for very high dimensional normal targets.

1.1. Context and overview of results

The Cholesky factorization is the conventional way to produce samples from a moderately sized multivariate normal distribution (Rue [52], Rue and Held [53]), and is also the preferred method for solving moderately sized linear systems. For *large* linear systems, iterative solvers are the methods of choice due to their inexpensive cost per iteration, and small computer memory requirements.

Gibbs samplers applied to normal distributions are essentially identical to stationary iterative methods from numerical linear algebra. This connection was exploited by Adler [1], and independently by Barone and Frigessi [8], who noted that the component-wise Gibbs sampler is a stochastic version of the Gauss–Seidel linear solver, and accelerated the Gibbs sampler by introducing a relaxation parameter to implement the stochastic version of the successive over-relaxation (SOR) of Gauss–Seidel. This pairing was further analyzed by Goodman and Sokal [30].

This equivalence is depicted in panels A and B of Figure 1. Panel B shows the contours of a normal density $\pi(\mathbf{x})$, and a sequence of coordinate-wise conditional samples taken by the Gibbs sampler applied to π. Panel A shows the contours of the quadratic minus $\log(\pi(\mathbf{x}))$ and the Gauss–Seidel sequence of coordinate optimizations,[1] or, equivalently, solves of the normal equations $\nabla \log \pi(\mathbf{x}) = 0$. Note how in Gauss–Seidel the step sizes decrease towards convergence, which is a tell-tale sign that convergence (in value) is geometric. In Section 4, we will show that the iteration operator is identical to that of the Gibbs sampler in panel B, and hence the Gibbs sampler also converges geometrically (in distribution). Slow convergence of these algorithms is usually understood in terms of the same intuition; high correlations correspond to long narrow contours, and lead to small steps in coordinate directions and many iterations being required to move appreciably along the long axis of the target function.

Roberts and Sahu [51] considered forward then backward sweeps of coordinate-wise Gibbs sampling, with relaxation parameter, to give a sampler they termed the *REGS* sampler. This is a stochastic version of the symmetric-SOR (SSOR) iteration, which comprises forward then backward sweeps of SOR.

The equality of iteration operators and error polynomials, for these pairs of fixed-scan Gibbs samplers and iterative solvers, allows existing convergence results in numerical analysis texts (for example, Axelsson [5], Golub and Van Loan [29], Nevanlinna [45], Saad [54], Young [64]) to be used to establish convergence results for the corresponding Gibbs sampler. Existing results for rates of distributional convergence by fixed-sweep Gibbs samplers (Adler [1], Barone and Frigessi [8], Liu *et al.* [39], Roberts and Sahu [51]) may be established this way.

The methods of Gauss–Seidel, SOR, and SSOR, give stationary linear iterations that were used as linear solvers in the 1950s, and are now considered very slow. The corresponding fixed-scan Gibbs samplers are slow for precisely the same reason. The last fifty years has seen an explosion of theoretical results and algorithmic development that have made linear solvers faster and more efficient, so that for large problems, stationary methods are used as preconditioners at best, while the method of preconditioned conjugate gradients, GMRES, multigrid, or fast-multipole methods

[1] Gauss–Seidel optimization was rediscovered by Besag [10] as iterated conditional modes.

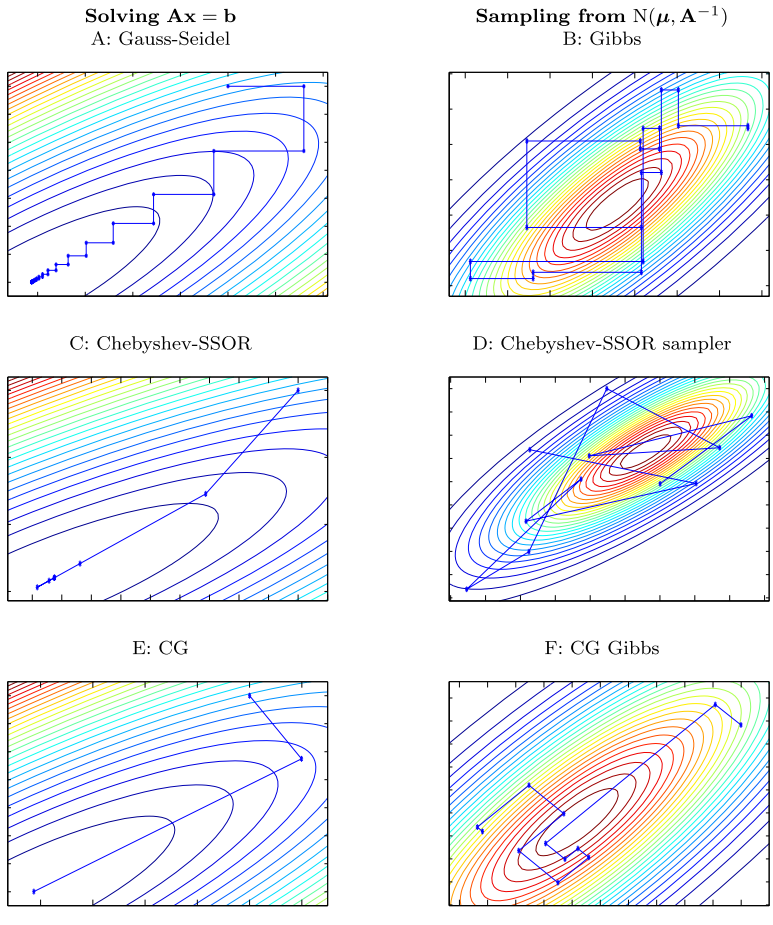

Figure 1. The panels in the left column show the contours of a quadratic function $\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$ in two dimensions and the iteration paths for some common optimizers towards the minimizer $\boldsymbol{\mu} = \mathbf{A}^{-1}\mathbf{b}$, or equivalently the path of iterative linear solvers of $\mathbf{A}\mathbf{x} = \mathbf{b}$. The right column presents the iteration paths of the samplers corresponding to each linear solver, along with the contours of the normal density $k\exp\{-\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}\}$, where k is the normalizing constant. In all panels, the matrix $\mathbf{A}$ has eigenpairs $\{(10, [1\ 1]^T), (1, [1\ -1]^T)\}$. The Gauss–Seidel solver took 45 iterations to converge to $\boldsymbol{\mu}$ (shown are the 90 coordinate steps; each iteration is a "sweep" of the two coordinate directions), the Chebyshev polynomial accelerated SSOR required just 16 iterations to converge, while CG finds the minimizer in 2 iterations. For each of the samplers, 10 iterations are shown (the 20 coordinate steps are shown for the Gibbs sampler). The correspondence between these linear solvers/optimizers and samplers is treated in the text (CG in the supplementary material [20]).

are the current state-of-the-art for solving linear systems in a finite number of steps (Saad and van der Vorst [55]).

Linear iterations derived from a symmetric splitting may be sped up by polynomial acceleration, particularly Chebyshev acceleration that results in optimal error reduction amongst methods that have a fixed non-stationary iteration structure (Fox and Parker [21], Axelsson [5]). The Chebyshev accelerated SSOR solver and corresponding Chebyshev accelerated SSOR sampler (Fox and Parker [19]) are depicted in panels C and D of Figure 1. Both the solver and sampler take steps that are more aligned with the long axis of the target, compared to the coordinate-wise algorithms, and hence achieve faster convergence. However, the step size of Chebyshev-SSOR solving still decreases towards convergence, and hence convergence for both solver and sampler is still asymptotically geometric, albeit with much improved rate.

Fox and Parker [19] considered point-wise convergence of the mean and variance of a Gibbs SSOR sampler accelerated by Chebyshev polynomials. In this paper, we prove convergence in distribution for Gibbs samplers corresponding to *any* matrix splitting and accelerated by *any* polynomial that is independent of the Gibbs iterations. We then apply a polynomial accelerated sampler to solve a massive Bayesian linear inverse problem that is infeasible to solve using conventional techniques.

Chebyshev acceleration requires estimates of the extreme eigenvalues of the error operator, which we obtain via a conjugate-gradient (CG) algorithm at no significant computational cost (Meurant [41]). The CG algorithm itself may be adapted to sample from normal distributions; the CG solver and corresponding sampler, depicted in panels E and F of Figure 1, were analysed by Parker and Fox [47] and is discussed in the supplementary material [20].

1.2. Structure of the paper

In Section 2, we review efficient methods for sampling from normal distributions, highlighting Gibbs sampling in various algorithmic forms. Standard results for stationary iterative solvers are presented in Section 3. Theorems in Section 4 establish equivalence of convergence and convergence factors for iterative solvers and Gibbs samplers. Application of polynomial acceleration methods to linear solvers and Gibbs sampling is given in Section 5, including explicit expressions for convergence of the first and second moments of a polynomial accelerated sampler, from which it follows that distributional convergence occurs with rate determined by the same error polynomial. Numerical verification of convergence results is presented in Section 6.

2. Sampling from multivariate normal distributions

We consider the problem of sampling from an n-dimensional normal distribution $\mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ defined by the mean n-vector $\boldsymbol{\mu}$, and the $n \times n$ symmetric and positive definite (SPD) covariance matrix $\boldsymbol{\Sigma}$. Since if $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma})$ then $\mathbf{z} + \boldsymbol{\mu} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, it often suffices to consider drawing samples from normal distributions with zero mean. An exception is when $\boldsymbol{\mu}$ is defined implicitly, which we discuss in Section 4.1.

In Bayesian formulations of inverse problems that use a GMRF as a prior distribution, typically the precision matrix $\mathbf{A} = \boldsymbol{\Sigma}^{-1}$ is explicitly modeled and available (Rue and Held [53],

Higdon [35]), perhaps as part of a hierarchical model (Banerjee *et al.* [6]). Typically then the precision matrix (conditioned on hyperparameters) is large though sparse, if the neighborhoods specifying conditional independence are small. We are particularly interested in this case, and throughout the paper will focus on sampling from $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ when $\mathbf{A}$ is sparse and large, or when some other property makes operating by $\mathbf{A}$ easy, that is, one can evaluate $\mathbf{Ax}$ for any vector $\mathbf{x}$.

Standard sampling methods for moderately sized normal distributions utilize the Cholesky factorization (Rue [52], Rue and Held [53]) since it is fast, incurring approximately $(1/3)n^3$ floating point operations (flops) and is backwards stable (Watkins [62]). Samples can also be drawn using the more expensive eigen-decomposition (Rue and Held [53]), that costs approximately $(10/3)n^3$ flops, or more generally using mutually conjugate vectors (Fox [18], Parker and Fox [47]). For stationary Gaussian random fields defined on the lattice, Fourier methods can lead to efficient sampling for large problems (Gneiting *et al.* [28]).

Algorithm 1 shows the steps for sampling from $\mathrm{N}(\mathbf{0}, \mathbf{\Sigma})$ using Cholesky factorization, when the covariance matrix $\mathbf{\Sigma}$ is available (Neal [42], MacKay [40], Higdon [35]).

Algorithm 1: Cholesky sampling using a covariance matrix $\mathbf{\Sigma}$

input : Covariance matrix $\mathbf{\Sigma}$
output: $\mathbf{y} \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma})$

Cholesky factor $\mathbf{\Sigma} = \mathbf{C}\mathbf{C}^T$;
sample $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$;
$\mathbf{y} = \mathbf{C}\mathbf{z}$;

When the precision matrix $\mathbf{A}$ is available, a sample $\mathbf{y} \sim \mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ may be drawn using Algorithm 2 given by Rue [52] (see also Rue and Held [53]).

Algorithm 2: Cholesky sampling using a precision matrix $\mathbf{A}$

input : Precision matrix $\mathbf{A}$
output: $\mathbf{y} \sim \mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$

Cholesky factor $\mathbf{A} = \mathbf{B}\mathbf{B}^T$;
sample $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$;
solve $\mathbf{B}^T\mathbf{y} = \mathbf{z}$ by back substitution;

The computational cost of Algorithm 2 depends on the bandwidth of $\mathbf{A}$, that also bounds the bandwidth of the Cholesky factor $\mathbf{B}$. For a bandwidth b, calculation of the Cholesky factorization requires $\mathcal{O}(b^2 n)$ flops, which provides savings over the full-bandwidth case when $b \ll n/2$ (Golub and Van Loan [29], Rue [52], Watkins [62]). For GMRF's defined over 2-dimensional domains, the use of a bandwidth reducing permutation often leads to substantial computational

savings (Rue [52], Watkins [62]). In 3-dimensions and above, typically no permutation exists that can significantly reduce the bandwidth below $n^{2/3}$, hence the cost of sampling by Cholesky factoring is at least $\mathcal{O}(n^{7/3})$ flops. Further, Cholesky factorizing requires that the precision matrix and the Cholesky factor be stored in computer memory, which can be prohibitive for large problems. In Section 6, we give an example of sampling from a large GMRF for which Cholesky factorization is prohibitively expensive.

2.1. Gibbs sampling from a normal distribution

Iterative samplers, such as Gibbs, are an attractive option when drawing samples from high dimensional multivariate normal distributions due to their inexpensive cost per iteration and small computer memory requirements (only vectors of size n need be stored). If the precision matrix is sparse with $\mathcal{O}(n)$ non-zero elements, then, regardless of the bandwidth, iterative methods cost only about $2n$ flops per iteration, which is comparable with sparse Cholesky factorizations. However, when the bandwidth is $\mathcal{O}(n)$, the cost of the Cholesky factorization is high at $\mathcal{O}(n^3)$ flops, while iterative methods maintain their inexpensive cost per iteration. Iterative methods are then preferable when requiring significantly fewer than $\mathcal{O}(n^2)$ iterations for adequate convergence. In the examples presented in Section 6 we find that $\mathcal{O}(n)$ iterations give convergence to machine precision, so the iterative methods are preferable for large problems.

2.1.1. *Componentwise formulation*

One of the simplest iterative sampling methods is the component-sweep Gibbs sampler (Geman and Geman [25], Gelman *et al.* [24], Gilks *et al.* [27], Rue and Held [53]). Let $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T \in \Re^n$ denote a vector in terms of its components, and let $\mathbf{A}$ be an $n \times n$ precision matrix with elements $\{a_{ij}\}$. One sweep over all n components can be written as in Algorithm 3 (Barone and Frigessi [8]), showing that the algorithm can be implemented using vector and scalar operations only, and storage or inversion of the precision matrix $\mathbf{A}$ is not required.

