NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

Iterative Numerical Methods for Sampling from High Dimensional Gaussian Distributions

by

Aune, E., Eidsvik, J. and Pokern, Y.

N T N U

NORWEGIAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
TRONDHEIM, NORWAY

Iterative Numerical Methods for Sampling from High Dimensional Gaussian Distributions

Erlend Aune[1], Jo Eidsvik[1] and Yvo Pokern[2]
1 *Department of Mathematical Sciences, NTNU, Norway*
2 *Department of Statistics, University College, London, UK*

Many applications require efficient sampling from Gaussian distributions. The method of choice depends on the dimension of the problem as well as the structure of the covariance- ($\boldsymbol{\Sigma}$) or precision matrix ($\boldsymbol{Q}$). The most common black-box routine for computing a sample is based on Cholesky factorisation. In high dimensions, computing the Cholesky factor of $\boldsymbol{\Sigma}$ or $\boldsymbol{Q}$ may be prohibitive due to massive fill-in. We compare different methods for computing the samples iteratively adapting ideas from numerical linear algebra. These methods assume that matrix-vector products, $\boldsymbol{Qv}$, are fast to compute. We show that some of the methods are competitive and faster than Cholesky sampling and that a parallel version of one method on a Graphical Processing Unit (GPU) using CUDA can introduce a speed-up of up to 30x. Moreover, one method is used to sample from the posterior distribution of petroleum reservoir parameters in a North Sea field, given seismic reflection data on a large 3D grid.

1 Introduction

With the increased acquisition and storage of massive datasets much statistical research is focusing on inference and sampling in very high dimensions. The machine learning community constructs models for identifying information in such datasets, see e.g. Rasmussen and Wiliams (2006). In spatial statistics the introduction of new scientific methods, such as global positioning, seismic data acquisition and information systems acquiring massive datasets, are influencing the focus in models and methods, see e.g. Banerjee et al. (2008), Cressie and Johannesson (2008) and Buland et al. (2003). The most common distribution for such high dimensional problems is the Gaussian distribution. In diverse applications this model choice tends to give reasonable results, while maintaining computational tractability. The requirements for inference are then evaluation of a quadratic form and a determinant. For sampling based approaches we must be able to draw a variable with the right mean and covariance structure. Sampling is one way to do inference using Markov chain Monte Carlo sampling, but samples are also important for generating ensembles. Climate models, hydrological models, weather forecasting, petroleum reservoir prediction, and many other dynamic applications, rely on the propagation of such ensembles over time.

For applications which can be represented by a graphical structure, Gaussian Markov Random Fields (GMRFs), or conditional autoregresive (CAR) models, provide useful conditional independence representations of Gaussian processes. For instance, spatio-temporal data on a grid or on a regionalized areal model, are often modeled by a GMRF prior model, see e.g. Besag et al. (1991) and Rue and Held (2005). A GMRF is characterized by a sparse precision matrix obtained from the conditional formulation, giving non-zero entries only on the diagonal and at entries within the neighbourhood structure of the related graph. This sparse structure allows for efficient computations. In contrast, the covariance matrix (inverse precision matrix) tends to be almost full. The sampling methods using the covariance matrix may thus be much less efficient in high dimensions, unless one is able to utilize some approximation or basis representation of the process. For instance, one can use the fast fourier transform for stationary Gaussian processes on a torus (Gray (2006)).

In this paper we explore iterative methods for sampling high dimensional Gaussian processes. We assume that the precision matrix $\boldsymbol{Q}$ has a sparse Markov structure, or the matrix-vector product $\boldsymbol{Qx}$ is available as a fast black-box procedure. Here, 'iterative methods' mean iterative numerical linear algebra methods, see Saad (2003), Golub and van Loan (1996) and Trefethen and Bau (1997). We do not refer to random iterative sampling methods such as Markov chain Monte Carlo (MCMC). Our proposed sampling methods use numerical methods to solve $\boldsymbol{x} = \boldsymbol{Q}^{-1/2}\boldsymbol{z}$, where $\boldsymbol{z}$ is a vector of iid normal variables. The error can be controlled using the accuracy of the functional (numerical) approximation. This is very different from MCMC algorithms which study the 'error' by checking the loss of any transient phase and the lack of autocorrelation in the Markov chain.

We outline three general procedures for sampling. All three are based on Krylov methods (Saad (2003)) and rational approximations to a specific matrix function (Higham (2008)). The first of these procedures is the traditional Lanczos method and variations over that (Saad (2003)). The second is based on a quadrature representation formula arising from Cauchy's integral formula (for an accessible introduction to complex analysis, see Stein and Shakarchi (2003)) and conformal mappings of regions $U \in \mathbb{C}$, see Hale et al. (2008). The third is a continuous deformation method based on a system of ODEs found in Allen et al. (2000). Iterative methods are used for solving $\boldsymbol{Qx} = \boldsymbol{b}$ many times. We employ variants of the conjugate gradient (CG) method of Hestenes and Stiefel (1952) for this. The basis of these methods was developed by the numerical linear algebra community, and they are common in many applications of large datasets with sparse structure. Our impression is that their merits can be useful to statisticians, since their computational and mathematical properties can be superior to the more classical (direct) linear algebra tools that are commonly used in statistics today. Moreover, methods using sparse matrices can be implemented quite easily on the graphics card (GPU), allowing fast parallel computing. A tutorial on using CUDA, a C++/C interface to the GPU, can be found on nVidias website (http://www.nvidia.com/cuda/). Statisticians will certainly use more of these recent parallel developments in the future. In our examples we get speed-ups of up to a factor of 30.

We briefly review recently proposed methods for iterative sampling utilizing and adapting ideas from numerical linear algebra. Schneider and Willsky (2003) and Fox (2008) study Krylov subspace approximations, where samples are the directions obtained by the CG method. Their realizations are fast to generate, but they oversmooth the process and are impractical for large problems due to inherrent instability in the presence in round-off error in the orthogonal vectors generated by the Krylov method. Simpson et al. (2007) describe

restarted Lanczos routines for constructing a sample. For very high dimension the storage capacity seems very large for this method, but this can be overcome using a so-called two-pass strategy described in Frommer and Simoncini (2008). In the machine learning literature, Belabbas and Wolfe (2009) used iterative methods to approximate the eigenvalues, and capture the most important feature of the Gaussian process. Their approach seems to work well for moderate dimensions, but it is unclear what its properties are, and how to tune this method in high dimensions.

The paper is organised as follows: Section 2 gives some background model assumptions and a review of direct sampling methods. In Section 3 we present the iterative sampling procedures, which are applied to examples in Section 4. In this example section, we compare timings for two different sparse models, and sample the posterior model given seismic 3D data from a North Sea reservoir.

2   Modelling assumptions and direct sampling methods

The distribution of a k dimensional Gaussian random variable $\boldsymbol{x} = (x_1, \ldots, x_k)'$, denoted $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q}^{-1})$ for an invertible $\boldsymbol{Q}$, is given by

$$p(\boldsymbol{x}) = \frac{|\boldsymbol{Q}|^{1/2}}{(2\pi)^{k/2}} \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{\mu}))$$
$$\propto \exp(-\frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{b}), \tag{1}$$

where covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{Q}^{-1}$, and $\boldsymbol{Q}$ is the precision matrix. The linear canonical parameter $\boldsymbol{b} = \boldsymbol{Q}\boldsymbol{\mu}$ links the mean $\boldsymbol{\mu}$ and the precision matrix. For GMRFs the precision matrix $\boldsymbol{Q}$ is sparse, with $Q_{i,j} = 0$ unless i, j are neighbors on a graph. On a grid, the first order neighborhood is defined by the cells north, east, south and west. For a map of regions, the neighbors have a common border. The extension to second and higher order neighborhoods follows naturally (Rue and Held (2005)). We will treat the precision matrix and the mean as fixed. In the application to seismic data one typically determines these parameters from auxiliary data sources.