Algorithm 3: Component-sweep Gibbs sampling using a precision matrix $\mathbf{A}$

input : Precision matrix $\mathbf{A}$, initial state $\mathbf{y}^{(0)} = (y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)})^T$, and maximum iteration $k_{\max}$

output: $\{\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(k_{\max})}\}$ where $\mathbf{y}^{(k)} \xrightarrow{\mathcal{D}} \mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ as $k \to \infty$

for $k = 1, 2, \ldots, k_{\max}$ **do**
  for $i = 1, 2, \ldots, n$ **do**
    Sample $z \sim \mathrm{N}(0, 1)$;
    $y_i^{(k)} = \frac{z}{\sqrt{a_{ii}}} - \frac{1}{a_{ii}}\left(\sum_{j>i} a_{ij} y_j^{(k-1)} - \sum_{j<i} a_{ij} y_j^{(k)}\right)$
  end
end

The index k may be omitted (and with "=" interpreted as assignment) to give an algorithm that can be evaluated *in place*, requiring minimal storage.

2.1.2. *Matrix formulation*

One iteration in Algorithm 3 consists of a sweep over all n components of $\mathbf{y}^{(k)}$ in sequence. The iteration can be written succinctly in the matrix form (Goodman and Sokal [30])

$$\mathbf{y}^{(k+1)} = -\mathbf{D}^{-1}\mathbf{L}\mathbf{y}^{(k+1)} - \mathbf{D}^{-1}\mathbf{L}^T\mathbf{y}^{(k)} + \mathbf{D}^{-1/2}\mathbf{z}^{(k)}, \tag{1}$$

where $\mathbf{z}^{(k)} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{D} = \mathrm{diag}(\mathbf{A})$, and $\mathbf{L}$ is the strictly lower triangular part of $\mathbf{A}$. This equation makes clear that the computational cost of each sweep is about $2n^2$ flops, when $\mathbf{A}$ is dense, due to multiplication by the triangular matrices $\mathbf{L}$ and $\mathbf{L}^T$, and $\mathcal{O}(n)$ flops when $\mathbf{A}$ is sparse.

Extending this formulation to sweeps over any other *fixed* sequence of coordinates is achieved by putting $\mathbf{PAP}^T$ in place of $\mathbf{A}$ for some permutation matrix $\mathbf{P}$. The use of *random* sweep Gibbs sampling has also been suggested (Amit and Grenander [4], Fishman [17], Liu *et al.* [39], Roberts and Sahu [51]), though we do not consider that here.

2.1.3. *Convergence*

If the iterates $\mathbf{y}^{(k)}$ in (1) converge in distribution to a distribution Π which is independent of the starting state $\mathbf{y}^{(0)}$, then the sampler is *convergent*, and we write

$$\mathbf{y}^{(k)} \xrightarrow{\mathcal{D}} \Pi.$$

It is well known that the iterates $\mathbf{y}^{(k)}$ in the Gibbs sampler (1) converge in distribution geometrically to $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1}) = \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma})$ (Roberts and Sahu [51]). We consider geometric convergence in detail in Section 4.

3. Linear stationary iterative methods as linear equation solvers

Our work draws heavily on existing results for stationary linear iterative methods for solving linear systems. Here we briefly review the main results that we will use.

Consider a system of linear equations written as the matrix equation

$$\mathbf{Ax} = \mathbf{b}, \tag{2}$$

where $\mathbf{A}$ is a given $n \times n$ nonsingular matrix and $\mathbf{b}$ is a given n-dimensional vector. The problem is to find an n-dimensional vector $\mathbf{x}$ satisfying equation (2). Later we will consider the case where $\mathbf{A}$ is symmetric positive definite (SPD) as holds for covariance and precision matrices (Feller [16]).

3.1. Matrix splitting form of stationary iterative algorithms

A common class of methods for solving (2) are the linear iterative methods based on a *splitting* of $\mathbf{A}$ into $\mathbf{A} = \mathbf{M} - \mathbf{N}$. The matrix splitting is the standard way of expressing and analyzing linear iterative algorithms, following its introduction by Varga [61]. The system (2) is then transformed to $\mathbf{Mx} = \mathbf{Nx} + \mathbf{b}$ or, if $\mathbf{M}$ is nonsingular, $\mathbf{x} = \mathbf{M}^{-1}\mathbf{Nx} + \mathbf{M}^{-1}\mathbf{b}$. The iterative methods use this equation to compute successively better approximations $\mathbf{x}^{(k)}$ to the solution using the iteration step

$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}\mathbf{N}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{b} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{g}. \tag{3}$$

We follow the standard terminology used for these methods (see, e.g., Axelsson [5], Golub and Van Loan [29], Saad [54], Young [64]). Such methods are termed linear stationary iterative methods (of the first degree); they are stationary[2] because the *iteration matrix* $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$ and the vector $\mathbf{g} = \mathbf{M}^{-1}\mathbf{b}$ do not depend on k. The splitting is *symmetric* when both $\mathbf{M}$ and $\mathbf{N}$ are symmetric matrices. The iteration, and splitting, is *convergent* if $\mathbf{x}^{(k)}$ tends to a limit independent of $\mathbf{x}^{(0)}$, the limit being $\mathbf{A}^{-1}\mathbf{b}$ (see, e.g., Young [64], Theorem 5.2).

The iteration (3) is often written in the residual form so that convergence may be monitored in terms of the norm of the residual vector, and emphasizes that $\mathbf{M}^{-1}$ is acting as an approximation to $\mathbf{A}^{-1}$, as in Algorithm 4.

Algorithm 4: Stationary iterative solve of $\mathbf{Ax} = \mathbf{b}$

input : The splitting $\mathbf{M}$, $\mathbf{N}$ of $\mathbf{A}$, initial state $\mathbf{x}^{(0)}$
output: $\mathbf{x}^{(k)}$ approximating $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$

$k = 0$
repeat
  $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$;
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{r}^{(k)}$;
  increment k;
until $\|\mathbf{r}^{(k)}\|$ *is sufficiently small*;

In computational algorithms, it is important to note that the symbol $\mathbf{M}^{-1}\mathbf{r}$ is interpreted as "solve the system $\mathbf{Mu} = \mathbf{r}$ for $\mathbf{u}$" rather than "form the inverse of $\mathbf{M}$ and multiply $\mathbf{r}$ by $\mathbf{M}^{-1}$" since the latter is much more computationally expensive (about $2n^3$ flops (Watkins [62])). Thus, the practicality of a splitting depends on the ease with which one can solve $\mathbf{Mu} = \mathbf{r}$ for any vector $\mathbf{r}$.

[2]This use of *stationary* corresponds to the term *homogeneous* when referring to a Markov chain. It is not to be confused with a *stationary distribution* that is invariant under the iteration. Later we will develop non-stationary iterations, inducing a non-homogeneous Markov chain that will, however, preserve the target distribution at each iterate.

3.1.1. *The Gauss–Seidel algorithm*

Many splittings of the matrix $\mathbf{A}$ use the terms in the expansion $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$ where $\mathbf{L}$ is the strictly lower triangular part of $\mathbf{A}$, $\mathbf{D}$ is the diagonal of $\mathbf{A}$, and $\mathbf{U}$ is the strictly upper triangular part.

For example, choosing $\mathbf{M} = \mathbf{L} + \mathbf{D}$ (so $\mathbf{N} = -\mathbf{U}$) allows $\mathbf{Mu} = \mathbf{r}$ to be solved by "forward substitution" (at a cost of n^2 flops when $\mathbf{A}$ is dense), and hence does not require inversion or Gauss-elimination of $\mathbf{M}$ (which would cost $2/3n^3$ flops when $\mathbf{A}$ is dense). Using this splitting in Algorithm 4 results in the *Gauss–Seidel* iterative algorithm. When $\mathbf{A}$ is symmetric, $\mathbf{U} = \mathbf{L}^T$, and the Gauss–Seidel iteration can be written as

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{D}^{-1}\mathbf{L}^T\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}. \tag{4}$$

Just as we pointed out for the Gibbs sampler, variants of the Gauss–Seidel algorithm such as "red-black" coordinate updates (Saad [54]), may be written in this form using a suitable permutation matrix.

The component-wise form of the Gauss–Seidel algorithm can be written in "equation" form just as the Gibbs sampler (1) was in Algorithm 3. The component-wise form emphasizes that Gauss–Seidel can be implemented using vector and scalar operations only, and neither storage nor inversion of the splitting is required.

3.2. Convergence

A fundamental theorem of linear stationary iterative methods states that the splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$, where $\mathbf{M}$ is nonsingular, is convergent (i.e., $\mathbf{x}^{(k)} \to \mathbf{A}^{-1}\mathbf{b}$ for any $\mathbf{x}^{(0)}$) if and only if $\varrho(\mathbf{M}^{-1}\mathbf{N}) < 1$, where $\varrho(\cdot)$ denotes the spectral radius of a matrix (Young [64], Theorem 3.5.1). This characterization is often used as a definition (Axelsson [5], Golub and Van Loan [29], Saad [54]).

The *error* at step k is $\mathbf{e}^{(k+1)} = \mathbf{x}^{(k+1)} - \mathbf{x}^*$, where $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$. It follows that

$$\mathbf{e}^{(k+1)} = \left(\mathbf{M}^{-1}\mathbf{N}\right)^k \mathbf{e}^{(0)} \tag{5}$$

and hence the asymptotic average reduction in error per iteration is the multiplicative factor

$$\lim_{k\to\infty} \left(\frac{\|\mathbf{e}^{(k+1)}\|_2}{\|\mathbf{e}^{(0)}\|_2} \right)^{1/k} = \varrho\left(\mathbf{M}^{-1}\mathbf{N}\right) \tag{6}$$

(Axelsson [5], page 166). In numerical analysis, this is called the (asymptotic average) *convergence factor* (Axelsson [5], Saad [54]). Later, we will show that this is exactly the same as the quantity called the geometric *convergence rate* in the statistics literature (see, e.g., Robert and Casella [49]), for the equivalent Gibbs sampler. We will use the term "convergence factor" throughout this paper to avoid a clash of terminology, since in numerical analysis the *rate* of convergence is minus the log of the convergence factor (see, e.g., Axelsson [5], page 166).

Table 1. Common stationary linear solvers generated by splittings $\mathbf{A} = \mathbf{M} - \mathbf{N}$, and conditions that guarantee convergence when $\mathbf{A}$ is SPD

Splitting	$\mathbf{M}$	Convergence
Richardson (R)	$\frac{1}{\omega}\mathbf{I}$	$0 < \omega < \frac{2}{\varrho(\mathbf{A})}$
Jacobi (J)	$\mathbf{D}$	$\mathbf{A}$ strictly diagonally dominant
Gauss–Seidel (GS)	$\mathbf{D} + \mathbf{L}$	always
SOR	$\frac{1}{\omega}\mathbf{D} + \mathbf{L}$	$0 < \omega < 2$
SSOR	$\frac{\omega}{2-\omega}\mathbf{M}_{\text{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\text{SOR}}^T$	$0 < \omega < 2$

3.3. Common matrix splittings

We now present the matrix splittings corresponding to some common stationary linear iterative solvers, with details for the case where $\mathbf{A}$ is symmetric, as holds for precision or covariance matrices.

We have seen that the Gauss–Seidel iteration uses the splitting $\mathbf{M}_{\text{GS}} = \mathbf{L} + \mathbf{D}$ and $N_{\text{GS}} = -\mathbf{L}^T$. Gauss–Seidel is one of the simplest splittings and solvers, but is also quite slow. Other splittings have been developed, though the speed of each method is often problem specific. Some common splittings are shown in Table 1, listed with, roughly, greater speed downwards. Speed of convergence in a numerical example is presented later in Section 6.

The method of *successive over-relaxation* (SOR) uses the splitting

$$\mathbf{M}_{\text{SOR}} = \frac{1}{\omega}\mathbf{D} + \mathbf{L} \quad \text{and} \quad \mathbf{N}_{\text{SOR}} = \frac{1-\omega}{\omega}\mathbf{D} - \mathbf{L}^T \tag{7}$$

in which ω is a *relaxation* parameter chosen with $0 < \omega < 2$. SOR with $\omega = 1$ is Gauss–Seidel. For optimal values of ω such that $\varrho(\mathbf{M}_{\text{SOR}}^{-1}\mathbf{N}_{\text{SOR}}) < \varrho(\mathbf{M}_{\text{GS}}^{-1}\mathbf{N}_{\text{GS}})$, SOR is an accelerated Gauss–Seidel iteration. Unfortunately, there is no closed form for the optimal value of ω for an arbitrary matrix $\mathbf{A}$, and the interval of values of ω which admits accelerated convergence can be quite narrow (Young [64], Golub and Van Loan [29], Saad [54]).

The *symmetric-SOR* method (SSOR) incorporates both a forward and backward sweep of SOR so that if $\mathbf{A}$ is symmetric then the splitting is symmetric (Golub and Van Loan [29], Saad [54]),

$$\mathbf{M}_{\text{SSOR}} = \frac{\omega}{2-\omega}\mathbf{M}_{\text{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\text{SOR}}^T \quad \text{and} \quad \mathbf{N}_{\text{SSOR}} = \frac{\omega}{2-\omega}\mathbf{N}_{\text{SOR}}^T\mathbf{D}^{-1}\mathbf{N}_{\text{SOR}}. \tag{8}$$

We will make use of symmetric splittings in conjunction with polynomial acceleration in Section 5.

When the matrix $\mathbf{A}$ is dense, Gauss–Seidel and SOR cost about $3n^2$ flops per iteration, with $2n^2$ due to multiplication by the matrix $\mathbf{A}$ (in order to calculate the residual) and another n^2 for the forward substitution to solve $\mathbf{M}\mathbf{u} = \mathbf{r}$. Richardson incurs no cost to solve $\mathbf{M}\mathbf{u} = \mathbf{r}$, while a solve with the diagonal Jacobi matrix incurs n flops. Iterative methods are particularly attractive when the matrix $\mathbf{A}$ is sparse, since then the cost per iteration is only $O(n)$ flops.

Many theorems establish convergence of splittings by utilizing properties of $\mathbf{A}$ in specific applications. Some general conditions that guarantee convergence when $\mathbf{A}$ is SPD are given in the right column of Table 1 (Golub and Van Loan [29], Saad [54], Young [64]).

4. Equivalence of stationary linear solvers and Gibbs samplers

We first consider the equivalence between linear solvers and stochastic iterations in the case where the starting state and noise are not necessarily normally distributed, then in Section 4.2 et seq. we restrict consideration to normal distributions.

4.1. General noise

The striking similarity between the Gibbs sampler (1) and the Gauss–Seidel iteration (4) is no coincidence. It is an example of a general equivalence between the stationary linear solver derived from a splitting and the associated stochastic iteration used as a sampler. We will make explicit the equivalence in the following theorems and corollary. In the first theorem we show that a splitting is convergent (in the sense of stationary iterative solvers) if and only if the associated stochastic iteration is convergent in distribution.