2.1   Gaussian linear model

In many applications the GMRF constitutes a latent process, while the data are noisy, possibly indirect, measurements of this process. Then $p(\boldsymbol{x})$ takes the form of a prior model, while the data $\boldsymbol{y} = (y_1, \ldots, y_n)^T$ are represented via a likelihood model. Here, we consider a Gaussian linear likelihood model $\boldsymbol{y}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{G}\boldsymbol{x}, \boldsymbol{R}^{-1})$, with $n \times k$ forward matrix $\boldsymbol{G}$ and noise covariance matrix $\boldsymbol{R}^{-1}$ determined by the data acquisition procedure. The posterior distribution for the latent process $\boldsymbol{x}$ given the data $\boldsymbol{y}$ is

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto \exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{\mu})-$$
$$\frac{1}{2}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x})^T \boldsymbol{R}(\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x}))$$
$$\propto \exp(-\frac{1}{2}\boldsymbol{x}^T(\boldsymbol{Q} + \boldsymbol{G}^T \boldsymbol{R}\boldsymbol{G})\boldsymbol{x}+$$
$$\boldsymbol{x}^T(\boldsymbol{Q}\boldsymbol{\mu} + \boldsymbol{G}^T \boldsymbol{R}\boldsymbol{y})). \tag{2}$$

This posterior is a Gaussian process with precision $\boldsymbol{Q} \to \boldsymbol{Q} + \boldsymbol{G}^T \boldsymbol{R}\boldsymbol{G}$ and canonical parameter $\boldsymbol{b} \to \boldsymbol{b} + \boldsymbol{G}^T \boldsymbol{R}\boldsymbol{y}$, compared with the prior distribution. The posterior mean is $E(\boldsymbol{x}|\boldsymbol{y}) = (\boldsymbol{Q}+\boldsymbol{G}^T \boldsymbol{R}\boldsymbol{G})^{-1}(\boldsymbol{Q}\boldsymbol{\mu}+\boldsymbol{G}^T \boldsymbol{R}\boldsymbol{y})$. Commonly, the likelihood is modeled as conditionally independent, i.e. $\boldsymbol{G}$ and $\boldsymbol{R}$ are diagonal. For this situation, the posterior $p(\boldsymbol{x}|\boldsymbol{y})$ inherits the neighborhood structure of the prior $p(\boldsymbol{x})$, with a change in $\boldsymbol{b}$ and the diagonal entries of $\boldsymbol{Q}$ alone. In some applications the likelihood might involve smoothing, either from $\boldsymbol{G}$ or $\boldsymbol{R}$. This increases the neighborhood of the posterior model, and some of the sparse computational benefits might be reduced.

We consider sampling methods for the prior in (1) and the posterior in (2). Posterior sampling can be done either by sampling from the Gaussian process defined via the conditional $p(\boldsymbol{x}|\boldsymbol{y})$, or as a three-step procedure which i) generates a variable from the prior Gaussian process, ii) draws a randomized data value from the Gaussian likelihood, and iii) computes a linear combination of the two that maintains the correct conditional mean and variance. The simpler strategy will depend on the situation.

2.2 Direct sampling of Gaussian processes

In this section we review direct methods for sampling a Gaussian process. A key observation when sampling from a Gaussian is the following: If the precision matrix $\boldsymbol{Q} = \boldsymbol{L}\boldsymbol{L}^T$, then the covariance is $\boldsymbol{\Sigma} = (\boldsymbol{L}\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-T}\boldsymbol{L}^{-1}$. If we want a sample $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}^{-1})$, it is enough to compute $\boldsymbol{v} = \boldsymbol{L}^{-T}\boldsymbol{z}$ for $\boldsymbol{z} \sim \mathcal{N}(0, I)$, since $\operatorname{Cov}(\boldsymbol{L}^{-T}\boldsymbol{z}) = \boldsymbol{L}^{-T}\operatorname{Cov}(\boldsymbol{z})\boldsymbol{L}^{-1} = (\boldsymbol{L}\boldsymbol{L}^T)^{-1}$. This approach is called the Cholesky sampling from a GMRF (Rue (2001) and Rue and Held (2005)). In an autoregressive graph with first order neighborhood, $\boldsymbol{L}$ has non-zero entries only along the diagonal and the first subdiagonal. The Cholesky factor is fast to compute from $\boldsymbol{Q}$; more specifically for the autoregressive graph, the computational cost is of order $O(n)$. For a two dimensional grid the cost is $O(n^{3/2})$, for a three dimensional grid it is $O(n^2)$. Moreover, the storage requirements for computing $\boldsymbol{L}$ become enormous in high dimensions because of the large fill-in between the non-zero structure of $\boldsymbol{Q}$ and the larger non-zero structure of $\boldsymbol{L}$. One can reduce the storage requirements by intelligent sorting of the n indices in the graph, but for a three dimensional grid, the sortings we tried did not prove particularly helpful in our study. A remedy is to apply Cholesky factorisation for block updating in an MCMC sampler (Roberts and Sahu (1997)), but the burn-in and mixing of the resulting Markov chain can be quite slow.

A different point of view comes from considering $\boldsymbol{Q}^{-1/2}$ as the principal square root of the matrix $\boldsymbol{Q}$. Since $\boldsymbol{Q}$ is symmetric positive definite, $\boldsymbol{Q} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$, where $\boldsymbol{V}$ is the orthogonal eigenvector matrix and $\boldsymbol{D}$ has the eigenvalues of $\boldsymbol{Q}$ on its diagonal. Consequently, we have $\boldsymbol{Q}^{-1/2} = \boldsymbol{V}\boldsymbol{D}^{-1/2}\boldsymbol{V}^T$. Then for $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$,

$$
\begin{aligned}
\operatorname{Cov}(\boldsymbol{V}\boldsymbol{D}^{-1/2}\boldsymbol{V}^T\boldsymbol{z}) &= \boldsymbol{V}\boldsymbol{D}^{-1/2}\boldsymbol{V}^T\boldsymbol{I}(\boldsymbol{V}\boldsymbol{D}^{-1/2}\boldsymbol{V}^T)^T \\
&= \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{V}^T = \boldsymbol{Q}^{-1},
\end{aligned}
\tag{3}
$$

as desired. In high dimensions the eigenvalues and eigenvectors are very hard to compute directly, namely it is $\mathcal{O}(n^3)$, unless there is particular structure in the model. For instance, if the precision matrix $\boldsymbol{Q}$ is circulant or is well approximated by a circulant matrix, the eigenvalues and eigenvectors are easy to compute using the fast Fourier transform (Gray (2006)). This gives an algorithm of order $\mathcal{O}(n \log n)$. Any stationary GMRF may be approximated by a circulant $\boldsymbol{Q}$ through tapering.

It is important to note that Cholesky and matrix function sampling give different samples $\boldsymbol{x}$, even though they use the same input iid variable $\boldsymbol{z}$. Still, both realizations are from the correct distribution, and correspond to one another through an injective function. Even though the Cholesky method and the eigenrepresentation do not allow direct sampling in very high dimensions, they are often used as building blocks for iterative sampling methods. For instance, the fast fourier transform on circulant matrices and incomplete Cholesky factorization are popular preconditioning methods for iterative numerical methods. Some of the iterative techniques presented in Section 3 rely on approximating representation (3).

3 Iterative numerical methods for sampling

In our setting, the term 'iterative' means that we build a Krylov subspace $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0) = \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{Q}\boldsymbol{r}_0, \boldsymbol{Q}^2\boldsymbol{r}_0, \ldots, \boldsymbol{Q}^m\boldsymbol{r}_0\}$, or a solution in that space in an iterative fashion. Here, $\boldsymbol{r}_0$ is the initial residual, $\boldsymbol{r}_0 = \boldsymbol{z} - \boldsymbol{Q}\boldsymbol{x}_0$, where $\boldsymbol{x}_0$ is the initial guess. We use $\boldsymbol{x}_0 = \boldsymbol{0}$ and $\boldsymbol{r}_0 = \boldsymbol{z}$, where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. From representation (3), we note that it is sufficient to consider function approximations of the inverse square root on the spectrum of $\boldsymbol{Q}$. More specifically, we use rational approximations to $f_N(\boldsymbol{Q})\boldsymbol{z} \approx \boldsymbol{Q}^{-1/2}\boldsymbol{z}$ on the spectrum of $\boldsymbol{Q}$. That is, we set

$$
\boldsymbol{x} = \boldsymbol{Q}^{-1/2}\boldsymbol{z} \approx f_N(\boldsymbol{Q})\boldsymbol{z} = \sum_{j=1}^{N} \alpha_j (\boldsymbol{Q} - \sigma_j \boldsymbol{I})^{-1}\boldsymbol{z},
\tag{4}
$$

for $\sigma_j \in \mathbb{R}_-, \alpha_j \in \mathbb{R}$. The specification of α_j, σ_j differs between the various iterative methods in 3.1-3.3. Using the rational approximation (4) requires that we solve the linear system

$$
(\boldsymbol{Q} - \sigma_j \boldsymbol{I})\boldsymbol{x} = \boldsymbol{z},
\tag{5}
$$

for many σ_js. A straightforward way of solving these systems is the CG algorithm first published by Hestenes and Stiefel (1952). This Krylov subspace method is summarised in Algorithm 1. See also Saad (2003). For

---

Algorithm 1 Conjugate gradient algorithm for computing $\boldsymbol{x} = \boldsymbol{Q}^{-1}\boldsymbol{z}$

Set: $\boldsymbol{r}_{cur} = \boldsymbol{z} - \boldsymbol{Q}\boldsymbol{x}_0$, $\boldsymbol{p} = \boldsymbol{r}_{cur}$

for $j = 1...$ to converged **do**

   $\boldsymbol{q}_p = \boldsymbol{Q}\boldsymbol{p}$

   $\alpha = \frac{\langle \boldsymbol{r}_{cur}, \boldsymbol{r}_{cur} \rangle}{\langle \boldsymbol{q}_p, \boldsymbol{p} \rangle}$