Theorem 1. *Let* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *be a splitting with* $\mathbf{M}$ *invertible, and let* $\pi(\cdot)$ *be some fixed probability distribution with zero mean and fixed non-zero covariance. For any fixed vector* $\mathbf{b}$*, and random vectors* $\mathbf{c}^{(k)} \overset{i.i.d.}{\sim} \pi$, $k = 0, 1, 2, \ldots$, *the stationary linear iteration*

$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}\mathbf{N}\mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{b} \tag{9}$$

converges, with $\mathbf{x}^{(k)} \to \mathbf{A}^{-1}\mathbf{b}$ *as* $k \to \infty$ *whatever the initial vector* $\mathbf{x}^{(0)}$*, if and only if there exists a distribution* Π *such that the stochastic iteration*

$$\mathbf{y}^{(k+1)} = \mathbf{M}^{-1}\mathbf{N}\mathbf{y}^{(k)} + \mathbf{M}^{-1}\mathbf{c}^{(k)} \tag{10}$$

converges in distribution to Π*, with* $\mathbf{y}^{(k)} \xrightarrow{\mathcal{D}} \Pi$ *as* $k \to \infty$ *whatever the initial state* $\mathbf{y}^{(0)}$.

Proof. If the linear iteration (9) converges, then $\varrho(\mathbf{M}^{-1}\mathbf{N}) < 1$ (Theorems 3–5.1 in Young [64]). Hence, there exists a unique distribution Π with $\mathbf{y}^{(k+1)} \xrightarrow{\mathcal{D}} \Pi$ (Theorem 2.3.18–4 of Duflo [15]), which shows necessity. Conversely, if the linear solver does not converge to a limit independent of $\mathbf{x}^{(0)}$ for some $\mathbf{b}$, that also holds for $\mathbf{b} = \mathbf{0}$ and hence initializing the sampler with $\mathrm{E}[\mathbf{y}^{(0)}] = \mathbf{x}^{(0)}$ yields $\mathrm{E}[\mathbf{y}^{(k+1)}] = (\mathbf{M}^{-1}\mathbf{N})^k\mathbf{x}^{(0)}$ which does not converge to a value independent of $\mathbf{y}^{(0)}$. Sufficiency holds by the contrapositive. □

Convergence of the stochastic iteration (10) could also be established via the more general theory of Diaconis and Freedman [14] that allows the iteration operator $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$ to be random, with convergence in distribution guaranteed when $\mathbf{G}$ is *contracting on average*; see Diaconis and Freedman [14] for details.

The following theorem shows how to design the noise distribution π so that the limit distribution Π has a desired mean μ and covariance $\mathbf{\Sigma} = \mathbf{A}^{-1}$.

Theorem 2. *Let* $\mathbf{A}$ *be SPD,* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *be a convergent splitting,* $\boldsymbol{\mu}$ *a fixed vector, and* $\pi(\cdot)$ *a fixed probability distribution with finite mean* ν *and non-zero covariance* $\mathbf{V}$*. Consider the stochastic iteration* (10) *where* $\mathbf{c}^{(k)} \overset{i.i.d.}{\sim} \pi$, $k = 0, 1, 2, \ldots$. *Then, whatever the starting state* $\mathbf{y}^{(0)}$*, the following are equivalent*:

1. $\mathrm{E}[\mathbf{c}^{(k)}] = \boldsymbol{\nu}$ *and* $\mathrm{Var}(\mathbf{c}^{(k)}) = \mathbf{V} = \mathbf{M}^T + \mathbf{N}$;
2. *the iterates* $\mathbf{y}^{(k)}$ *converge in distribution to some distribution* Π *that has mean* $\boldsymbol{\mu} = \mathbf{A}^{-1}\nu$ *and covariance matrix* $\mathbf{A}^{-1}$; *in particular* $\mathrm{E}[\mathbf{y}^{(k)}] \to \boldsymbol{\mu}$ *and* $\mathrm{Var}(\mathbf{y}^{(k)}) \to \mathbf{A}^{-1}$ *as* $k \to \infty$.

Proof. Appendix A.1. □

Additionally, the mean and covariance converge geometrically, with convergence factors given by the convergence factors for the linear iterative method, as established in the following corollary.

Corollary 3. *The first and second moments of iterates in the stochastic iteration in Theorem 2 converge geometrically. Specifically,* $\mathrm{E}(\mathbf{y}^{(k)}) \to \boldsymbol{\mu}$ *with convergence factor* $\varrho(\mathbf{M}^{-1}\mathbf{N})$ *and* $\mathrm{Var}(\mathbf{y}^{(k)}) = \mathbf{A}^{-1} + \mathbf{G}^k(\mathrm{Var}(\mathbf{y}^{(0)}) - \mathbf{A}^{-1})(\mathbf{G}^k)^T \to \mathbf{A}^{-1}$ *with convergence factor* $\varrho(\mathbf{M}^{-1}\mathbf{N})^2$.

Proof. Appendix A.1. □

Note that the matrix splitting has allowed an explicit construction of the noise covariance to give a desired precision matrix of the target distribution. We see from Theorem 2 that the stochastic iteration may be designed to converge to a distribution with non-zero target mean, essentially by adding the deterministic iteration (9) to the stochastic iteration (10). This is particularly useful when the mean is defined implicitly via solving a matrix equation. In cases where the mean is known explicitly, the mean may be added after convergence of the stochastic iteration with zero mean, giving an algorithm with faster convergence since the covariance matrix converges with factor $\varrho(\mathbf{M}^{-1}\mathbf{N})^2 < \varrho(\mathbf{M}^{-1}\mathbf{N})$ (this was also noted by Barone *et al.* [9]). Convergence in variance for non-normal targets was considered in Fox and Parker [19].

Using Theorems 1 and 2, and Corollary 3 we can draw on the vast literature in numerical linear algebra on stationary linear iterative methods to find random iterations that are computationally efficient and provably convergent in distribution with desired mean and covariance. In particular, results in Amit and Grenander [4], Barone and Frigessi [8], Galli and Gao [22], Roberts and Sahu [51], and Liu *et al.* [39] are special cases of the general theory of matrix splittings presented here.

4.2. Sampling from normal distributions using matrix splittings

We now restrict attention to the case of normal target distributions.

Table 2. Some generalized Gibbs samplers for drawing from $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ adapted from common stationary linear solvers. Each Gibbs iteration requires sampling the noise vector $\mathbf{c}^{(k)} \sim \mathrm{N}(\mathbf{0}, \mathbf{M}^T + \mathbf{N})$

Sampler	$\mathbf{M}$	$\mathrm{Var}(\mathbf{c}^{(k)}) = \mathbf{M}^T + \mathbf{N}$
Richardson	$\frac{1}{\omega}\mathbf{I}$	$\frac{2}{\omega}\mathbf{I} - \mathbf{A}$
Jacobi	$\mathbf{D}$	$2\mathbf{D} - \mathbf{A}$
Gibbs (Gauss–Seidel)	$\mathbf{D} + \mathbf{L}$	$\mathbf{D}$
SOR	$\frac{1}{\omega}\mathbf{D} + \mathbf{L}$	$\frac{2-\omega}{\omega}\mathbf{D}$
SSOR (REGS)	$\frac{\omega}{2-\omega}\mathbf{M}_{\mathrm{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\mathrm{SOR}}^T$	$\frac{\omega}{2-\omega}(\mathbf{M}_{\mathrm{SOR}}\mathbf{D}^{-1}\mathbf{M}_{\mathrm{SOR}}^T + \mathbf{N}_{\mathrm{SOR}}^T\mathbf{D}^{-1}\mathbf{N}_{\mathrm{SOR}})$

Corollary 4. *If in Theorem 2 we set* $\pi = \mathrm{N}(\boldsymbol{\nu}, \mathbf{V})$, *for some non-zero covariance matrix* $\mathbf{V}$, *then, whatever the starting state* $\mathbf{y}^{(0)}$, *the following are equivalent*: (i) $\mathbf{V} = \mathbf{M}^T + \mathbf{N}$; (ii) $\mathbf{y}^{(k)} \xrightarrow{\mathcal{D}} \mathrm{N}(\boldsymbol{\mu}, \mathbf{A}^{-1})$ *where* $\boldsymbol{\mu} = \mathbf{A}^{-1}\nu$.

Proof. Since π is normal, then Π in Theorem 2 is normal. Since a normal distribution is sufficiently described by its first two moments, the corollary follows. □

Using Corollary 4, we found normal sampling algorithms corresponding to some common stationary linear solvers. The results are given in Table 2. A sampler corresponding to a convergent splitting is implemented in Algorithm 5.

Algorithm 5: Stationary sampler of $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$

input : SPD precision matrix $\mathbf{A}$, $\mathbf{M}$ and $\mathbf{N}$ defining a convergent splitting of $\mathbf{A}$, number of steps $k_{\max}$, initial state $\mathbf{y}^{(0)}$
output: $\mathbf{y}^{(k)}$ approximately distributed as $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$

for $k = 0, \ldots, k_{\max}$ **do**
  sample $\mathbf{c}^{(k)} \sim \mathrm{N}(\mathbf{0}, \mathbf{M}^T + \mathbf{N})$;
  $\mathbf{y}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{N}\mathbf{y}^{(k)} + \mathbf{c}^{(k)})$
end

The assignment $\mathbf{y}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{N}\mathbf{y}^{(k)} + \mathbf{c}^{(k)})$ in Algorithm 5 can be replaced by the slightly more expensive steps $\mathbf{r}^{(k)} = \mathbf{c}^{(k)} - \mathbf{A}\mathbf{y}^{(k)}$ and $\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \mathbf{M}^{-1}\mathbf{r}^{(k)}$, which allows monitoring of the residual, and emphasizes the equivalence with the stationary linear solver in Algorithm 4. Even though convergence may not be diagnosed by a decreasing norm of the residual, lack of convergence can be diagnosed when the residual diverges in magnitude. In practice, the effective convergence factor for a sampler may be calculated by solving the linear system (2) (perhaps with a random right-hand side) using the iterative solver derived from the splitting and monitoring the decrease in error to evaluate the asymptotic average convergence factor using equation (6). By Corollary 3, this estimates the convergence factor for the sampler.

The practicality of a sampler derived from a convergent splitting depends on the ease with which one can solve $\mathbf{My} = \mathbf{r}$ for any $\mathbf{r}$ (as for the stationary linear solver) and also the ease of drawing i.i.d. noise vectors from $\mathrm{N}(\mathbf{0}, \mathbf{M}^T + \mathbf{N})$. Sampling the noise vector is simple when a matrix square root, such as the Cholesky factorization, of $\mathbf{M}^T + \mathbf{N}$ is cheaply available. Thus, a sampler is at least as expensive as the corresponding linear solver since, in addition to operations in each iteration, the sampler must factor the $n \times n$ matrix $\mathbf{Var}(\mathbf{c}^{(k)}) = \mathbf{M}^T + \mathbf{N}$. For the samplers listed in Table 2 it is interesting that the simpler the splitting, the more complicated is the variance of the noise. Neither Richardson nor Jacobi splittings give useful sampling algorithms since the difficulty of sampling the noise vector is no less than the original task of sampling from $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$. The Gauss–Seidel splitting, giving the usual Gibbs sampler, is at a kind of sweet spot, where solving equations in $\mathbf{M}$ is simple while the required noise variance is diagonal, so posing a simple sampling problem.

The SOR stationary sampler uses the SOR splitting $\mathbf{M}_{\mathrm{SOR}}$ and $\mathbf{N}_{\mathrm{SOR}}$ in (7) for $0 < \omega < 2$ and the noise vector $\mathbf{c}^{(k)} \sim \mathrm{N}(\mathbf{0}, \mathbf{M}_{\mathrm{SOR}}^T + \mathbf{N}_{\mathrm{SOR}} = \frac{2-\omega}{\omega}\mathbf{D})$ (Table 2). This sampler was introduced by Adler [1], rediscovered by Barone and Frigessi [8], and has been studied extensively (Barone *et al.* [9], Galli and Gao [22], Liu *et al.* [39], Neal [43], Roberts and Sahu [51]). For $\omega = 1$, the SOR sampler is a Gibbs (Gauss–Seidel) sampler. For values of ω such that $\varrho(\mathbf{M}_{\mathrm{SOR}}^{-1}\mathbf{N}_{\mathrm{SOR}}) < \varrho(\mathbf{M}_{\mathrm{GS}}^{-1}\mathbf{N}_{\mathrm{GS}}))$, the SOR-sampler is an accelerated Gibbs sampler. As for the linear solver, implementation of the Gibbs and SOR samplers by Algorithm 5 requires multiplication by the upper triangular $\mathbf{N}$ and forward substitution with respect to $\mathbf{M}$ at a cost of $2n^2$ flops. In addition, these samplers must take the square root of the diagonal matrix $\frac{2-\omega}{\omega}\mathbf{D}$ at a mere cost of $O(n)$ flops.

Implementation of an SSOR sampler instead of a Gibbs or SOR sampler is advantageous since the Markov chain $\{\mathbf{y}^{(k)}\}$ is reversible (Roberts and Sahu [51]). SSOR sampling uses the symmetric-SOR splitting $\mathbf{M}_{\mathrm{SSOR}}$ and $\mathbf{N}_{\mathrm{SSOR}}$ in (8). The SSOR stationary sampler is most easily implemented by forward and backward SOR sampling sweeps as in Algorithm 6, so the matrices $\mathbf{M}_{\mathrm{SSOR}}$ and $\mathbf{N}_{\mathrm{SSOR}}$ need never be explicitly formed.

Algorithm 6: SSOR sampling from $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$

input : The SOR splitting $\mathbf{M}, \mathbf{N}$ of $\mathbf{A}$, relaxation parameter ω, initial state $\mathbf{y}$, $k_{\max}$
output: $\mathbf{y}$ approximately distributed as $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$

set $\gamma = (\frac{2}{\omega} - 1)^{1/2}$;

for $k = 1, \ldots, k_{\max}$ **do**
  sample $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$;
  $\mathbf{x} := \mathbf{M}^{-1}(\mathbf{Ny} + \gamma \mathbf{D}^{1/2}\mathbf{z})$;
  sample $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$;
  $\mathbf{y} := \mathbf{M}^{-T}(\mathbf{N}^T\mathbf{x} + \gamma \mathbf{D}^{1/2}\mathbf{z})$
end

We first encountered restricted versions of Corollary 4 for normal distributions in Amit and Grenander [4] and in Barone and Frigessi [8] where geometric convergence of the covariance

matrices was established for the Gauss–Seidel and SOR splittings. These and the SSOR splitting were investigated in Roberts and Sahu [51] (who labelled the sampler REGS).

Corollary 4 and Table 1 show that the Gibbs, SOR and SSOR samplers converge for any SPD precision matrix $\mathbf{A}$. This summarizes results in Barone and Frigessi [8], Galli and Gao [22] and the deterministic sweeps investigated in Amit and Grenander [4], Roberts and Sahu [51], Liu *et al.* [39]. Corollary 4 generalizes these results for *any* matrix splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ by guaranteeing convergence of the random iterates (10) to $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ with convergence factor $\varrho(\mathbf{M}^{-1}\mathbf{N})$ (or $\varrho(\mathbf{M}^{-1}\mathbf{N})^2$ if $\mu = \mathbf{0}$).

5. Non-stationary iterative methods

5.1. Acceleration of linear solvers by polynomials

A common scheme in numerical linear algebra for accelerating a stationary method when $\mathbf{M}$ and $\mathbf{N}$ are symmetric is through the use of polynomial preconditioners (Axelsson [5], Golub and Van Loan [29], Saad [54]). Equation (5) shows that after k steps the error in the stationary method is a kth order polynomial of the matrix $\mathbf{I} - \mathbf{G} = \mathbf{M}^{-1}\mathbf{A}$. The idea behind polynomial acceleration is to implicitly implement a different kth order polynomial $P_k(\mathbf{M}^{-1}\mathbf{A})$ such that $\varrho(P_k(\mathbf{M}^{-1}\mathbf{A})) < \varrho((\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})^k)$. The coefficients of $P_k(\mathbf{M}^{-1}\mathbf{A})$ are functions of a set of *acceleration parameters* $\{\{\alpha_k\}, \{\tau_k\}\}$, introduced by the second order iteration

$$\mathbf{x}^{(k+1)} = (1 - \alpha_k)\mathbf{x}^{(k-1)} + \alpha_k \mathbf{x}^{(k)} + \alpha_k \tau_k \mathbf{M}^{-1}\big(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}\big). \tag{11}$$

At the first step, $\alpha_0 = 1$ and $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \tau_0 \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(0)})$. Setting $\alpha_k = \tau_k = 1$ for every k yields a basic un-accelerated stationary method. The accelerated iteration in (11) is implemented at a negligible increase in cost of $\mathcal{O}(n)$ flops per iteration (due to scalar-vector multiplication and vector addition) over the corresponding stationary solver (3).

It can be shown (e.g., Axelsson [5]) that the $(k+1)$st order polynomial P_{k+1} generated recursively by the second order non-stationary linear solver (11) is

$$P_{k+1}(\lambda) = (\alpha_k - \alpha_k \tau_k \lambda) P_k(\lambda) + (1 - \alpha_k) P_{k-1}(\lambda). \tag{12}$$

Hence, the k-step error $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{A}^{-1}\mathbf{b}$ may be written as $\mathbf{e}^{(k+1)} = P_k(\mathbf{M}^{-1}\mathbf{A})\mathbf{e}^{(0)}$, which can be compared directly to (5).

When estimates of the extreme eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$ of $\mathbf{I} - \mathbf{G} = \mathbf{M}^{-1}\mathbf{A}$ are available ($\lambda_{\min}$ and $\lambda_{\max}$ are real when $\mathbf{M}$ and $\mathbf{N}$ are symmetric), then the coefficients $\{\tau_k, \alpha_k\}$ can be chosen to generate the *scaled Chebyshev* polynomials $\{Q_k\}$, which give optimal error reduction at every step. The Chebyshev acceleration parameters are

$$\tau_k = \frac{2}{\lambda_{\max} + \lambda_{\min}}, \qquad \beta_k = \left(\frac{1}{\tau_k} - \beta_{k-1}\left(\frac{\lambda_{\max} - \lambda_{\min}}{4}\right)^2\right)^{-1}, \qquad \alpha_k = \frac{\beta_k}{\tau_k}, \tag{13}$$

where $\alpha_0 = 1$ and $\beta_0 = \tau_0$ (Axelsson [5]). Note that these parameters are independent of the iterates $\{\mathbf{x}^{(k)}\}$. Since $\mathbf{M}$ is required to be symmetric, applying Chebyshev acceleration to SSOR is a common pairing; its effectiveness as a linear solver is shown later in Table 3.

Whereas the stationary methods converge with asymptotic average convergence factor $\varrho(\mathbf{M}^{-1}\mathbf{N})$, the convergence factor for the Chebyshev accelerated method depends on $\operatorname{cond}(\mathbf{M}^{-1}\mathbf{A}) = \lambda_{\max}/\lambda_{\min}$. Specifically the scaled Chebyshev polynomial $Q_k(\boldsymbol{\lambda})$ minimizes $\max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} P_k(\lambda)$ over all kth order polynomials P_k, with

$$\max_{\lambda\in[\lambda_{\min},\lambda_{\max}]} \bigl|Q_k(\lambda)\bigr| = \frac{2\sigma^k}{1+\sigma^{2k}}. \tag{14}$$

Since the error at the kth step of a Chebyshev accelerated linear solver is $\mathbf{e}^{(k+1)} = Q_k(\lambda)\mathbf{e}^{(0)}$, then the asymptotic convergence factor is bounded above by

$$\sigma = \frac{1-\sqrt{\lambda_{\min}/\lambda_{\max}}}{1+\sqrt{\lambda_{\min}/\lambda_{\max}}} \tag{15}$$

(Axelsson [5], page 181). Since $\sigma \in [0, 1)$, the polynomial accelerated scheme is guaranteed to converge even if the original splitting was not convergent. Further, the convergence factor of the stationary iterative solver is bounded below by $\rho = \frac{1-\lambda_{\min}/\lambda_{\max}}{1+\lambda_{\min}/\lambda_{\max}}$ (see, e.g., Axelsson [5], Theorem 5.9). Since $\sigma < \rho$ (except when $\lambda_{\min} = \lambda_{\max}$ in which case $\sigma = 0$), polynomial acceleration always reduces the convergence factor, so justifies the term *acceleration*. The Chebyshev accelerated iteration (11) is amenable to preconditioning that reduces the condition number, and hence reduces σ, such as incomplete Cholesky factorization or graphical methods (Axelsson [5], Saad [54]). Axelsson also shows that after

$$k^* = \left\lceil \frac{\ln(\varepsilon/2)}{\ln\sigma} \right\rceil \tag{16}$$

iterations of the Chebyshev solver, the error reduction is $\|\mathbf{e}^{(k^*)}\|_{\mathbf{A}^\nu}/\|\mathbf{e}^{(0)}\|_{\mathbf{A}^\nu} \le \varepsilon$ for some real number ν and any $0 < \varepsilon < 1$ (Axelsson [5], equation (5.32)).

5.2. Acceleration of Gibbs sampling by polynomials

Any acceleration scheme devised for a stationary linear solver is a candidate for accelerating convergence of a Gibbs sampler. For example, consider the second order stochastic iteration

$$\mathbf{y}^{(k+1)} = (1-\alpha_k)\mathbf{y}^{(k-1)} + \alpha_k\mathbf{y}^{(k)} + \alpha_k\tau_k\mathbf{M}^{-1}\bigl(\mathbf{c}^{(k)} - \mathbf{A}\mathbf{y}^{(k)}\bigr) \tag{17}$$

analogous to the linear solver in (11) but now the vector $\mathbf{b}$ has been replaced by a random vector $\mathbf{c}^{(k)}$. The equivalence between polynomial accelerated linear solvers and polynomial accelerated samplers is made clear in the next three theorems.

Theorem 5. *Let* $\mathbf{A}$ *be SPD and* $\mathbf{A} = \mathbf{M} - \mathbf{N}$ *be a symmetric splitting. Consider a set of independent noise vectors* $\{\mathbf{c}^{(k)}\}$ *with moments*

$$\mathrm{E}\bigl(\mathbf{c}^{(k)}\bigr) = \boldsymbol{\nu} \quad \text{and} \quad \mathrm{Var}\bigl(\mathbf{c}^{(k)}\bigr) = a_k\mathbf{M} + b_k\mathbf{N}$$

such that $a_k := \frac{2-\tau_k}{\tau_k} + (b_k - 1)(\frac{1}{\tau_k} + \frac{1}{\kappa_k} - 1)$, $b_k := \frac{2(1-\alpha_k)}{\alpha_k}(\frac{\kappa_k}{\tau_k}) + 1$, $\kappa_{k+1} := \alpha_k\tau_k + (1-\alpha_k)\kappa_k$, *and* $\kappa_1 = \tau_0$. *If the polynomial accelerated linear solver* (11) *converges to* $\mathbf{A}^{-1}\mathbf{b}$ *with a set of parameters* $\{\{\alpha_k\}, \{\tau_k\}\}$ *that are independent of* $\{\mathbf{x}^{(k)}\}$, *then the polynomial accelerated stochastic iteration* (17) *converges in distribution to a distribution* Π *with mean* $\mathbf{A}^{-1}\boldsymbol{\nu}$ *and covariance matrix* $\mathbf{A}^{-1}$. *Furthermore, if the* $\{\mathbf{c}^{(k)}\}$ *are normal, then*

$$\mathbf{y}^{(k)} \xrightarrow{\mathcal{D}} \mathrm{N}\big(\boldsymbol{\mu} = \mathbf{A}^{-1}\boldsymbol{\nu}, \mathbf{A}^{-1}\big).$$

Proof. Appendix A.2. □

Given a second order linear solver (11) that converges, Theorem 5 makes clear how to construct a second order sampler (17) that is guaranteed to converge. The next corollary shows that the polynomial P_k that acts on the linear solver error $\mathbf{x}^{(k)} - \mathbf{A}^{-1}\mathbf{b}$ is the same polynomial that acts on the errors in the first and second moments of the sampler, $\mathrm{E}(\mathbf{y}^{(k)}) - \mathbf{A}^{-1}\boldsymbol{\nu}$ and $\mathrm{Var}(\mathbf{y}^{(k)}) - \mathbf{A}^{-1}$ respectively. In other words, the convergence factors for a polynomial accelerated solver and sampler are the same.

Corollary 6. *Suppose that the polynomial accelerated linear solver* (11) *converges with asymptotic convergence factor* $\sigma = (\lim_{k\to\infty} \max_\lambda |P_k(\lambda)|)^{1/k}$, *where* P_k *is the kth order polynomial recursively generated by* (12). *Then under the conditions of Theorem* 5,

$$\mathrm{E}\big(\mathbf{y}^{(k)}\big) - \mathbf{A}^{-1}\boldsymbol{\nu} = P_k\big(\mathbf{M}^{-1}\mathbf{A}\big)\big(\mathrm{E}\big(\mathbf{y}^{(0)}\big) - \mathbf{A}^{-1}\boldsymbol{\nu}\big) \to \mathbf{0}$$

with asymptotic convergence factor σ, *and*

$$\mathrm{Var}\big(\mathbf{y}^{(k)}\big) - \mathbf{A}^{-1} = P_k\big(\mathbf{M}^{-1}\mathbf{A}\big)\big(\mathbf{Var}\big(y^{(0)} - \mathbf{A}^{-1}\big)\big)\big(P_k\big(\mathbf{M}^{-1}\mathbf{A}\big)\big)^T \to \mathbf{0}$$

with asymptotic convergence factor σ^2.

Proof. Appendix A.2. □

While Corollary 6 treats convergence in L^2-norm, or, equivalently, the spectral radius of the error in covariance matrices, standard results in matrix analysis such as Corollary 5.6.13 in Horn and Johnson [36] then imply that convergence in covariance matrices also occurs elementwise, and with the same convergence factor.

Corollary 6 allows a direct comparison of the convergence factor for a polynomial accelerated sampler (σ, or σ^2 if $\mu = \mathbf{0}$) to the convergence factor given previously for the corresponding un-accelerated stationary sampler ($\varrho(\mathbf{M}^{-1}\mathbf{N})$, or $\varrho(\mathbf{M}^{-1}\mathbf{N})^2$ if $\mu = \mathbf{0}$). In particular, given a second order linear solver with accelerated convergence compared to the corresponding stationary iteration, the corollary guarantees that the second order Gibbs sampler (17) will converge faster than the stationary Gibbs sampler (10).

The explicit forms for error in mean and covariance given in Corollary 6 may be used to give explicit forms for distributional convergence of k-step distributions to the target distribution. An example is the following:

Corollary 7. *Consider the sampler in Corollary 6, initialized at* $\mathbf{y}^{(0)} \sim \pi^{(0)} = \mathrm{N}(\boldsymbol{\mu}^{(0)}, \boldsymbol{\Sigma}^{(0)})$, *with possibly* $\boldsymbol{\Sigma}^{(0)} = \mathbf{0}$, *and targeting* $\pi = \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. *Denote the k-step distribution* $\pi^{(k)} = \mathrm{N}(\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)})$, *that is,* $\mathrm{E}(\mathbf{y}^{(k)}) = \boldsymbol{\mu}^{(k)}$ *and* $\mathrm{Var}(\mathbf{y}^{(k)}) = \boldsymbol{\Sigma}^{(k)}$. *Then the* χ^2*-divergence of* π *from* $\pi^{(k)}$

$$\chi^2\big(\pi^{(k)} \| \pi\big) = \int \frac{[\pi^{(k)}(\mathbf{y}) - \pi(\mathbf{y})]^2}{\pi(\mathbf{y})}\, d\mathbf{y}$$

converges to 0 *as* $k \to \infty$ *with asymptotic average convergence factor* σ^2.

Proof.

$$\chi^2\big(\pi^{(k)} \| \pi\big) = |\mathbf{W}|^{-1/2} |2\mathbf{I} - \mathbf{W}|^{-1/2} \exp(S) - 1 \tag{18}$$

in which $|\cdot|$ denotes determinant, $\mathbf{W} = \boldsymbol{\Sigma}^{(k)} \boldsymbol{\Sigma}^{-1} = \mathbf{I} + P_k(\boldsymbol{\Sigma}^{(0)} - \boldsymbol{\Sigma}) P_k^T \boldsymbol{\Sigma}^{-1} \to \mathbf{I}$, and $S = (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T (2\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^{(k)})^{-1} (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}) = (\boldsymbol{\mu}^{(0)} - \boldsymbol{\mu})^T P_k \boldsymbol{\Sigma}^{-1} (2\mathbf{I} - \mathbf{W})^{-1} P_k^T (\boldsymbol{\mu}^{(0)} - \boldsymbol{\mu}) \to 0$, both with asymptotic convergence factor σ^2. We have omitted the argument of the error polynomial P_k for brevity. The result follows.

Corollary 7 may be used to show that expectations of square π-integrable functions also converge with asymptotic convergence factor σ^2, for any $\mathbf{y}^{(0)}$ and in π-expectation over $\mathbf{y}^{(0)}$, establishing distributional convergence with the same asymptotic convergence factor; see Appendix A of Roberts and Sahu [51] for the latter calculation in the setting of stationary iterations.[3] □

Just as Chebyshev polynomials are guaranteed to accelerate linear solvers, Corollaries 6 and 7 assure that Chebyshev polynomials can also accelerate a Gibbs sampler. Using Theorem 5, we derived the Chebyshev accelerated SSOR sampler (Fox and Parker [19]) by iteratively updating parameters via (13) and then generating a sampler via (17). Explicit implementation details of the Chebyshev accelerated sampler are provided in the supplementary material [20]. The polynomial accelerated sampler is implemented at a negligible increase in cost of $\mathcal{O}(n)$ flops per iteration over the cost ($4n^2$ flops) of the SSOR sampler (Algorithm 6). The asymptotic convergence factor is given by the next Corollary, which follows from Corollary 6 and equation (15).