   $\boldsymbol{x} \rightarrow \boldsymbol{x} + \alpha \boldsymbol{p}$

   $\boldsymbol{r}_{new} = \boldsymbol{r}_{cur} - \alpha \, \boldsymbol{q}_p$

   $\beta = \frac{\langle \boldsymbol{r}_{new}, \boldsymbol{r}_{new} \rangle}{\langle \boldsymbol{r}_{cur}, \boldsymbol{r}_{cur} \rangle}$

   $\boldsymbol{p} = \boldsymbol{r}_{new} + \beta \boldsymbol{p}$

   $\boldsymbol{r}_{cur} = \boldsymbol{r}_{new}$

end for

---

these methods, the matrix vector product $\boldsymbol{Qz}$ is the computationally intensive part, and sparse matrices or fast black-box matrix-vector product routines are essential for fast convergence in high dimensions. Convergence speed also depends on the condition number, $\kappa = \lambda_{max}/\lambda_{min}$, of $\boldsymbol{Q}$, see Saad (2003) and Golub and van Loan (1996). If a matrix has a particularly bad condition number, a possible remedy is using a preconditioner, $\boldsymbol{M}$. Now, we solve, for instance, the system $\boldsymbol{MQx} = \boldsymbol{Mb}$ instead of the original system, and hopefully this system has spectrum better suited for CG iterations. Note that for this to be efficient the matrix-vector product $\boldsymbol{Mr}$ must be fast to compute. Typically, $\boldsymbol{M} \approx \boldsymbol{Q}^{-1}$, but is much faster to compute than $\boldsymbol{Q}^{-1}$. We mention also that apart from in the usual CG method, preconditioning can be difficult. Moreover, preconditioners can be hard to parallelize. This needs to be considered when employing a method in practice.

We also mention that all the methods in this article can easily be modified to facilitate models in which the matrix-vector product $\boldsymbol{\Sigma x}$ is fast, where $\boldsymbol{\Sigma}$ is a covariance matrix.

3.1   Lanczos methods for the inverse square root

In the context of iterative methods for sampling $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}^{-1})$, the Lanczos method plays a prominent role. It is the building block for self-adjoint Krylov methods and is one of the easiest ways of forming an orthonormal basis for $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0)$. The method for computing the inverse square root is basically: Obtain $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0)$ and compute the inverse square root on this subspace - stop when the approximation is good enough. The Lanczos algorithm is presented in Algorithm 2. This algorithm is also the basis for the cg algorithm, and to deduce it, use the $\boldsymbol{LU}$ factorisation sequentially on the tridiagonal matrix produced. In problems in which the matrices are not symmetric, the non-symmetric counterpart to the Lanczos method is the Arnoldi method. In this case, the Hessenberg matrix is not tridiagonal.

Algorithm 2 Lanczos algorithm for computing $\boldsymbol{x} = \boldsymbol{Q}^{-1/2}\boldsymbol{z}$

Set: $\boldsymbol{r}_0 = \boldsymbol{z} - \boldsymbol{Q}\boldsymbol{x}_0$, $\beta_0 = \|\boldsymbol{r}_0\|$, $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta_0$

for $j = 1$ to m **do**

   $\boldsymbol{w}_j = \boldsymbol{Q}\boldsymbol{v}_j - \beta_j \boldsymbol{v}_{j-1}$ $(\boldsymbol{v}_0 = \boldsymbol{0})$

   $\alpha_j = \langle \boldsymbol{w}_j, \boldsymbol{v}_j \rangle$

   $\boldsymbol{w}_j = \boldsymbol{w}_j - \alpha_j \boldsymbol{v}_j$

   $\beta_{j+1} = \|\boldsymbol{w}_j\|$

   $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/\beta_{j+1}$

end for

Set: $\boldsymbol{T}_m = \text{tridiag}(\beta, \alpha, \beta)$, $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m]$

Compute: $\boldsymbol{x} = \boldsymbol{x}_0 + \beta_0 \boldsymbol{V} \boldsymbol{T}^{-1/2} \boldsymbol{e}_1$

A challenge in this type of method is loss of orthogonality of the basis for $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0)$ and also choosing the number of basis vectors to store from iteration to iteration. Reorthogonalisation and restarting are ways to address this, another way to circumvent the storage bloating is to use a so-called 2-pass strategy which essentially entails computing the Lanczos approximation given in Algorithm 2 two times. This 2-pass strategy is mentioned in Frommer and Simoncini (2008) and we implement a version of it here. The restarted Arnoldi method used

in ARPACK is described in Lehoucq et al. (1998), a restarted Lanczos method for computing $\boldsymbol{x} = \boldsymbol{Q}^{-1/2}\boldsymbol{z}$ is found in Ilic et al. (2009), a reorthogonalisation procedure is for instance Algorithm 6.6 in Saad (2003). It has, however, been showed that loss of orthogonality is not a big issue in practice (Simpson (2008)).

In the Lanczos algorithm m is typically much smaller than the dimension of the matrix. If $m < 2000$, it is possible to compute the eigen decomposition of $\boldsymbol{T}$ and the resulting sample x in a reasonable amount of time. An alternative approach comes from considering rational approximations, $\boldsymbol{T}^{-1/2}\boldsymbol{e}_1 \approx f_N(\boldsymbol{T})\boldsymbol{e}_1$, as in (4). This essentially requires a fast tridiagonal solver, and such solvers have computational complexity of $\mathcal{O}(n)$.

In the 2-pass Lanczos algorithm (variant of Algorithm 2) we only need $\boldsymbol{v}_j, \boldsymbol{v}_{j-1}$ in each iteration to compute α_j, β_j. We exploit this, and compute $\boldsymbol{T}$) first, discarding the older $\boldsymbol{v}_i$s. In the next pass, we compute $\boldsymbol{x}_j = (\boldsymbol{T}^{-1/2}\boldsymbol{e}_1)_j\boldsymbol{v}_j$ and sum the $\boldsymbol{x}$s as we pass through the Lanczos iterations once more. The link between the Lanczos procedure and the conjugate gradient algorithm (Saad (2003), chapter 6) gives an equivalence between looking at Lanczos approximations coupled with rational approximations and rational approximations with solutions of the shifted systems computed with Krylov methods.

Simpson et al. (2007) present a theorem for the error of the Lanczos approximation: Let $\boldsymbol{Q}$ be symmetric positive definite with largest and smallest eigenvalues $\lambda_{min}, \lambda_{max}$ respectively. Then

$$\|\boldsymbol{Q}^{-1/2}\boldsymbol{z} - \|\boldsymbol{z}\|_2\boldsymbol{V}\boldsymbol{T}^{-1/2}\boldsymbol{e}_1\|_2 \leq \lambda_{min}^{-1/2}\|\boldsymbol{r}\|, \tag{6}$$

where $\boldsymbol{r}$ is the residual after m iterations of conjugate gradients to solve $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{z}$. This theorem essentially says that we can use the residual of the CG algorithm to find the number of iterations required to obtain an appropriate approximation. The CG coefficients are available essentially for free through explicit formulae (Saad (2003)), and we can modify the algorithm to accomodate this. Since we want to compute several samples, it is practically more efficient to precompute the number of Krylov dimensions required using the CG algorithm. We do this precomputation on a number of samples $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$, and use the dimension of the largest Krylov subspace needed in the Lanczos approximation of the inverse square root times a vector.

Another Lanczos-type algorithm we have implemented involves deflating some orthonormal vectors into the Lanczos procedure. That is, given some orthonormal vectors $\{\boldsymbol{w}_i\}_{i=1}^s$ $(s < n, m)$ in the eigenspace of $\boldsymbol{Q}$, we construct a Krylov space $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0) \perp \boldsymbol{w}_i \ \forall i \in 1, \ldots, s$. This has the effect of improving the conditioning of the system as per a preconditioner Saad et al. (1999). Specifically, let $\boldsymbol{W} = [\boldsymbol{w}_1 \boldsymbol{w}_2 \cdots \boldsymbol{w}_s]$ be the given orthonormal eigenvectors, and $\lambda_i, i = 1, \ldots, s$ are the correponding eigenvalues. In order to construct a Krylov basis which is orthogonal to $\boldsymbol{W}$, let $\boldsymbol{x}_0 = \boldsymbol{x}_{-1} + \boldsymbol{W} \times (\boldsymbol{W}^T\boldsymbol{Q}\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{r}_{-1}$, with $\boldsymbol{x}_{-1}$ arbitrary (e.g. $\boldsymbol{x}_{-1} = \boldsymbol{0}$) and $\boldsymbol{r}_{-1} = \boldsymbol{z} - \boldsymbol{Q}\boldsymbol{x}_{-1}$. This is a projection of the solution of $\boldsymbol{Q}\boldsymbol{x} = \boldsymbol{z}$ onto the space spanned by the $\boldsymbol{w}_i$s. This initial value, $\boldsymbol{x}_0$, ensures that the Krylov vectors $\boldsymbol{v}_i$ are orthogonal to the $\boldsymbol{w}_i$s. Now, compute the Lanczos decomposition using a two-pass version, compute the approximation $\boldsymbol{x}_{Krylov} = \boldsymbol{V}\boldsymbol{T}^{-1/2}\boldsymbol{e}_1$, and set $\boldsymbol{x}_{Proj} = \boldsymbol{W}\Lambda^{-1/2}\boldsymbol{W}'\boldsymbol{z}$. Finally, the approximate solution is $\boldsymbol{x} = \boldsymbol{x}_{Krylov} + \boldsymbol{x}_{Proj}$. Note that if $\boldsymbol{W}$ has eigenvector columns we get $\boldsymbol{W}^T\boldsymbol{Q}\boldsymbol{W} = \text{diag}(\lambda_1, \ldots, \lambda_s)$. The procedure is summarised in algorithm 3 ($\oslash$ is element-wise division).