Corollary 8. *If the Chebyshev accelerated linear solver converges, then the mean* $\mathrm{E}(\mathbf{y}^{(k)})$ *of the corresponding Chebyshev accelerated stochastic iteration* (17) *converges to* $\boldsymbol{\mu} = \mathbf{A}^{-1}\boldsymbol{\nu}$ *with asymptotic convergence factor* $(\frac{1-\sqrt{\lambda_{\min}/\lambda_{\max}}}{1+\sqrt{\lambda_{\min}/\lambda_{\max}}})$ *and the covariance matrix* $\mathrm{Var}(\mathbf{y}^{(k)})$ *converges to* $\mathbf{A}^{-1}$ *with asymptotic convergence factor* $(\frac{1-\sqrt{\lambda_{\min}/\lambda_{\max}}}{1+\sqrt{\lambda_{\min}/\lambda_{\max}}})^2$.

Corollary 8 and (14) show that a Chebyshev accelerated normal sampler is guaranteed to converge faster than any other acceleration scheme that has the parameters $\{\{\tau_k, \alpha_k\}\}$ independent of the iterates $\{\mathbf{y}^{(k)}\}$. This result shows that the preconditioning ideas presented in Section 5.1 to reduce $\mathrm{cond}(\mathbf{M}^{-1}\mathbf{A}) = \lambda_{\max}/\lambda_{\min}$ can also be used to speed up Chebyshev accelerated samplers. We do not investigate such preconditioning here.

[3] Note that (18) differs from equation (15) in Roberts and Sahu [51] that appears to be incorrect.

Corollary 8 and equation (16) suggest that, for any $\varepsilon > 0$, after k^* iterations the Chebyshev error reduction for the mean is smaller than ε. But even sooner, after $k^{**} = k^*/2$ iterations, the Chebyshev error reduction for the variance is predicted to be smaller than ε (Fox and Parker [19]).

6. Computed examples

The iterative sampling algorithms we have investigated are designed for problems where *operating* by the precision matrix is cheap. A common such case is when the precision matrix is sparse, as occurs when modeling a GMRF with a local neighbourhood structure. Then, typically, the precision matrix has $\mathcal{O}(n)$ non-zero elements, so direct matrix-vector multiplication has $\mathcal{O}(n)$ cost. We give two examples of sampling using sparse precision matrices: first, we present a small $n = 100$ example where complete diagnostics can be computed for evaluating the quality of convergence; and second, we present a $n = 10^6$ Bayesian linear inverse problem that demonstrates computational feasibility for large problems. The samplers are initialized with $\mathbf{y}^{(0)} = \mathbf{0}$ in both examples.

6.1. A 10 × 10 lattice example ($n = 100$)

A *first order locally linear* sparse precision matrix $\mathbf{A}$, considered by Higdon [35], Rue and Held [53], is

$$[\mathbf{A}]_{ij} = 10^{-4}\delta_{ij} + \begin{cases} n_i, & \text{if } i = j, \\ -1, & \text{if } i \neq j \text{ and } \|s_i - s_j\|_2 \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The discrete points $\{s_i\}$ are on a regular 10×10 lattice ($n = 100$) over the two dimensional domain $\mathcal{S} = [1, 10] \times [1, 10]$. The scalar n_i is the number of points neighbouring s_i, i.e., with distance 1 from s_i. The resulting matrix $\mathbf{A}$ is 100×100, $\|\mathbf{A}\|_2 = 7.8$ and $\|\boldsymbol{\Sigma} = \mathbf{A}^{-1}\|_2 = 10^4$. The sparsity of $\mathbf{A}$ is shown in the left panel of Figure 2. Although $n^2 = 10^4$, the number of non-zero elements of $\mathbf{A}$ is $\mathcal{O}(n)$ (460 in this example), hence each iteration of an iterative method costs $\mathcal{O}(n)$ flops. Since the bandwidth of $\mathbf{A}$ is $\mathcal{O}(n^{1/2})$ ($b = 11$ in this example), a Cholesky factorization costs $\mathcal{O}(b^2 n) = \mathcal{O}(n^2)$ flops.

To provide a comparison between linear solvers and samplers, we solved the system $\mathbf{Ax} = \mathbf{b}$ using linear solvers with different matrix splittings (Table 1), where $\mathbf{b}$ is fixed and non-zero, all initialized with $\mathbf{x}^{(0)} = \mathbf{0}$. The results are given in Table 3. The Richardson method does not converge (DNC) since the spectral radius of the iteration operator is greater than 1. The SOR iteration was run at the optimal relaxation parameter value of $\omega = 1.9852$. SSOR was run at its optimal value of $\omega = 1.6641$. Chebyshev accelerated SSOR (Cheby-SSOR), CG accelerated SSOR (CG-SSOR) (both run with $\omega = 1.6641$) and CG utilize a different implicit operator for each iteration, and so the spectral radius given in these cases is the geometric mean spectral radius of these operators (estimated using (5)). Even for this small example, Chebyshev acceleration

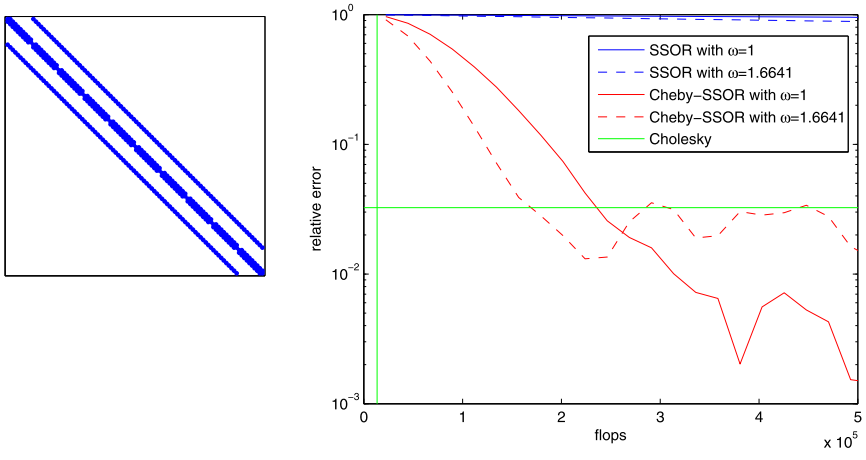

Figure 2. Left panel: Location of non-zero elements in the 100×100 precision matrix $\mathbf{A}$. Right panel: Relative error in covariance $\|\mathbf{A}^{-1} - \mathbf{S}_y^{(k)}\|_2 / \|\mathbf{A}^{-1}\|_2$ versus number of floating point operations (flops) for a sampler implemented with SSOR and $\omega = 1$, SSOR with optimal relaxation $\omega = 1.6641$, and SSOR with Chebyshev acceleration. When generating a single sample using Cholesky factoring, the green horizontal line indicates the fixed relative error in covariance, and the green vertical line indicates the fixed cost.

reduces the computational effort required for convergence by about two orders of magnitude, while CG acceleration reduces work by nearly two more orders of magnitude.

We investigated the following Gibbs samplers: SOR, SSOR, and the Chebyshev accelerated SSOR. These samplers are guaranteed to converge since the corresponding solver converges

Table 3. The number of iterations and the total number of floating point operations performed by some common stationary and accelerated linear solvers, and the Cholesky factorization, used to solve $\mathbf{Ax} = \mathbf{b}$ for fixed non-zero $\mathbf{b}$. Each solver was run until the residual became sufficiently small, $\|\mathbf{b} - \mathbf{Ax}^{(k)}\|_2 < 10^{-8}$. Details in Section 6

Solver	ω	$\varrho(\mathbf{M}^{-1}\mathbf{N})$	Number of iterations	Flops
Richardson	1	6.8	DNC	–
Jacobi	–	0.999972	4.01×10^5	5.69×10^7
Gauss–Seidel	–	0.999944	2.44×10^5	4.34×10^8
SSOR	1.6641	0.999724	6.7×10^4	2.39×10^8
SOR	1.9852	0.985210	1655	2.95×10^6
Cheby-SSOR	1	0.9786	958	3.41×10^6
Cheby-SSOR	1.6641	0.9673	622	2.21×10^6
CG	–	0.6375	48	9.22×10^4
CG-SSOR	1.6641	0.4471	29	6.66×10^4
Cholesky	–	–	–	1.35×10^4

(Theorems 1 and 5). Since the convergence factor for a sampler is equal to the convergence factor for the corresponding solver (Corollaries 3 and 6) then Gibbs samplers implemented with any of the matrix splittings in Table 1 exhibit the same convergence behavior as shown for the linear solvers in Table 3.

To numerically assess sampler convergence, the empirical sample covariance $\mathbf{S}_y^{(k)} \approx \mathbf{Var}(\mathbf{y}^{(k)})$ was calculated for each iteration k using 10^4 chains of samples. Convergence of $\mathbf{Var}(\mathbf{y}^{(k)}) \to \mathbf{A}^{-1}$ is depicted in the right panel of Figure 2 in terms of the relative error in covariance, $\|\mathbf{A}^{-1} - \mathbf{S}_y^{(k)}\|_2 / \|\mathbf{A}^{-1}\|_2$, as a function of the flop count. Each sampler iteration costs about 2.24×10^3 flops, so the figure shows the results from just over 220 sampler iterations. Since the sample means were uniformly close to zero, error in the mean is not shown.

The benchmark for evaluation of convergence of the iterative samplers in finite precision is the Cholesky factorization. The cost to generate a single sample by Cholesky factoring is about 1.34×10^4 flops. This cost is depicted by the green vertical line in the right panel of Figure 2. The relative error in covariance for a Cholesky sample, estimated empirically using 10^4 chains as for the iterative samplers, is depicted as the green horizontal line in the right panel of Figure 2. The figure shows that the Cholesky sample covariance $\mathbf{S}_y^{(k)}$ does not converge precisely to $\mathbf{A}^{-1}$ due to both the variability of the empirical sample covariance (calculated from 10^4 chains), and the effect of finite precision. To briefly explain the latter, when calculated in finite precision, the Cholesky factorization is $\mathbf{A} = \mathbf{C}\mathbf{C}^T + \mathbf{E}$ where $\mathbf{E}$ is the finite precision error with $\|\mathbf{E}\|_2 < \frac{2n^{3/2}\varepsilon}{1-2n^{3/2}\varepsilon}\|\mathbf{A}\|_2 + \mathcal{O}(\varepsilon^2)$ and ε is machine precision (Watkins [62]). Figure 2 shows that the iterative sample covariances become more precise than Cholesky with more computing time (indicated by the relative error in covariance for the iterative samplers eventually falling below the green horizontal line). Hence, for this example, the iterative samplers produce better samples than a Cholesky sampler.

The slow geometric convergence of the unaccelerated SSOR (REGS) samples $\mathbf{y}^{(k)}$ to $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ is clear in Figure 2. Even after 5×10^5 flops ($k = 220$ iterations), the sampler is not even close to convergence. This is not surprising given the large number of iterations necessary for the same stationary method to converge to a solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ (see Table 3). The accelerated convergence of the Chebyshev samplers, suggested by the faster convergence of the corresponding linear solvers in Table 3, is also evident in Figure 2, with convergence after only 1.70×10^5 flops (76 iterations) for the Chebyshev accelerated SSOR sampler with optimal relaxation parameter $\omega = 1.6641$, and the somewhat slower convergence at 2.37×10^5 flops (106 iterations) when $\omega = 1$.

We monitored convergence to the normal target distribution $\mathrm{N}(\mathbf{0}, \mathbf{A}^{-1})$ in this example by showing convergence in L^2-norm of the sample covariance to $\mathbf{A}^{-1}$, $\|\mathbf{A}^{-1} - \mathbf{S}_y^{(k)}\|_2$. Corollary 6 shows that L^2 convergence of covariance occurs with k-step factor bounded by $\|P_k(\mathbf{M}^{-1}\mathbf{N})\|^2$. As shown in Corollary 7, this same term bounds the asymptotic rate of distributional convergence. Hence, the plots of empirical L^2 convergence of covariance in Figure 2 may also be interpreted as numerical estimates for the distributional convergence of the k-step distributions to the target distribution.

6.2. A 100 × 100 × 100 ($n = 10^6$) linear inverse problem in biofilm imaging

We now perform accelerated sampling from a GMRF in 3-dimensions, as a stylized example of estimating a voxel image of a biofilm from confocal scanning laser microscope (CSLM) data (Lewandowski and Beyenal [37]). This large example illustrates the feasibility of Chebyshev accelerated sampling in large problems for which sampling by Cholesky factorization of the precision matrix is too computationally and memory intensive to be performed on a standard desktop computer.

We consider the problem of reconstructing a $100 \times 100 \times 100$ voxel image $\mathbf{x}$ of a bacterial biofilm, that is, a community of bacteria aggregated together as *slime*, given a subsampled $100 \times 100 \times 10$ CSLM data set $\mathbf{y}$. For this exercise, we synthesized a "true" image $\mathbf{x}_t$ of a 90 μm tall ellipsoidal column of biofilm attached to a surface, taking value 10 inside the biofilm column, and 0 outside, in arbitrary units. Similar geometry has been observed experimentally for *Pseudomonas aeruginosa* biofilms (Swogger and Pitts [59]), and is also predicted by mathematical models of biofilm growth (Alpkvist and Klapper [2]). CSLM captures a set of planar "images" at different distances from the bottom of the biofilm where it is attached to a surface. In nature biofilms attach to any surface over which water flows, for example, human teeth and creek bottoms. Each horizontal planar image in this example is 100×100 pixels; the distance between pixels in each plane is typically about 1 μm, with the exact spatial resolution set by the microscope user. The vertical distance between planar slices in a CSLM image is typically an order of magnitude larger than the horizontal distance between pixels; for this example, the vertical distance between CSLM planes is 10 μm.

Given the "true" image $\mathbf{x}_t$, we generated synthetic $100 \times 100 \times 10$ CSLM data by

$$\mathbf{y} = \mathbf{F}\mathbf{x}_t + \varepsilon,$$

where the $10^5 \times 10^6$ matrix $\mathbf{F}$ arithmetically averages over 10 pixels in the vertical dimension of $\mathbf{x}$, to approximate the point spread function (PSF) of CSLM (Sheppard and Shotton [56]), and $\varepsilon \sim \mathrm{N}(\mathbf{0}, \mathbf{P}^{-1} = \mathbf{I})$. The data is displayed in the left panel of Figure 3 as layers of pixels, or "slices", located at the centre of sensitivity of the CSLM, that is, the centre of the PSF. Thus, the likelihood we consider is $\pi(\mathbf{y}|\mathbf{x}) = \mathrm{N}(\mathbf{F}\mathbf{x}, \mathbf{P}^{-1})$.

To encapsulate prior knowledge that the bacteria in the biofilm aggregate together, we model $\mathbf{x}$ by the GMRF $\mathbf{x} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_R^{-1})$ where the precision matrix $\mathbf{Q}_R$ models local smoothness of the density of the biofilm and background. We construct the matrix $\mathbf{Q}_R$ as a sparse inverse of the dense covariance matrix corresponding to the exponential covariance function. This construction uses the relationship between stationary Gaussian random fields and partial differential equations (PDEs) that was noted by Whittle [63] for the Matérn (or Whittle–Matérn (Guttorp and Gneiting [33])) class of covariance functions, that was also exploited by Cui *et al.* [13] and Lindgren *et al.* [38]. Rather than stating the PDE, we find it more convenient to work with the equivalent variational form, in this case (the square of)

$$\mathcal{Q}(x) = \int_{\mathcal{D}} \left(\frac{R}{4} |\nabla x|^2 + \frac{1}{4R} x^2 \right) dv + \int_{\partial \mathcal{D}} \frac{x^2}{2} \, ds,$$

where x is a continuous stochastic field, dv is the volume element in the domain $\mathcal{D}$ and ds is the surface element on the boundary $\partial \mathcal{D}$. This form has Euler–Lagrange equations being the

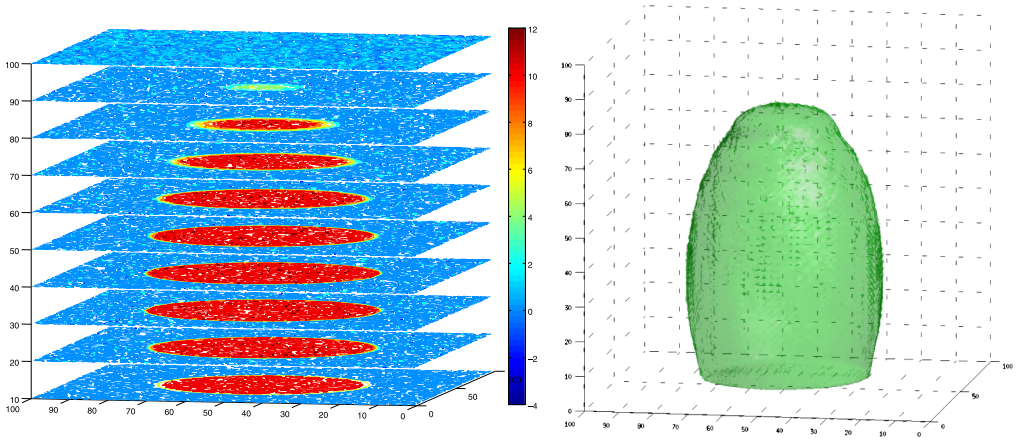

Figure 3. The left panel depicts a $100 \times 100 \times 10$ pixelated confocal scanning microscope image, $\mathbf{y}$ of a simulated ellipsoidal column of a bacterial biofilm; the distance between horizontal pixels is 1 μm, the distance between vertical pixels is 10 μm. The right panel shows a surface rendering of a sample from the $n = 10^6$ dimensional multivariate normal posterior distribution conditioned on hyperparameters.

Helmholtz operator with (local) Robin boundary conditions $x + R\frac{\partial x}{\partial n} = 0$ on $\partial\mathcal{D}$, induced by the $\frac{x^2}{2}$ term. In our example, we apply the Hessian of this form twice, which can be thought of as squaring the Helmholtz operator. When the quadratic form is written in the operator form $\mathcal{Q}(x) = x^T H x$, where H is the Hessian, the resulting Gaussian random field has density

$$\pi(x) \propto \exp\{-x^T H^2 x\}. \tag{19}$$

We chose this operator because the discretized precision matrix is sparse, while the covariance function (after scaling) is close to $\exp\{-r/R\}$, having length-scale R.

The GMRF over the discrete field $\mathbf{x}$ is then defined using FEM (finite element method) discretization; we used cubic-elements between nodes at voxel centres in the cubic domain, and tri-linear interpolation from nodal values within each element. To verify this construction, we show in Figure 4 contours of the resulting covariance function, between the pixel at the centre of the normalised cubic domain $[0, 1]^3$ and all other pixels, for length scale $R = 1/4$. The contours are logarithmically spaced in value, hence the evenly spaced spherical contours show that the covariance indeed has exponential dependence with distance. The contours look correct at the boundaries, indicating that the local Robin boundary conditions[4] give the desired covariance function throughout the domain. In contrast, Dirichlet conditions would make the cubic boundary a contour, while Neumann conditions as used by Lindgren *et al.* [38] would make contours perpendicular to the cubic boundary; neither of those pure boundary conditions produce the desired covariance function.

[4]Local boundary conditions are approximate but preserve sparseness. The exact boundary conditions are given by the boundary integral equation for the exterior Helmholtz operator, resulting in a dense block in H that is inconvenient for computation (Neumayer [44]).

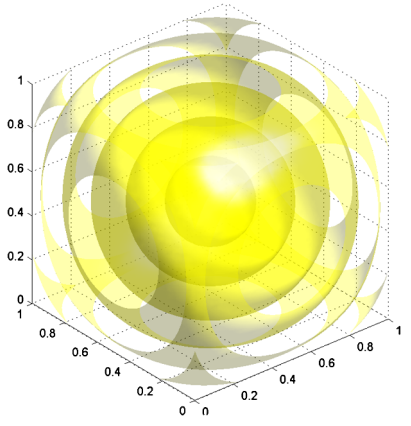

Figure 4. Contours of the effective covariance function centred on the cubic domain, logarithmically spaced in value.

In the deterministic setting, this image recovery problem is an example of a linear inverse problem. In the Bayesian setting, we may write the hierarchical model in the general form

$$\mathbf{y}|\mathbf{x},\boldsymbol{\theta} \sim \mathrm{N}\big(\mathbf{F}\mathbf{x}, \mathbf{P}_{\boldsymbol{\theta}}^{-1}\big), \tag{20}$$

$$\mathbf{x}|\boldsymbol{\theta} \sim \mathrm{N}\big(\boldsymbol{\mu}, \mathbf{Q}_{\boldsymbol{\theta}}^{-1}\big), \tag{21}$$

$$\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}), \tag{22}$$

where $\boldsymbol{\theta}$ is a vector of hyperparameters. This stochastic model occurs in many settings (see, e.g., Simpson *et al.* [57], Rue and Held [53]) with $\mathbf{y}$ being *observed data*, $\mathbf{x}$ is a *latent field*, and $\boldsymbol{\theta}$ is a vector of *hyperparameters* that parameterize the precision matrices $\mathbf{P}$ and $\mathbf{Q}$. The (hyper)prior $\pi(\boldsymbol{\theta})$ models uncertainty in covariance of the two random fields.

There are several options for performing sample-based inference on the model (20), (21), (22). Most direct is forming the posterior distribution $\pi(\mathbf{x},\boldsymbol{\theta}|\mathbf{y})$ via Bayes' rule and implementing Markov chain Monte Carlo (MCMC) sampling, typically employing Metropolis–Hastings dynamics with a random walk proposal on $\mathbf{x}$ and $\boldsymbol{\theta}$. Such an algorithm can be very slow due to high correlations within the latent field $\mathbf{x}$, and between the latent field and hyperparameters $\boldsymbol{\theta}$. More efficient algorithms block the latent field, noting that the distribution over $\mathbf{x}$ given everything else is a multivariate normal, and hence can be sampled efficiently as we have discussed in this paper. Higdon [35] and Bardsley [7] utilized this structure, along with conjugate hyperpriors on the components of $\boldsymbol{\theta}$, to demonstrate a Gibbs sampler that cycled through sampling from the conditional distributions for $\mathbf{x}$ and components of $\boldsymbol{\theta}$. When the normalizing constant for $\pi(\mathbf{x},\boldsymbol{\theta}|\mathbf{y})$ is available, up to a multiplicative constant independent of state, a more efficient algorithm is the *one block* algorithm (Rue and Held [53], Section 4.1.2) in which a candidate $\boldsymbol{\theta}'$ is drawn from a random walk proposal, then a draw $\mathbf{x}' \sim \pi(\mathbf{x}'|\mathbf{y},\boldsymbol{\theta}')$, with the joint proposal $(\boldsymbol{\theta}',\mathbf{x}')$ accepted with the standard Metropolis–Hastings probability. The resulting transition kernel in $\boldsymbol{\theta}$ is in detailed balance with the distribution over $\boldsymbol{\theta}|\mathbf{y}$, and hence can improve efficiency

dramatically. A further improvement is the *marginal algorithm* in which MCMC is performed directly on $\pi(\boldsymbol{\theta}|\mathbf{y})$ as indicated by Simpson *et al.* [57], with subsequent independent sampling $\mathbf{x} \sim \pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$ to facilitate Monte Carlo evaluation of statistics. In each of these schemes, computational cost is dominated by the cost of drawing samples from the large multivariate normal $\mathbf{x} \sim \pi(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$. We now demonstrate that sampling step for this synthetic example.

In our example, the distribution over the $100 \times 100 \times 100$ image $\mathbf{x}$, conditioned on everything else, is the multivariate normal

$$\pi(\mathbf{x}|\mathbf{y}, \theta = R) = \mathrm{N}\big(\mathbf{x}; \boldsymbol{\mu} = \mathbf{A}^{-1}\mathbf{F}^T\mathbf{P}\mathbf{y}, \boldsymbol{\Sigma} = \mathbf{A}^{-1}\big) \tag{23}$$

with precision matrix $\mathbf{A} = \mathbf{F}^T\mathbf{P}\mathbf{F} + \mathbf{Q}_R$ (cf. Calvetti and Somersalo [12], Higdon [35]). For this calculation, we used the same covariance matrix as shown above, so $R = 1/4$ in units of the width of the domain, though for sample-based inference one would use samples from the distribution over $R|\mathbf{y}$. The right panel of Figure 3 depicts a reconstructed surface derived from a sample from the conditional distribution in (23) using the Chebyshev polynomial accelerated SSOR sampler. The sampler was initialized with the precision matrix $\mathbf{A}$, $\mathrm{E}(\mathbf{c}^{(k)}) = \mathbf{F}^T\mathbf{P}\mathbf{y}$ for all k, and relaxation parameter $\omega = 1$. The contour is at value 6, after smoothing over $3 \times 3 \times 3$ voxels, displaying a sample surface that separates regions for which the average over $3 \times 3 \times 3$ voxel blocks is less than 6 (outside surface) and greater than 6 (inside surface). As can be seen, the surface makes an informative reconstruction of the ellipsoidal phantom.

Using CG, estimates of the extreme eigenvalue of $\mathbf{M}_{\mathrm{SSOR}}^{-1}\mathbf{A}$ were $\hat{\lambda}_{\min} = 4.38 \times 10^{-6}$ and $\hat{\lambda}_{\max} = 1 - 1.36 \times 10^{-8}$. By Corollary 6, the asymptotic convergence factors for the Chebyshev sampler are $\sigma \approx 0.9958$ for the mean and $\sigma^2 = 0.9917$ for the covariance matrix. Using this information, equation (16) predicts the number of sampler iterations until convergence. After $k^* = 4566$ iterations of the Chebyshev accelerated sampler, it is predicted that the mean error is reduced by $\varepsilon = 10^{-8}$; that is

$$\big\|\boldsymbol{\mu} - \mathrm{E}\big(\mathbf{y}^{(k^*)}\big)\big\|_2 \approx 10^{-8}\big\|\boldsymbol{\mu} - \mathrm{E}\big(\mathbf{y}^{(0)}\big)\big\|_2.$$

But even sooner, after only $k^{**} = k^*/2 = 2283$ iterations, it is predicted that the covariance error is

$$\big\|\mathbf{A}^{-1} - \mathrm{Var}\big(\mathbf{y}^{(k^{**})}\big)\big\| \approx 10^{-8}\big\|\mathbf{A}^{-1} - \mathrm{Var}\big(\mathbf{y}^{(0)}\big)\big\|.$$

Contrast these Chebyshev polynomial convergence results to the performance of the non-accelerated stationary SSOR sampler that has convergence factors $\varrho(\mathbf{M}^{-1}\mathbf{N}) \approx 1 - \hat{\lambda}_{\min} = 1 - 4.38 \times 10^{-6}$ for the mean error, and $\rho(\mathbf{M}^{-1}\mathbf{N})^2 = 1 - 8.76 \times 10^{-6}$ for covariance error. These convergence factors suggest that after running the non-accelerated SSOR Gibbs sampler for only 4566 iterations, the covariance error will be reduced to only $\varrho(\mathbf{M}^{-1}\mathbf{N})^{2 \cdot 4566} \approx 0.96$ of the original error; 1.9×10^6 iterations are required for a 10^{-8} reduction.

The cost difference between the Cholesky factorization and an iterative sampler in this example is dramatic. After finding a machine with the necessary n^2 memory requirements, the Cholesky factorization would cost about $b^2 n = 10^{16}$ flops (since the bandwidth of the precision matrix $\mathbf{A}$ is about $b = 10^5$). Since the number of non-zero elements of $\mathbf{A}$ is 3.3×10^8, an iterative sampler costs about 6.6×10^8 flops per iteration, much less than n^2. The sample in Figure 3 was generated by $k_{\max} = 5 \times 10^3$ iterations of the Chebyshev accelerated SSOR sampler, at a total cost of 3.3×10^{12} flops, which is about 10^4 times faster than Cholesky factoring.

7. Discussion

This work began, in part, with a curiosity about the convergence of the sequence of covariance matrices in Gibbs sampling applied to multivariate normal distributions, as studied by Liu *et al.* [39]. Convergence of that sequence indicates that the algorithm is implicitly implementing some factorization of the target covariance or precision matrix. Which one?

The answer was given by Goodman and Sokal [30], Amit and Grenander [4], Barone and Frigessi [8], and Galli and Gao [22], that the standard component-sweep Gibbs sampler corresponds to the classical Gauss–Seidel iterative method. That result is given in Section 4.2, generalized to arbitrary matrix splittings, showing that any matrix splitting used to generate a deterministic relaxation also induces a stochastic relaxation that is a generalized Gibbs sampler; the linear iterative relaxation and the stochastic relaxation share exactly the same iteration operator, conditions for convergence, and convergence factor, which may be summarized by noting that they share exactly the same error polynomial.