An obvious drawback of algorithm 3 is the additional storage requirements of the approximated eigenvectors of $\boldsymbol{Q}$. Also, there is some overhead in the matrix vector, matrix-matrix computations in the algorithm, but most of this can be overcome by precomputing $\boldsymbol{Q}\boldsymbol{W}, \boldsymbol{W}^T\boldsymbol{Q}$ and $(\boldsymbol{W}\boldsymbol{Q}\boldsymbol{W}^T)^{-1}$. This comes at the cost of approximately tripling the initial vector storage requirements. However, this precomputation seems necessary to make the algorithm competitive.

The orthonormal vectors need not necessarily be eigenvectors, but can be a wavelet decomposition or any other basis decomposition that contains much information with few vectors. We may optionally choose to focus the projection on a subspace where we need more accuracy, and for that we need to deflate an orthonormal basis of that subspace.

3.2 Optimal rational approximations with linear solves

One way of approaching rational approximations as in (4) is through numerical quadrature of a contour integral. For functions that are analytic in some domain containing the spectrum of $\boldsymbol{Q}$ it is possible to compute $f(\boldsymbol{Q})\boldsymbol{z}$ by Cauchy's integral formula

$$f(\boldsymbol{Q})\boldsymbol{z} = \frac{1}{2\pi i} \oint_{\Gamma} f(\zeta)(\zeta\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}d\zeta, \tag{7}$$

Algorithm 3 2-pass deflated eigenvector Lanczos algorithm for $\boldsymbol{Q}^{-1/2}\boldsymbol{z}$

Input: $\boldsymbol{W}$, $\boldsymbol{z}$, $\boldsymbol{Q}$, $\boldsymbol{W}_Q = \boldsymbol{W}^T\boldsymbol{Q}$, $\boldsymbol{Q}_W = \boldsymbol{Q}\boldsymbol{W}$, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_s)^T$

Set: $\boldsymbol{r}_{-1} = \boldsymbol{z}$, $\boldsymbol{x}_0 = \boldsymbol{W}\left((\boldsymbol{W}^T\boldsymbol{r}_{-1}) \oslash \boldsymbol{\lambda}\right)$, $\boldsymbol{r}_0 = \boldsymbol{z} - \boldsymbol{Q}\boldsymbol{x}_0$, $\beta_1 = \|\boldsymbol{r}_0\|$ $\boldsymbol{v}_{cur} = \boldsymbol{r}_0/\beta_1, \boldsymbol{v}_{old} = \boldsymbol{0}$

for $j = 1$ to m or converged (1st pass) **do**

    if $j = 1$ **then**

        $\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{v}_{cur} - \boldsymbol{Q}_W(\boldsymbol{W}_Q\boldsymbol{v}_{cur} \oslash \boldsymbol{\lambda})$

    else

        $\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{v}_{cur} - \boldsymbol{Q}_W(\boldsymbol{W}_Q\boldsymbol{v}_{cur} \oslash \boldsymbol{\lambda}) - \beta_j\boldsymbol{v}_{old}$

    end if

    $\alpha_j = \langle \boldsymbol{w}, \boldsymbol{v}_{cur} \rangle$

    $\boldsymbol{w} = \boldsymbol{w} - \alpha_j\boldsymbol{v}_{cur}$

    $\beta_{j+1} = \|\boldsymbol{w}\|$

    $\boldsymbol{v}_{old} = \boldsymbol{v}_{cur}$

    $\boldsymbol{v}_{cur} = \boldsymbol{w}/\beta_{j+1}$

end for

Set: $\boldsymbol{q} \approx \beta_1 \operatorname{trid}(\beta_2^m, \alpha_1^m, \alpha_2^m)^{-1/2}\boldsymbol{e}_1$ using (4) (here $\boldsymbol{e}_1 = (1, 0, \ldots, 0) \in \mathbb{R}^m$)

$\boldsymbol{v}_{cur} = \boldsymbol{r}_0/\beta_1$, $\boldsymbol{x}_{Krylov} = \boldsymbol{0}$

for $j = 1$ to m (2nd pass) **do**

    $\boldsymbol{x}_{Krylov} = \boldsymbol{x}_{Krylov} + q_j\boldsymbol{v}_{cur}$

    if $j = 1$ **then**

        $\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{v}_{cur} - \boldsymbol{Q}_W(\boldsymbol{W}_Q v_{cur} \oslash \boldsymbol{\lambda})$

    else

        $\boldsymbol{w} = \boldsymbol{Q}\boldsymbol{v}_{cur} - \boldsymbol{Q}_W(\boldsymbol{W}_Q\boldsymbol{v}_{cur} \oslash \boldsymbol{\lambda}) - \beta_j\boldsymbol{v}_{old}$

    end if

    $\boldsymbol{w} = \boldsymbol{w} - \alpha_j\boldsymbol{v}_{cur}$

    $\boldsymbol{v}_{old} = \boldsymbol{v}_{cur}$

    $\boldsymbol{v}_{cur} = \boldsymbol{w}/\beta_{j+1}$

end for

$\boldsymbol{x} = \boldsymbol{W}(\boldsymbol{\lambda}^{-1/2} \odot (\boldsymbol{W}^T\boldsymbol{z})) + \boldsymbol{x}_{Krylov}$

---

where Γ is a curve which encloses the spectrum of $\boldsymbol{Q}$. In our case, we have

$$\boldsymbol{Q}^{-1/2}\boldsymbol{z} = \frac{1}{2\pi i} \oint_\Gamma \zeta^{-1/2}(\zeta\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}d\zeta. \tag{8}$$

We have to make two important choices when approximating this integral: i) which curve Γ to use, ii) what type of quadrature to employ. Davies and Higham (2005) show that direct quadrature is inefficient in the sense that we need an enormous number of quadrature points to achieve good accuracy. For our function, $f(\zeta) = \zeta^{-1/2}$, it is possible to modify (8) and obtain quadrature points which are optimal. A thorough description of this can be found in Hale et al. (2008). We next describe our particular method briefly.

First, observe that $\boldsymbol{Q}^{-1}f(\boldsymbol{Q})\boldsymbol{z} = \frac{1}{2\pi i} \oint f(\zeta)\zeta^{-1}(\zeta\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}d\zeta$, and set $\omega^2 = \zeta$, so that $2\omega d\omega = d\zeta$. We next use $f(\zeta) = \zeta^{1/2}$, and get

$$\boldsymbol{Q}^{-1/2}\boldsymbol{z} = \frac{1}{2\pi i} \oint_{\Gamma_\omega} \omega^{-2}\omega(\omega^2\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}2\omega d\omega$$
$$= \frac{1}{\pi i} \oint_{\Gamma_\omega} (\omega^2\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}d\omega. \tag{9}$$

Since $\boldsymbol{Q}$ has positive real spectrum, we may integrate over the imaginary axis to enclose the spectrum. In essence, this is the contour we integrate over. To choose the quadrature points optimally, Hale et al. (2008) suggest a conformal mapping of the Jacobi elliptic function $\omega = \lambda_{min}^{1/2}\mathrm{sn}(t|k^2)$ from the rectangle $(-K, K) \times (0, K')$ to $\mathbb{R}$, where $\mathrm{sn}(t|k^2)$ adheres to standard notation for elliptic functions (see e.g. Akhiezer (1990)) and the second co-ordinate is the imaginary part. Here $k = (\lambda_{min}/\lambda_{max})^{1/2}$, where $\lambda_{min}, \lambda_{max}$ are the smallest and largest eigenvalues of Q respectively, and K, K' is implicitly determined by k and the logarithm of $\mathrm{sn}(\cdot|k^2)$. In this transformation, quadrature points are sampled evenly on the line $0 \times (0, K')$ in the rectangle $(-K, K) \times (0, K')$. The approximation resulting from using these contours rediscovers a result from Zolotarev concerning optimal rational approximations of $t^{-1/2}$ on defined intervals (see Zolotarev (1877) and Akhiezer (1990)). Using this quadrature, we get an approximation as in (4):

$$\boldsymbol{Q}^{-1/2}\boldsymbol{z} = -\frac{1}{\pi i} \oint_{R_-} (\boldsymbol{Q} - \omega^2\boldsymbol{I})^{-1}\boldsymbol{z}d\omega$$
$$\approx \sum_{j=1}^{N} \alpha_j(\boldsymbol{Q} - \sigma_j\boldsymbol{I})^{-1}\boldsymbol{z} \tag{10}$$