Equivalence of error polynomials is important because they are the central object in designing accelerated solvers including the multigrid, Krylov space, and parallel algorithms. We demonstrated that equivalence explicitly for polynomial acceleration, the basic non-stationary acceleration scheme for linear solvers, showing that this control of the error polynomial can be applied to Gibbs sampling from normal distributions. It follows that, just as for linear solvers, Chebyshev-polynomial accelerated samplers have a smaller average asymptotic convergence factor than their un-accelerated stationary counterparts.

The equivalences noted above are strictly limited to the case of normal target distributions. We are also concerned with continuous non-normal target distributions and whether acceleration of the normal case can usefully inform acceleration of sampling from non-normal distributions. Convergence of the unaccelerated, stationary, iteration applied to bounded perturbations of a normal distribution was established by Amit [3], though carrying over convergence rates proved more problematic.

There are several possibilities for extending the acceleration techniques to non-normal distributions. A straightforward generalization is to apply Gibbs sampling to the non-normal target, assuming the required conditional distributions are easy to sample from, though using the directions determined by the accelerated algorithm. Simply applying the accelerated algorithm to the non-normal distribution does not lead to optimal acceleration, as demonstrated by Goodman and Sokal [30].

A second route, that looks more promising to us, is to exploit the connection between Gibbs samplers and linear iterative methods that are often viewed as *local* solvers for non-linear problems, or equivalently, optimizers for local quadratic approximations to non-quadratic functions. Since a local quadratic approximation to $\log \pi$ is a local Gaussian approximation to π, the iterations developed here may be used to target this local approximation and hence provide local proposals in an MCMC. We imagine an algorithm along the lines of the trust-region methods from optimization in which the local quadratic (Gaussian) approximation is trusted up to some distance from the current state, implemented via a distance penalty. One or more steps of the iterative sampler would act as a proposal to a Metropolis–Hastings accept/reject step that ensures the correct target distribution. Metropolis adjusted Langevin (MALA) and hybrid Monte Carlo (HMC) turn out to be examples of this scheme (Norton and Fox [46]), as is the algorithm presented by Green and Han [31]. This naturally raises the question of whether acceleration of

the local iteration can accelerate the Metropolised algorithm. This remains a topic for ongoing research.

Appendix

A.1. Stationary sampler convergence (Proof of Theorem 2 and Corollary 3)

First, the theorem and corollary are established for the mean. Since $\mathbf{A} = \mathbf{M} - \mathbf{N}$ is a convergent splitting, then (10) and Theorem 1 show that $E(\mathbf{c}^{(k)}) = \boldsymbol{\nu}$ if and only if $E(\mathbf{y}^{(k)}) \to \mathbf{A}^{-1}\boldsymbol{\nu}$ with the same convergence factor as for the linear solver. To establish convergence of the variance, let $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$ in (10), then $\mathbf{y}^{(k)} = \mathbf{G}^k\mathbf{y}^{(0)} + \sum_{i=0}^{k-1} \mathbf{G}^i(\mathbf{M}^{-1}\mathbf{c}^{(k-1-i)})$. This equation and the independence of $\{\mathbf{c}^{(i)}\}$ show that $\mathbf{Var}(\mathbf{y}^{(k)}|\mathbf{y}^{(0)}) = \sum_{i=0}^{k-1}(\mathbf{G}^i\mathbf{M}^{-1}\operatorname{Var}(\mathbf{c}^i)\mathbf{M}^{-T}(\mathbf{G}^i)^T)$. Theorem 1 establishes the existence of a unique limiting distribution with a non-zero covariance matrix $\boldsymbol{\Gamma}$. Thus, for $\mathbf{y}^{(i)}, \mathbf{y}^{(i+1)} \sim \Pi$, (10) implies

$$\boldsymbol{\Gamma} = \mathbf{G}\boldsymbol{\Gamma}\mathbf{G}^T + \mathbf{M}^{-1}\operatorname{Var}\big(\mathbf{c}^{(i)}\big)\mathbf{M}^{-T} \tag{24}$$

since $\mathbf{y}^{(i)}$ and $\mathbf{c}^{(i)}$ are independent. Thus $\mathbf{Var}(\mathbf{y}^{(k)}|\mathbf{y}^{(0)}) = \boldsymbol{\Gamma} - \mathbf{G}^k\boldsymbol{\Gamma}(\mathbf{G}^k)^T$, and so

$$\mathbf{Var}\big(\mathbf{y}^{(k)}\big) = \boldsymbol{\Gamma} - \mathbf{G}^k\big(\boldsymbol{\Gamma} - \mathbf{Var}\big(\mathbf{y}^{(0)}\big)\big)\big(\mathbf{G}^k\big)^T. \tag{25}$$

That is, $\operatorname{Var}(\mathbf{y}^{(k)}) \to \boldsymbol{\Gamma}$ with convergence factor $\varrho(\mathbf{M}^{-1}\mathbf{N})^2$. To prove that part (b) of the theorem implies part (a), consider the starting vector $\mathbf{y}^{(0)} \sim \Pi$ with covariance matrix $\boldsymbol{\Gamma} = \mathbf{A}^{-1}$. Since $\mathbf{c}^{(k)}$ is independent of $\mathbf{y}^{(k)}$, the relation (24) shows that $\operatorname{Var}(\mathbf{c}^{(k)}) = \mathbf{M}(\mathbf{A}^{-1} - \mathbf{G}\mathbf{A}^{-1}\mathbf{G}^T)\mathbf{M}^T = \mathbf{M}\mathbf{A}^{-1}\mathbf{M}^T - \mathbf{N}\mathbf{A}^{-1}\mathbf{N}^T$. Substituting in $\mathbf{N} = \mathbf{M} - \mathbf{A}$ shows that $\operatorname{Var}(\mathbf{c}^{(k)}) = \mathbf{M}^T + \mathbf{N}$. To prove that (a) implies (b), consider $\mathbf{y}^{(0)} \sim \mathrm{N}(\boldsymbol{\mu}, \mathbf{A}^{-1})$. By (25), $\boldsymbol{\Gamma} - \operatorname{Var}(\mathbf{y}^{(1)}) = \mathbf{G}(\boldsymbol{\Gamma} - \mathbf{A}^{-1})\mathbf{G}^T$. Substituting $\operatorname{Var}(\mathbf{c}^{(0)}) = \mathbf{M}(\mathbf{A}^{-1} - \mathbf{G}\mathbf{A}^{-1}\mathbf{G}^T)\mathbf{M}^T$ into equation (24) shows $\boldsymbol{\Gamma} - \mathbf{A}^{-1} = \mathbf{G}(\boldsymbol{\Gamma} - \mathbf{A}^{-1})\mathbf{G}^T$. Thus $\operatorname{Var}(\mathbf{y}^{(1)}) = \mathbf{A}^{-1}$, which shows that $\operatorname{Var}(\mathbf{y}^{(k)})$ has converged to $\mathbf{A}^{-1}$. By Theorem 1, $\boldsymbol{\Gamma} = \mathbf{A}^{-1}$.

A.2. Polynomial accelerated sampler convergence (Proof of Theorem 5 and Corollary 6)

If the polynomial accelerated linear solver (11) converges, then $E(\mathbf{y}^{(k+1)}) \to \mathbf{A}^{-1}E(\mathbf{c}^{(k)}) = \boldsymbol{\mu}$. To determine $\operatorname{Var}(\mathbf{c}^{(k)})$ rewrite the iteration (17) as $\mathbf{y}^{(k+1)} = (1-\alpha_k)\mathbf{y}^{(k-1)} + \alpha_k\mathbf{G}^{(k)}\mathbf{y}^{(k)} + \alpha_k(\mathbf{M}^{(k)})^{-1}\mathbf{c}^{(k)}$ where $\mathbf{M}^{(k)} = \frac{1}{\tau_k}\mathbf{M}$, $\mathbf{N}^{(k)} = \mathbf{M}^{(k)} - \mathbf{A}$, and $\mathbf{G}^{(k)} = \mathbf{I} - \tau_k(\mathbf{M}^{(k)})^{-1}\mathbf{A} = (\mathbf{M}^{(k)})^{-1}\mathbf{N}^{(k)}$. First, we will consider $\mathbf{y}^{(i-1)} \sim \mathrm{N}(\boldsymbol{\mu}, \mathbf{A}^{-1})$ and then find $\operatorname{Var}(\mathbf{c}^{(k)})$ that will guarantee that $\mathbf{y}^{(i)}, \mathbf{y}^{(i+1)} \sim \mathrm{N}(\boldsymbol{\mu}, \mathbf{A}^{-1})$. Since $\{\mathbf{c}^{(i)}\}$ are independent of $\{\mathbf{y}^{(i)}\}$, the above equation for $\mathbf{y}^{(k+1)}$ shows that $\operatorname{Var}(\mathbf{c}^{(k)})$ is equal to

$$\frac{1}{\alpha_k^2}\mathbf{M}^{(k)}\big(\big(1-(1-\alpha_k)^2\big)\mathbf{A}^{-1} - 2(1-\alpha_k)\alpha_k\big(\mathbf{G}^{(k)}\mathbf{K}^{(k)} + \mathbf{K}^{(k)T}\mathbf{G}^{(k)T}\big) - \alpha_k^2\mathbf{G}^{(k)}\mathbf{A}^{-1}\mathbf{G}^{(k)T}\big)\mathbf{M}^{(k)},$$

where $\mathbf{K}^{(k)} := \mathbf{Cov}(\mathbf{y}^{(k-1)}, \mathbf{y}^{(k)})$. To simplify this expression, we need Lemma 9, which gives $\mathbf{K}^{(k)}$ explicitly. Parts (1) and (2) of the lemma show that

$$\operatorname{Var}\bigl(\mathbf{c}^{(k)}\bigr) = \frac{1}{\alpha_k^2}\mathbf{M}^{(k)}\bigl(\alpha_k^2\bigl(\mathbf{A}^{-1} - \mathbf{G}^{(k)}\mathbf{A}^{-1}\mathbf{G}^{(k)T}\bigr) + 2(1-\alpha_k)\alpha_k\bigl(\mathbf{A}^{-1} - \mathbf{G}_\kappa^{(k)}\mathbf{A}^{-1}\mathbf{G}^{(k)T}\bigr)\bigr)\mathbf{M}^{(k)}.$$

Part (3) of Lemma 9 shows that $\operatorname{Var}(\mathbf{c}^{(k)})$ has the form specified in the theorem.

Lemma 9. *For a symmetric splitting* $\mathbf{A} = \mathbf{M} - \mathbf{N}$,

1. $\mathbf{K}^{(k)}$ *is symmetric*.
2. $\mathbf{K}^{(k)} = \mathbf{G}_\kappa^{(k)}\mathbf{A}^{-1}$, *where* $\mathbf{G}_\kappa^{(k)} = \mathbf{I} - \kappa_k\mathbf{M}^{-1}\mathbf{A}$ *and* $\kappa_{k+1} := \alpha_k\tau_k + (1-\alpha_k)\kappa_k$.
3. $\mathbf{A}^{-1} - \mathbf{G}_\tau\mathbf{A}^{-1}\mathbf{G}_\kappa^T = \tau\kappa\mathbf{M}^{-1}((1/\tau + 1/\kappa)\mathbf{M} - \mathbf{A})\mathbf{M}^{-1}$.

Proof. To nail down $\mathbf{K}^{(k)}$, rewrite the Chebyshev iteration (17) as

$$\mathbf{Y}^{(k+1)} = \begin{pmatrix} \alpha_k\mathbf{G}^{(k)} & (1-\alpha_k)\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{pmatrix}\mathbf{Y}^{(k)} + \alpha_k\begin{pmatrix} \mathbf{g}^{(k)} \\ \mathbf{0} \end{pmatrix},$$

where $\mathbf{Y}^{(0)} = \binom{\mathbf{y}^{(0)}}{\mathbf{0}}$, $\mathbf{Y}^{(k+1)} = \binom{\mathbf{y}^{(k+1)}}{\mathbf{y}^{(k)}}$ and $\mathbf{g}^{(k)} = (\mathbf{M}^{(k)})^{-1}\mathbf{c}^{(k)}$. Letting $\mathcal{G}^{(k)} = \left(\begin{smallmatrix} \alpha_k\mathbf{G}^{(k)} & (1-\alpha_k)\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{smallmatrix}\right)$ shows that

$$\operatorname{Var}\bigl(\mathbf{Y}^{(k+1)}\bigr) = \mathcal{G}^{(k)}\operatorname{Var}\bigl(\mathbf{Y}^{(k)}\bigr)\mathcal{G}^{(k)T} + \alpha_k^2\begin{pmatrix} \operatorname{Var}\bigl(\mathbf{g}^{(k)}\bigr) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{26}$$

If $\operatorname{Var}(\mathbf{y}^{(0)}) = \mathbf{A}^{-1}$ then $\operatorname{Var}(\mathbf{y}^{(k)}) = \mathbf{A}^{-1}$ for $k \geq 1$ in which case $\operatorname{Var}(\mathbf{Y}^{(k+1)})$ is

$$\begin{pmatrix} \mathbf{A}^{-1} & \mathbf{K}^{(k+1)} \\ \mathbf{K}^{(k+1)T} & \mathbf{A}^{-1} \end{pmatrix} = \mathcal{G}^{(k)}\begin{pmatrix} \mathbf{A}^{-1} & \mathbf{K}^{(k)} \\ \mathbf{K}^{(k)T} & \mathbf{A}^{-1} \end{pmatrix}\mathcal{G}^{(k)T} + \begin{pmatrix} \alpha_k^2\operatorname{Var}\bigl(g^{(k)}\bigr) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{27}$$

By definition of $\mathbf{Y}^{(0)}$, $\mathbf{K}^{(0)} = \mathbf{0}$; for $k \geq 0$,

$$\mathbf{K}^{(k+1)} = \alpha_k\mathbf{G}^{(k)}\mathbf{A}^{-1} + (1-\alpha_k)\mathbf{K}^{(k)T}. \tag{28}$$

Since $\alpha_0 = 1$, then $\mathbf{K}^{(1)} = \mathbf{G}^{(0)}\mathbf{A}^{-1}$ which proves parts (1) and (2) of the lemma for $k = 0$ since $\kappa_1 = \tau_0$ and $\mathbf{G}_\kappa^{(k)}\mathbf{A}^{-1}$ is symmetric. Assuming that $\mathbf{K}^{(k)} = \mathbf{G}_\kappa^{(k)}\mathbf{A}^{-1}$ for $k > 0$, the recursion in (28) gives $\mathbf{K}^{(k+1)} = (\mathbf{I} - [\alpha_k\tau_k + (1-\alpha_k)\kappa_k]\mathbf{M}^{-1}\mathbf{A})\mathbf{A}^{-1}$ so the expansion and recursion hold for $k+1$, and parts (1) and (2) of the lemma follow by induction. Part (c) of the lemma follows from the equation

$$\mathbf{A}^{-1} - \mathbf{G}_\tau\mathbf{A}^{-1}\mathbf{G}_\kappa^T = \mathbf{M}_\tau^{-1}\bigl(\mathbf{M}_\tau\mathbf{A}^{-1}\mathbf{M}_\kappa^T\bigr)\mathbf{M}_\kappa^{-T} - \mathbf{M}_\tau^{-1}\mathbf{N}_\tau\mathbf{A}^{-1}\bigl(\mathbf{M}_\kappa^{-1}\mathbf{N}_\kappa\bigr)^T. \qquad \square$$