The algorithm requires estimation of the extremal eigenvalues of Q, $\lambda_{min}, \lambda_{max}$. One should underestimate λ_{min} and overestimate λ_{max} to cover the spectrum appropriately in the quadrature. In practice, however, the rational approximations seem to be fairly robust in perturbing $\lambda_{min}, \lambda_{max}$: we tested several approximations, some really coarse, and we did not lose so much in accuracy. Moreover, the number of terms, N, in the rational approximations (4) must be chosen. The number of quadrature points grows logarithmically with the condition number of the precision matrix. A more precise result is the following theorem by Hale et al. (2008), which can also be used to choose the number of quadrature points, N. Let $\boldsymbol{Q}$ be a real or complex matrix with spectrum contained in $[\lambda_{min}, \lambda_{max}]$. Then the rational approximations (4) with coefficients computed by quadrature converge to $\boldsymbol{Q}^{-1/2}$ at the rate

$$\|\boldsymbol{Q}^{-1/2} - f_N(\boldsymbol{Q})\| = \mathcal{O}(e^{\epsilon - 2\pi KN/K'}), \tag{11}$$

for any $\epsilon > 0$ for K, K' defined by the conformal maps above. The constant in the exponent is asymptotically $\pi K'/(2K) \sim 2\pi^2 \log(\lambda_{max}/\lambda_{min})$, as $\lambda_{max}/\lambda_{min} \to \infty$. For any $\lambda_{min}, \lambda_{max} \in \mathbb{R}_+$ we have

$$\|\boldsymbol{Q}^{-1/2} - f_N(\boldsymbol{Q})\| = \mathcal{O}(e^{-2\pi^2 N/(\log(\lambda_{max}/\lambda_{min})+3)}). \tag{12}$$

For a matrix with $\lambda_{max}/\lambda_{min} = 10^4$, which is common in our examples, this yields a convergence rate of $\mathcal{O}(5^{-N})$ for the entire matrix function. In practice, N between 2 and 9 is sufficient. Note that $\|\boldsymbol{Q}^{-1/2}\boldsymbol{z} - f_N(\boldsymbol{Q})\boldsymbol{z}\| \leq \|\boldsymbol{Q}^{-1/2} - f_N(\boldsymbol{Q})\|\|\boldsymbol{z}\|$, and therefore the theorem holds functions of a matrix times a vector as well.

For computing $(\omega^2 \boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{z}$ or equivalently $-(\boldsymbol{Q} - \omega^2\boldsymbol{I})^{-1}\boldsymbol{z}$ to identify ω and σ_i in (4), we employ versions of the conjugate gradient algorithms. Since ω^2 lies on the negative real axis, the conditioning of the system improves and this works as a stabilising agent for the sampling algorithm. When choosing the tolerance for our iterative solvers, we need them to be approximately the error we have in the approximation given by (12), divided by N. This can be regarded as tuning parameters in the algorithms.

The rational approximations have the property that all the systems that need to be solved are shifts of the initial system $\boldsymbol{Qx} = \boldsymbol{z}$. In relation to Krylov methods we have $\mathcal{K}_m(\boldsymbol{Q}, \boldsymbol{r}_0) = \mathcal{K}_m(\boldsymbol{Q} - \sigma_j, \boldsymbol{r}_0)$, for $\sigma_j \in \mathbb{C}$. This property is exploitable and Krylov methods for such shifts are developed in, e.g. van den Eshof and Sleijpen (2003) and Frommer and Simoncini (2008). The computational advantage in employing such a method comes from the fact that the coefficients in the CG algorithm for $\boldsymbol{Q}\sigma_j - \boldsymbol{I}$ can be computed for all the σ_j simultaneously. We pay by storing some additional vectors compared to the classical CG algorithm. We give here a version of CG-M developed in Jegerlehner (1996). We use it for computing rational approximations of $\boldsymbol{Q}^{-1/2}\boldsymbol{z}$. The main advantage of using this strategy is that the cost of producing a sample is essentially the the cost of solving one linear system.

Algorithm 4 CG-M for $\boldsymbol{Q}^{-1/2}\boldsymbol{z}$ according to (4)

Set: $\boldsymbol{r} = \boldsymbol{z}, \boldsymbol{p} = \boldsymbol{r}, \beta_{old} = 1, \alpha = 0, c_{cur} = \langle \boldsymbol{r}, \boldsymbol{r} \rangle, \boldsymbol{x}^\sigma = \boldsymbol{0}, \zeta_{cur}^\sigma = 1, \zeta_{old}^\sigma = 1, p^\sigma = z$

for $j = 1$ to m or hardest system converged **do**

   $\boldsymbol{p}_Q = \boldsymbol{Qp}$

   $\beta_{cur} = -\frac{c_{cur}}{\langle \boldsymbol{p}, \boldsymbol{p}_Q \rangle}$

   for $k = 1$ to n_σ **do**

      $\zeta_{new}^k = \beta_{old}\dfrac{\zeta_{cur}^k\zeta_{old}^k}{\beta_{cur}\alpha(\zeta_{old}^k - \zeta_{cur}^k) + \beta_{old}\zeta_{old}^k(1 - \sigma_k\beta_{cur})}$

      $\beta^k = \beta_{cur}\dfrac{\zeta_{new}^k}{\zeta_{cur}^k}$

      $\boldsymbol{x}^k = \boldsymbol{x}^k - \beta^k\boldsymbol{p}^k$

   end for

   $\boldsymbol{r} = \boldsymbol{r} + \beta_{cur}\boldsymbol{p}_Q$

   $c_{new} = \langle \boldsymbol{r}, \boldsymbol{r} \rangle$

   $\alpha = \frac{c_{new}}{c_{cur}}$

   $\boldsymbol{p} = \boldsymbol{r} + \alpha\boldsymbol{p}$

   for $k = 1$ to n_σ **do**

      $\alpha^k = \alpha\dfrac{\zeta_{new}^k\beta^k}{\beta_{cur}\zeta_{cur}^k}$

      $\boldsymbol{p}^k = \zeta_{new}^k\boldsymbol{r} + \alpha^k\boldsymbol{p}^k$

   end for

   Set: $\zeta_{old}^\sigma = \zeta_{cur}^\sigma, \zeta_{cur}^\sigma = \zeta_{new}^\sigma, \beta_{old} = \beta_{cur}$ and $c_{cur} = c_{new}$

end for

Set: $\boldsymbol{x} = \boldsymbol{0}$

for $k = 1$ to n_σ **do**

   $\boldsymbol{x} = \boldsymbol{x} + w_k\boldsymbol{x}^k$ according to (4)

end for

Note that the shift in Algorithm 4 only appears in the computation of ζ_{new}^σ, and the corresponding coefficients are those that would be obtained from running CG on the shifted systems. Algorithm 3 (with no deflated vectors) and Algorithm 4 are equivalent because of the invariance of Krylov subspaces under shifts. Algorithm 4 has the advantage that no two-pass strategy is required, and it can use a stopping criterion for the CG algorithm. The total number of matrix vector products required for convergence should therefore in theory be half that of two-pass Lanczos - in practice the story is a bit different.