The selection of $\mathbf{Var}(\mathbf{c}^{(k)}) = a_k\mathbf{M} + b_k\mathbf{N}$ assures that if $\mathrm{Var}(\mathbf{y}^{(0)}) = \mathbf{A}^{-1}$, then $\mathrm{Var}(\mathbf{y}^{(k)}) = \mathbf{A}^{-1}$ for $k \geq 1$. Thus, subtracting (27) from (26) gives

$$\mathrm{Var}\big(\mathbf{Y}^{(k+1)}\big)\begin{pmatrix}\mathbf{A}^{-1} & \mathbf{K}^{(k+1)}\\ \mathbf{K}^{(k+1)T} & \mathbf{A}^{-1}\end{pmatrix} = \mathcal{G}^{(k)}\left(\mathrm{Var}\big(\mathbf{Y}^{(k)}\big) - \begin{pmatrix}\mathbf{A}^{-1} & \mathbf{K}^{(k)}\\ \mathbf{K}^{(k)T} & \mathbf{A}^{-1}\end{pmatrix}\right)\mathcal{G}^{(k)T}$$

or $\mathcal{E}^{(k+1)} = \mathcal{G}^{(k)}\mathcal{E}^{(k)}\mathcal{G}^{(k)T}$ for $k \geq 0$, where $\mathcal{E}^{(k)} = \mathrm{Var}(\mathbf{Y}^{(k)}) - \left(\begin{smallmatrix}\mathbf{A}^{-1} & \mathbf{K}^{(k)}\\ \mathbf{K}^{(k)T} & \mathbf{A}^{-1}\end{smallmatrix}\right)$. Hence, by recursion, $\mathcal{E}^{(k)} = (\prod_{l=0}^{k-1}\mathcal{G}^{(l)})\mathcal{E}^{(0)}(\prod_{l=0}^{k-1}\mathcal{G}^{(l)})^T$.

Denote the polynomial of the block matrix by $\mathcal{P}^{(k+1)} = (\prod_{l=0}^{k}\mathcal{G}^{(l)})$ that satisfies

$$\mathcal{P}^{(k+1)} = \mathcal{G}^{(k)}\mathcal{P}^{(k)} = \begin{pmatrix}\alpha_k\mathbf{G}^{(k)} & (1-\alpha_k)\mathbf{I}\\ \mathbf{I} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathcal{P}_{11}^{(k)} & \mathcal{P}_{12}^{(k)}\\ \mathcal{P}_{21}^{(k)} & \mathcal{P}_{22}^{(k)}\end{pmatrix}$$

with $\mathcal{P}^{(1)} = \mathcal{G}^{(0)} = \left(\begin{smallmatrix}\mathbf{G}^{(0)} & \mathbf{0}\\ \mathbf{I} & \mathbf{0}\end{smallmatrix}\right)$. Thus

$$\mathcal{P}_{11}^{(k+1)} = \alpha_k\mathbf{G}^{(k)}\mathcal{P}_{11}^{(k)} + (1-\alpha_k)\mathcal{P}_{11}^{(k-1)} = \alpha_k\big(\mathbf{I} - \tau_k\mathbf{M}^{-1}\mathbf{A}\big)\mathcal{P}_{11}^{(k)} + (1-\alpha_k)\mathcal{P}_{11}^{(k-1)}$$

with $\mathcal{P}_{11}^{(1)} = \mathbf{G}^{(0)}$, which shows that $\mathcal{P}_{11}^{(k+1)} = P_{k+1}$ by (12). Furthermore, this shows that the error in variance at the kth iteration has the specified form and convergence factor.

Acknowledgments

This work was partially funded by the New Zealand Institute for Mathematics and its Applications (NZIMA) thematic programme on Analysis, Applications and Inverse Problems in PDEs, and Marsden contract UOO1015.

Supplementary Material

Implementation details of the Chebyshev accelerated sampler (DOI: 10.3150/16-BEJ863SUPP; .pdf). An explicit algorithm for the Chebyshev SSOR sampler is provided.

References

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

[2] Alpkvist, E. and Klapper, I. (2007). A multidimensional multispecies continuum model for heterogeneous biofilm development. *Bull. Math. Biol.* **69** 765–789.

[3] Amit, Y. (1991). On rates of convergence of stochastic relaxation for Gaussian and non-Gaussian distributions. *J. Multivariate Anal.* **38** 82–99. MR1128938

[4] Amit, Y. and Grenander, U. (1991). Comparing sweep strategies for stochastic relaxation. *J. Multivariate Anal.* **37** 197–222. MR1114473

[5] Axelsson, O. (1996). *Iterative Solution Methods*. Cambridge: Cambridge Univ. Press. MR1276069
[6] Banerjee, S., Gelfand, A.E. and Carlin, B.P. (2003). *Hierarchical Modeling and Analysis for Spatial Data*. London: Chapman & Hall.
[7] Bardsley, J.M. (2012). MCMC-based image reconstruction with uncertainty quantification. *SIAM J. Sci. Comput.* **34** A1316–A1332. MR2970254
[8] Barone, P. and Frigessi, A. (1990). Improving stochastic relaxation for Gaussian random fields. *Probab. Engrg. Inform. Sci.* **23** 2901–2904.
[9] Barone, P., Sebastiani, G. and Stander, J. (2002). Over-relaxation methods and coupled Markov chains for Monte Carlo simulation. *Stat. Comput.* **12** 17–26. MR1877576
[10] Besag, J. (1986). On the statistical analysis of dirty pictures. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **48** 259–302. MR0876840
[11] Besag, J. and Green, P.J. (1993). Spatial statistics and Bayesian computation. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **55** 25–37. MR1210422
[12] Calvetti, D. and Somersalo, E. (2007). *Introduction to Bayesian Scientific Computing. Surveys and Tutorials in the Applied Mathematical Sciences* **2**. New York: Springer. MR2351679
[13] Cui, T., Fox, C. and O'Sullivan, M.J. (2011). Bayesian calibration of a large-scale geothermal reservoir model by a new adaptive delayed acceptance Metropolis Hastings algorithm. *Water Resources Research* **47** W10521.
[14] Diaconis, P. and Freedman, D. (1999). Iterated random functions. *SIAM Rev.* **41** 45–76. MR1669737
[15] Duflo, M. (1997). *Random Iterative Models. Applications of Mathematics* (*New York*) **34**. Berlin: Springer. MR1485774
[16] Feller, W. (1968). *An Introduction to Probability Theory and Its Applications. Vol. I*, 3rd ed. New York: Wiley. MR0228020
[17] Fishman, G.S. (1996). Coordinate selection rules for Gibbs sampling. *Ann. Appl. Probab.* **6** 444–465. MR1398053
[18] Fox, C. (2008). A conjugate direction sampler for normal distributions, with a few computed examples. Technical Report 2008-1, Electronics Group, University of Otago.
[19] Fox, C. and Parker, A. (2014). Convergence in variance of Chebyshev accelerated Gibbs samplers. *SIAM J. Sci. Comput.* **36** A124–A147. MR3162409
[20] Fox, C. and Parker, A. (2016). Supplement to "Accelerated Gibbs sampling of normal distributions using matrix splittings and polynomials." DOI:10.3150/16-BEJ863SUPP.
[21] Fox, L. and Parker, I.B. (1968). *Chebyshev Polynomials in Numerical Analysis*. Oxford: Oxford Univ. Press. MR0228149
[22] Galli, A. and Gao, H. (2001). Rate of convergence of the Gibbs sampler in the Gaussian case. *Math. Geol.* **33** 653–677. MR1956389
[23] Gelfand, A.E. and Smith, A.F.M. (1990). Sampling-based approaches to calculating marginal densities. *J. Amer. Statist. Assoc.* **85** 398–409. MR1141740
[24] Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B. (1995). *Bayesian Data Analysis. Texts in Statistical Science Series*. London: Chapman & Hall. MR1385925
[25] Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence* **6** 721–741.
[26] Geyer, C.J. (2011). Introduction to Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo* (S. Brooks, A. Gelman, G. L. Jones and X.-L. Meng, eds.). *Chapman & Hall/CRC Handb. Mod. Stat. Methods* 3–48. Boca Raton, FL: CRC Press. MR2858443
[27] Gilks, W.R., Richardson, S. and Spiegelhalter, D.J., eds. (1996). *Markov Chain Monte Carlo in Practice. Interdisciplinary Statistics*. London: Chapman & Hall. MR1397966
[28] Gneiting, T., Ševčíková, H., Percival, D.B., Schlather, M. and Jiang, Y. (2006). Fast and exact simulation of large Gaussian lattice systems in $\mathbb{R}^2$: Exploring the limits. *J. Comput. Graph. Statist.* **15** 483–501. MR2291260

[29] Golub, G.H. and Van Loan, C.F. (1989). *Matrix Computations*, 2nd ed. *Johns Hopkins Series in the Mathematical Sciences* **3**. Baltimore, MD: Johns Hopkins Univ. Press. MR1002570
[30] Goodman, J. and Sokal, A.D. (1989). Multigrid Monte Carlo method. Conceptual foundations. *Phys. Rev. D* **40** 2035–2071.
[31] Green, P. and Han, X. (1992). Metropolis methods, Gaussian proposals and antithetic variables. In *Stochastic models, statistical methods, and algorithms in image analysis* (*Rome*, 1990) (A. Frigessi, P. Barone and M. Piccioni, eds.). *Lecture Notes in Statistics* **74** 142–164. Berlin: Springer. MR1188484
[32] Grenander, U. (1983). Tutorial in pattern theory. Technical report, Division of Applied Mathematics, Brown University.
[33] Guttorp, P. and Gneiting, T. (2005). On the Whittle–Matérn correlation family. Technical Report 80, NRCSE University of Washington.
[34] Haario, H., Saksman, E. and Tamminen, J. (2001). An adaptive Metropolis algorithm. *Bernoulli* **7** 223–242. MR1828504
[35] Higdon, D. (2006). A primer on space–time modelling from a Bayesian perspective. In *Statistics of Spatio-Temporal Systems* (B. Finkenstadt, L. Held and V. Isham, eds.) 217–279. New York: Chapman & Hall/CRC.
[36] Horn, R.A. and Johnson, C.R. (1985). *Matrix Analysis*. Cambridge: Cambridge Univ. Press. MR0832183
[37] Lewandowski, Z. and Beyenal, H. (2014). *Fundamentals of Biofilm Research*. Boca Raton, FL: CRC Press.
[38] Lindgren, F., Rue, H. and Lindström, J. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **73** 423–498. MR2853727
[39] Liu, J.S., Wong, W.H. and Kong, A. (1995). Covariance structure and convergence rate of the Gibbs sampler with various scans. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **57** 157–169. MR1325382
[40] MacKay, D.J.C. (2003). *Information Theory, Inference and Learning Algorithms*. New York: Cambridge Univ. Press. MR2012999
[41] Meurant, G. (2006). *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations. Software, Environments, and Tools* **19**. Philadelphia, PA: SIAM. MR2261212
[42] Neal, R. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical Report 9702, Department of Statistics, University of Toronto.
[43] Neal, R. (1998). Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. In *Learning in Graphical Models* (M.I. Jordan, ed.) 205–225. Norwell: Kluwer Academic.
[44] Neumayer, M. (2011). Accelerated Bayesian Inversion and Calibration for Electrical Tomography Ph.D. thesis, Graz University of Technology.
[45] Nevanlinna, O. (1993). *Convergence of Iterations for Linear Equations. Lectures in Mathematics ETH Zürich*. Basel: Birkhäuser. MR1217705
[46] Norton, R.A. and Fox, C. (2014). Efficiency and computability of MCMC with Langevin, Hamiltonian, and other matrix-splitting proposals. Unpublished manuscript.
[47] Parker, A. and Fox, C. (2012). Sampling Gaussian distributions in Krylov spaces with conjugate gradients. *SIAM J. Sci. Comput.* **34** B312–B334. MR2970281
[48] Robert, C. and Casella, G. (2011). A short history of Markov chain Monte Carlo: Subjective recollections from incomplete data. *Statist. Sci.* **26** 102–115. MR2849912
[49] Robert, C.P. and Casella, G. (1999). *Monte Carlo Statistical Methods. Springer Texts in Statistics*. New York: Springer. MR1707311
[50] Roberts, G.O. and Rosenthal, J.S. (2007). Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *J. Appl. Probab.* **44** 458–475. MR2340211

[51] Roberts, G.O. and Sahu, S.K. (1997). Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **59** 291–317. MR1440584
[52] Rue, H. (2001). Fast sampling of Gaussian Markov random fields. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **63** 325–338. MR1841418
[53] Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications. Monographs on Statistics and Applied Probability* **104**. Boca Raton, FL: Chapman & Hall/CRC. MR2130347
[54] Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM. MR1990645
[55] Saad, Y. and van der Vorst, H.A. (2000). Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.* **123** 1–33. MR1798516
[56] Sheppard, C.J.R. and Shotton, D.R. (1997). *Confocal Laser Scanning Microscopy*. New York: Garland Science.
[57] Simpson, D., Lindgren, F. and Rue, H. (2012). Think continuous: Markovian Gaussian models in spatial statistics. *Spatial Statistics* **1** 16–29.
[58] Sokal, A.D. (1993). Discussion on the meeting on the Gibbs sampler and other Markov chain Monte Carlo methods. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **55** 87.
[59] Swogger, E. and Pitts, B. (2005). CSLM 3D view of pseudomonas aeruginosa biofilm structure. Engineering. Available at http://www.biofilm.montana.edu/resources/movies/2005/2005m06.html.
[60] Turčin, V. (1971). On the computation of multidimensional integrals by the Monte Carlo method. *Theory Probab. Appl.* **16** 720–724.
[61] Varga, R.S. (1962). *Matrix Iterative Analysis*. Englewood Cliffs, NJ: Prentice-Hall. MR0158502
[62] Watkins, D.S. (2002). *Fundamentals of Matrix Computations*, 2nd ed. *Pure and Applied Mathematics* (*New York*). New York: Wiley. MR1899577
[63] Whittle, P. (1954). On stationary processes in the plane. *Biometrika* **41** 434–449. MR0067450
[64] Young, D.M. (1971). *Iterative Solution of Large Linear Systems*. New York: Academic Press. MR0305568

Received April 2015 and revised December 2015