3.3   Continuous deformation method

The continuous deformation method is based on solving the following ODE Allen et al. (2000)

$$d\boldsymbol{z}/dt = r(\boldsymbol{Q} - \boldsymbol{I})(t(\boldsymbol{Q} - \boldsymbol{I}) + \boldsymbol{I})^{-1}\boldsymbol{z}, \ t \in [0, 1], \tag{13}$$

with $\boldsymbol{z}(0) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $r = -1/2$. The solution at the endpoint, $\boldsymbol{z}(1) = \boldsymbol{x}$ is a sample from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}^{-1})$. This can for instance be shown by considering the eigen decomposition of the system as in Allen et al. (2000). Let $\boldsymbol{V}, \boldsymbol{\Lambda}$ be the eigenvectors and eigenvalues of $\boldsymbol{Q}$ respectively, and let further $\boldsymbol{z}(t) = \sum_j \alpha_j(t)\boldsymbol{v}_j$, where $\boldsymbol{v}_i$ is the i'th column of $\boldsymbol{V}$. Inserting this representation in (13) and taking the $i'th$ component gives

$$\frac{d}{dt}\alpha_i(t)\boldsymbol{v}_i = r\alpha_i(t)(\boldsymbol{Q} - \boldsymbol{I})(t(\boldsymbol{Q} - \boldsymbol{I}) + \boldsymbol{I})^{-1}\boldsymbol{v}_i. \tag{14}$$

This corresponds to the equation

$$\frac{d}{dt}\alpha_i(t) = r\alpha_i(t)(\lambda_i - 1)(t(\lambda_i - 1) + 1)^{-1}, \tag{15}$$

with initial condition $\alpha_i(0) = \frac{\boldsymbol{z}(0)^T \boldsymbol{v}_i}{\boldsymbol{v}_i^T \boldsymbol{v}_i}$. The first order ODE above is solved by separation of variables and gives

$$\alpha_i(t) = (1 + t(\lambda_i - 1))^r \frac{\boldsymbol{z}(0)^T \boldsymbol{v}_i}{\boldsymbol{v}_i^T \boldsymbol{v}_i}. \tag{16}$$

Setting $t = 1, r = -1/2$, $\alpha_i(1) = \lambda_i^{-1/2}$ and summing over the $i's$, we get

$$\boldsymbol{z}(1) = \sum_i \lambda_i^{-1/2} \frac{\boldsymbol{z}(0)^T \boldsymbol{v}_i}{\boldsymbol{v}_i^T \boldsymbol{v}_i} \boldsymbol{v}_i = \boldsymbol{Q}^{-1/2}\boldsymbol{z}(0). \tag{17}$$

This ODE, when discretised, leads to rational approximations which are different from those of the previous section, but can be reduced to a form similar to that of (4). Equation (17) shows in an explicit way that we only need to interpolate the inverse square root on the spectrum of $\boldsymbol{Q}$.

An alternative viewpoint comes from looking at a deformation matrix, $\boldsymbol{B}(t) = (1 - t)\boldsymbol{I} + t\boldsymbol{Q}$, take the inverse square root, and differentiate to see that we get the matrix ODE below.

$$\begin{aligned}\frac{d\boldsymbol{B}^{-1/2}}{dt} &= \frac{d}{dt}((1 - t)\boldsymbol{I} + t\boldsymbol{Q})^{-1/2} \\ &= \frac{1}{2}(\boldsymbol{Q} - \boldsymbol{I})\boldsymbol{B}^{-1/2-1}\end{aligned} \tag{18}$$

Projecting the matrix equation onto the start vector, $\boldsymbol{z}(0) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ yields (13).

The critical points for implementing this ODE routine are: i) a good solver for $(t(\boldsymbol{Q} - \boldsymbol{I}) + \boldsymbol{I})\boldsymbol{b} = \boldsymbol{z}$ for all $t \in [0, 1]$ and ii) an appropriate ODE solver. We use Krylov methods (CG) to solve $(t(\boldsymbol{Q} - \boldsymbol{I}) + \boldsymbol{I})\boldsymbol{b} = \boldsymbol{z}$ at every time step. The ODE is solved by matlabs ODE45 discretisation scheme. Natural tuning/accuracy parameters for this method are relative and absolute tolerances in the ODE-solver. If the matrix $\boldsymbol{Q}$ is badly conditioned, the ODE (13) becomes stiff. This can slow down the ODE solver for time steps close to $t = 1$ in our implementation, but is partially overcome by ODE45's adaptive timestepping. An advantage of this implementation over that of the previous section is, however, that the extremal eigenvalues of $\boldsymbol{Q}$ need not be estimated.

A natural extension of this method comes from looking at the class of ODEs defined by

$$\boldsymbol{z}'(t) = r(\boldsymbol{Q} - \boldsymbol{I})(\boldsymbol{I} + g(t)(\boldsymbol{Q} - \boldsymbol{I}))^{-1}g'(t)\boldsymbol{z}(t), \tag{19}$$

with the constraints $g \in C^1[0, 1]$, $g(0) = 0, g(1) = 1$ and $r = -1/2$. This may lead to better performance for some systems. Two examples are $g(t) = \frac{\ln(t+1)}{\ln(2)}$ and $g(t) = t^2$.

Table 1: Timings in seconds, random structure precision matrices

	8^3, $\kappa = 5.4$	16^3, $\kappa = 10.3$	32^3, $\kappa = 12.9$	64^3, $\kappa = 15.3$	128^3, $\kappa = 27.3$
CHOL	$3.57 \cdot 10^{-4}$	$3.29 \cdot 10^{-2}$	1.71	N/A	N/A
SEC-CG	$1.36 \cdot 10^{-3}$	$9.19 \cdot 10^{-3}$	$1.10 \cdot 10^{-1}$	1.08	30.1
RAT-CG-M	$1.11 \cdot 10^{-3}$	$4.70 \cdot 10^{-3}$	$4.54 \cdot 10^{-2}$	0.496	15.5
2pLANC	$4.52 \cdot 10^{-4}$	$3.40 \cdot 10^{-3}$	$4.10 \cdot 10^{-2}$	0.458	16.4
CONT-D	$3.20 \cdot 10^{-2}$	$3.71 \cdot 10^{-1}$	2.78	30.7	$6.01 \cdot 10^2$
CU-CG-M	$2.47 \cdot 10^{-3}$	$2.83 \cdot 10^{-3}$	$8.08 \cdot 10^{-3}$	$6.99 \cdot 10^{-2}$	0.743

Table 2: Matrix vector products, random structure precision matrices

	8^3, $\kappa = 5.4$	16^3, $\kappa = 10.3$	32^3, $\kappa = 12.9$	64^3, $\kappa = 15.3$	128^3, $\kappa = 27.3$
SEC-CG	4	5	5	5	5
RAT-CG-M	4	6	5	6	7
2pLANC	2×3	2×4	2×4	2×4	2×4
CONT-D	~ 100	~ 100	~ 80	~ 100	~ 100
CU-CG-M	7	7	8	8	8

3.4 GPU implementiation

The advent of CUDA by the nVidia corporation gives us the possibility to implement massively parallel algorithms on the GPU in "high level languages". It is natural to see if we can get speedup using such massively parallel computing hardware. In statistics, GPU implementations of Monte Carlo algorithms have been successful, see e.g. Lee et al. (2009). The basic idea using CUDA as an entry point for using such hardware is to have a built-in fine grain independent structure in the computations and then assign threads to these computations automatically using special code constructs.

For Krylov methods, the needed ingredient to implement the presented algorithms is a sparse matrix vector multiplication, $\boldsymbol{Qz}$. The cublas library by nVidia provides fast dense matrix operations. For sparse matrix vector products, we use the `cusp-library`, which is a further development of the work of Bell and Garland (2009) made available through google code. Both hardware and compilers have evolved since that point.

The above mentioned cusp library has a CG-M implementation available. With some work, it is possible to modify the CG-M code to facilitate the rational approximations in (4) - we have coded these modifications. The possible performance gain then essentially comes from faster matrix vector products and inner products of vectors. The performance of this CUDA implementation is presented in the examples below.

4 Examples

In Section 4.1 we present a random precision matrix model inspired by a space-time application of infectious disease count (see Held and Paul (2009)). In Section 4.2 and 4.3 we consider an application with seismic data. Section 4.2 is a comparative study of the computation time, while Section 4.3 samples elastic model parameters given seismic 3D reflection data from a North Sea reservoir.

The different algorithms we test are:

- **CHOL:** Cholesky sampling.

- **SEC-CG:** Sequential use of CG on each term in (4) using the coefficients developed in Section 3.2.

- **P-SEC-CG:** Preconditioned version of SEC-CG using a circulant preconditioner from a circulant approximation of the precision matrix.

- **RAT-CG-M:** Algorithm 4.

- **2pLANC:** 2-pass Lanczos sampling.

- **DEF-2pLANC:** The deflated version of 2-pass Lanczos, Algorithm 3.

- **COND-D:** The continuous deformation method.

- **CU-CG-M:** The GPU implementation, using CUDA, of algorithm RAT-CG-M.

In all our comparisons, we have used a relative tolerance of 0.005, which also defines our stopping criterion.

There is one important aspect that must not be ignored when comparing the algorithms: are the comparisons fair? That is to say, is, for instance, one of the algorithms favoured in implementation compared to the others? We have tried to implement the algorithms on equal grounds, but the deformation method (Section 3.3) may have quite superior implementations using a better ODE-solver for the problem at hand. We have chosen to use the fairly standard ODE45 solver (existing in Matlab).

Table 3: Timings for D-2pLANC, random structure precision matrix. The first column indicates number of deflated vectors.

	8^3	16^3	32^3	64^3
5	$4.91 \cdot 10^{-4}$	$4.81 \cdot 10^{-3}$	$6.78 \cdot 10^{-2}$	0.726
10	$5.02 \cdot 10^{-4}$	$4.96 \cdot 10^{-3}$	$7.44 \cdot 10^{-2}$	0.775
15	$5.15 \cdot 10^{-4}$	$5.17 \cdot 10^{-3}$	$8.15 \cdot 10^{-2}$	0.849
20	$5.27 \cdot 10^{-4}$	$5.41 \cdot 10^{-3}$	$8.91 \cdot 10^{-2}$	0.916

Lastly, we use an alternative, non-standard criterion for convergence in the 2pLANC methods, namely we look at a large number of samples and see what dimension of the Krylov subspace is needed to make a sample converge on average and use this as a fixed m in Algorithm 2.

All the timings presented in the following sections are given in seconds, and the abbreviations given in the list above are also used in the tables.

On top of Table 1, 2, 4, 5, 7 and 8, $\kappa = \lambda_{max}/\lambda_{min}$ denotes the condition number of the corresponding matrix. It is a well known fact that the number of matrix-vector products required for a Krylov is dependent on the condition number of a matrix; in fact, the following bound holds for the CG algorithm (Saad (2003))

$$\|x - x_m\| \le 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \|x - x_0\| \tag{20}$$

where m is the dimension of the Krylov subspace. Hence, as κ grows, $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1) \to 1$ and convergence is slow.

4.1 Random pattern precision matrices

The random pattern precision matrices of this section are generated by the following heuristic algorithm:

1. Pick a random entry, $i, j \in \{1, \ldots, n\}$

2. Add to $Q(i,j), Q(j,i)$ a realisation of $\mathcal{N}(0,1)$

3. Add this realisation to the diagonal of Q

4. Loop until enough non-zero entries.

5. Assert that Q is positive definite using a Hadamard criterion. If not, add another random entry.

This gives rise to an unstructured matrix. A similar unstructured pattern may emerge from using a non-standard spatio-temporal model (Held and Paul (2009)), where the spread of disease is simulated based on the neighborhood pattern obtained from airline routes across the world. An illustration of a matrix with such sparsity pattern can be found on the right in Figure 1. We generate matrices of size $n^3 \times n^3$ for $n = 8, 16, 32, 64, 128$ in order to have comparable results with the matrices in Section 4.2 and also with a comparable amount of non-zero entries. For these type of matrices, circulant preconditioners are inappropriate since the $\boldsymbol{Q}$ is completely

Table 4: Timings in seconds, prior seismic precision matrices

	8^3, $\kappa = 2.3 \cdot 10^2$	16^3, $\kappa = 2.9 \cdot 10^2$	32^3, $\kappa = 5.6 \cdot 10^6$	64^3, $\kappa = 1.5 \cdot 10^4$
CHOL	$1.20 \cdot 10^{-4}$	$7.55 \cdot 10^{-2}$	1.19	N/A
SEC-CG	$5.90 \cdot 10^{-3}$	0.364	41.9	23.9
P-SEC-CG	$4.01 \cdot 10^{-3}$	$4.70 \cdot 10^{-2}$	1.34	6.08
2pLANC	$1.40 \cdot 10^{-3}$	0.228	35.5	19.7
RAT-CG-M	$4.27 \cdot 10^{-3}$	0.300	26.5	16.6
CU-CG-M	$9.21 \cdot 10^{-3}$	0.175	2.24	0.502

Table 5: Matrix vector products, prior seismic precision matrices

	8^3, $\kappa = 2.3 \cdot 10^2$	16^3, $\kappa = 2.9 \cdot 10^2$	32^3, $\kappa = 5.6 \cdot 10^6$	64^3, $\kappa = 1.5 \cdot 10^4$
SEC-CG	25	448	3990	273
P-SEC-CG	12	41	132	58
2pLANC	2×15	2×268	2×1880	2×151
RAT-CG-M	25	610	4498	344
CU-CG-M	25	591	4386	311

non-stationary, and hence P-SEC-CG is not included in the comparison. Incidentally, the matrices constructed by this method are extremely ill-suited for Cholesky factorisations as the amount of fill-in (even after reordering) is enormous. Additionally, by construction, the condition number of a particular matrix is independent of the dimension. This makes these matrices particularly well suited for Krylov methods.

In Table 1 the timings of the different methods are displayed, while Table 2 shows the number of matrix vector products needed for the iterative methods. The number of matrix vector products for CONT-D is approximate, as our implementation does not allow for exact counts. An entry N/A means that the memory requirements are larger than 8Gb. We use a separate table for D-2pLANC with different degrees of deflation. This is sumarised in Table 3. The number of matrix vector products is the same as for 2pLANC in Table 2.

The timings given in Table 1 show that in low dimensions, i.e. 8^3 and 16^3, the sampling method is not particularly important. We get samples fast and at a comparable rate whatever method we choose. Note, however, that even in dimensions 8^3, the timings of all the Krylov methods are comparable to that of Cholesky sampling. As has been mentioned before, these random matrices are particularily ill-suited for Cholesky sampling, but nonetheless, if this structure information is available a priori, choosing a Krylov method seems very reasonable. The scaling of timings is much better using Krylov methods, which can be seen in the last three columns in Table 1. We believe this is mainly due to the approximate invariance of the condition number of the matrices as the number of dimensions increase.

The effects of RAT-CG-M requiring only one pass of the matrix vector products are seen in the last column, since the cost of the matrix vector products compared with the other operations increase more with dimension. This is also reflected in Table 2, comparing 2pLANC to RAT-CG-M.

The GPU-implementation of RAT-CG-M, namely CU-CG-M, shows different degrees of speedup/-down depending on the size of Q. For 8^3, CU-CG-M performs worse than the bulk of algorithms, but from 16^3 and up, we have different degrees of speedup; from a speedup of 1.3x in dimensions 16^3 to a speedup of 20.9x in dimensions 128^3. We believe this can be explained by the increasing importance of fast matrix-vector products as we increase the dimensions of the precision matrix.

For deflation, the results are summarised in Table 3. It appears that deflation is not a good choice for this particular type of matrices, and the most natural explanation is that the small eigenvalues of Q cluster together in a relative sense.

The CONT-D method compares unfavourably to the others, but one could possibly improve this by using a more favourable ODE-solvers for the equation (13). It is also one of the methods for the (inverse) square-root mentioned in Higham (2008).

Figure 1: Structure of posterior seismic- (left) and random structure (right) precision matrices
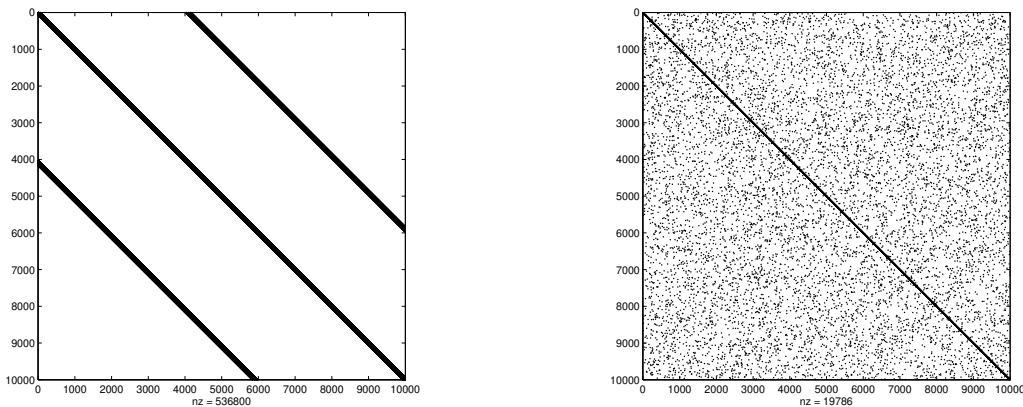
Table 7: Timings in seconds, posterior seismic precision matrices

	8^3, $\kappa = 4.5 \cdot 10$	16^3, $\kappa = 1.6 \cdot 10^2$	32^3, $\kappa = 3.7 \cdot 10^3$	64^3, $\kappa = 2.7 \cdot 10^3$
CHOL	$1.20 \cdot 10^{-4}$	$7.41 \cdot 10^{-2}$	1.13	N/A
SEC-CG	$3.72 \cdot 10^{-3}$	$3.59 \cdot 10^{-2}$	1.79	18.1
P-SEC-CG	$3.60 \cdot 10^{-3}$	$2.83 \cdot 10^{-2}$	0.918	13.2
2pLANC	$1.11 \cdot 10^{-3}$	$1.10 \cdot 10^{-2}$	0.736	8.12
RAT-CG-M	$4.67 \cdot 10^{-3}$	$1.19 \cdot 10^{-2}$	1.16	10.3
CU-CG-M	$4.91 \cdot 10^{-3}$	$1.10 \cdot 10^{-2}$	$9.50 \cdot 10^{-2}$	0.431

Table 6: Timings for D-2pLANC, seismic prior. The first column indicates number of deflated vectors.

	16^3	32^3	64^3
5	0.211	23.7	22.8
10	0.157	20.2	25.4
15	0.140	18.6	28.0
20	0.138	19.5	29.4
30	0.114	16.9	N/A

4.2 Seismic prior/posterior precision structures

Seismic data play an extremely important role in the exploration for oil and gas resources. The inversion of seismic reflection data to elastic parameters in transversely isotropic media is a well studied problem. The basic physical model is governed by the Zoepritz equations, see e.g. Stovas and Ursin (2003). We consider a linear approximation of the Zoeppritz equations (Buland and Omre (2003) and Rabben et al. (2008)). More precisely,

Table 8: Matrix vector products, posterior seismic precision matrices

	8^3, $\kappa = 4.5 \cdot 10$	16^3, $\kappa = 1.6 \cdot 10^2$	32^3, $\kappa = 3.7 \cdot 10^3$	64^3, $\kappa = 2.7 \cdot 10^3$
SEC-CG	25	70	193	186
P-SEC-CG	11	19	74	99
2pLANC	2×9	2×17	2×67	2×65
RAT-CG-M	17	30	155	141
CU-CG-M	15	29	152	137

15

for reflection angle θ, and north, east and depth reference (i, j, k), we have the following model

$$y_p = f(\theta, x(i, j, k)) = \frac{1}{2\cos^2\theta_P} \frac{\triangle I_P}{\overline{I}_P}(i, j, k) -$$

$$4\sin^2\theta_S \frac{\triangle I_S}{\overline{I}_S}(i, j, k)$$

$$-\frac{1}{2}\tan^2\theta_P \left(1 - 4\gamma^2(k)\cos^2\theta_P\right) \frac{\triangle\rho}{\overline{\rho}}(i, j, k), \tag{21}$$

where $\overset{\triangle}{\div}$ denotes relative change in the corresponding elastic P-impedance I_P, S-impedance I_S and density ρ, and γ is a background (average) v_P/v_S-trend. The response is a convolved signal of the physical reflections y_p, and Buland and Omre (2003) defined the following statistical model to describe the data at one angle:

$$y = W(Ax + \epsilon_1) + \epsilon_2, \tag{22}$$

with $\epsilon_1 \sim \mathcal{N}(\mathbf{0}, c_1 Q_1^{-1})$, $\epsilon_2 \sim \mathcal{N}(\mathbf{0}, c_2 Q_2^{-1})$ and $x \sim \mathcal{N}(\boldsymbol{\mu}, Q^{-1})$. Here, the coefficients in A are obtained using (21), and the matrix W contains the convolution model. We assume that all precision matrices Q, Q_1 and Q_2 are sparse Markov.

In this section, we will use a simplified version of (22) to compare the different sampling algorithms. We do an inversion on real seismic data from the North Sea in the next section. The simplification does not impose any change in the 3D spatial correlation, but includes a zero mean, uses only one of the elastic parameters and only one reflection angle θ. This simplification can be obtained directly from the full model by using only $\theta = 0$, so that we only are given information on P-impedances.

The matrices Q_1, Q are constructed as follows: We use an exponential correlation function and optimise for parameters in a $3 \times 3 \times 3$-neighbourhood in the Markov graph for Q. For exponential correlation, this approximation is very good (Rue and Tjelmeland (2002)). Note, however, that this choice is of minor importance when it comes to relative performance between the sampling procedures. We have used an effective correlation length of 10 cells. We may alternatively choose our parameters freely in a different way if we have convenient procedures for doing so. Q_1 is a diagonal matrix with linearly decreasing precision with depth. Since we are dealing with a field of size $n_x \times n_y \times n_t$, where n_x, n_y are lateral coordinates and n_t is a depth coordinate, this linear decrease comes in diagonal blocks of size n_t embedded in the larger matrix Q_1. Choosing $Q_2 = I$ gives the posterior precision matrix for the simplified model

$$Q_{post} = Q + Q_{lik}. \tag{23}$$

with $Q_{lik} = (c_1 W Q_{lik}^{-1} W^T + c_2 I)^{-1}$.

Sampling results for sampling from Q and Q_{post} as in (23) are given in Table 4,6 and 7. In all these cases, the sampling methods based on rational approximations and Krylov methods work really well. The structure of this posterior precision matrix can be found on the left in Figure 1. The prior precision matrix Q has similar structure to that of the posterior precision matrix, but has a narrower band close to the diagonal due to the exclusion of the convolution.

In Table 4, we see that Cholesky sampling remains competitive until it is impossible to do it due to memory constraints. This occurs between 32^3 and 64^3 in our model. The band 3D band structure makes Cholesky sampling a bit more forgiving than the structure of the matrices in the previous section. While Cholesky sampling is faster in all cases except the 16^3 case, we suspect that if Cholesky sampling was possible in the 64^3 case, it would perform relatively worse as the conditioning of the matrix improved in that case.

At this point, a comment regarding the conditioning of the 64^3 matrix is in place. We clearly see the condition number for the 64^3 matrix is better than that of the 32^3 matrix, and this explains why both the timing is better for the 64^3 case and that the number of matrix-vector product is less.

The Krylov methods with no preconditioning are very comparable to each other, with RAT-CG-M having an edge in higher dimensions, as expected. The prior precision matrix is by design close to circulant, and that is why we see a massive improvement in the P-SEC-CG row. Note that P-SEC-CG computes the solution of several linear systems and RAT-CG-M essentially only computes the solution of one. If we have a good preconditioner that can be extended to shifts, it is natural to use it, but implementation and parallelism can be major issues.

Figure 2: Vertical (top) and horisontal (bottom) slice of Norne data (left) and inverted mean (right)
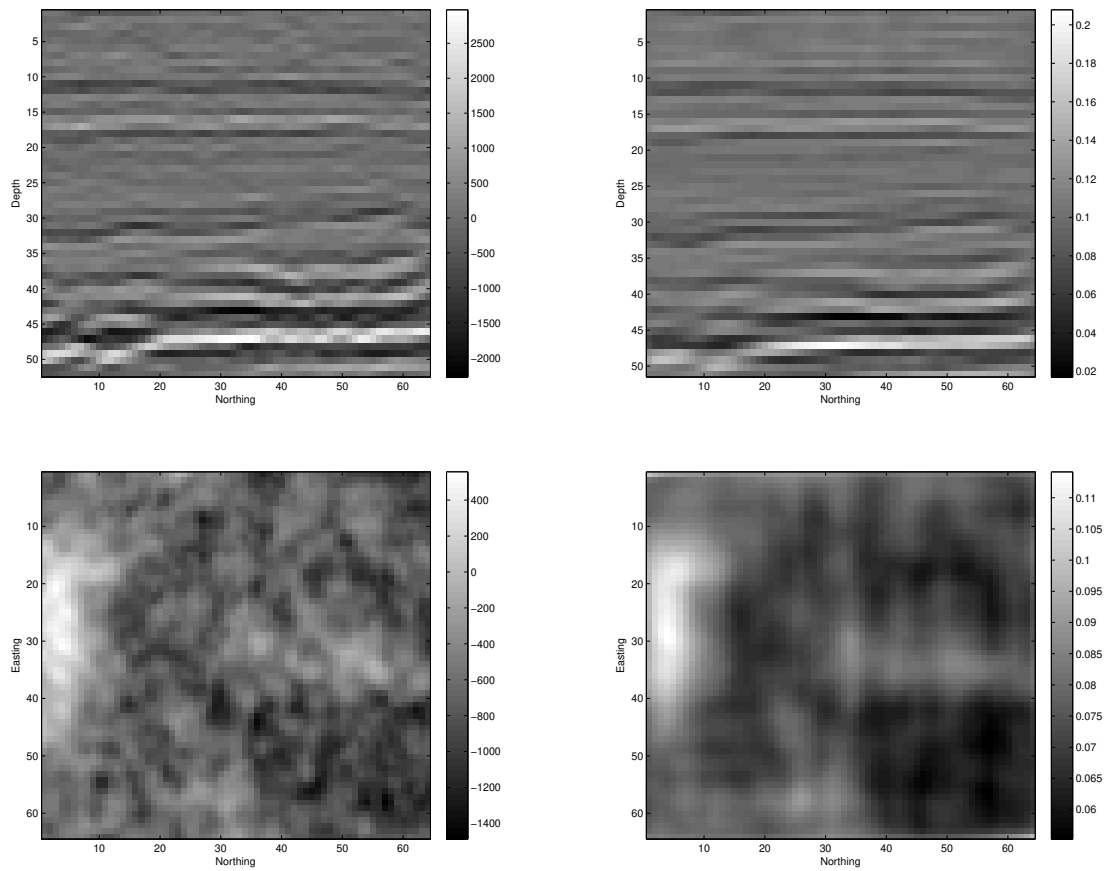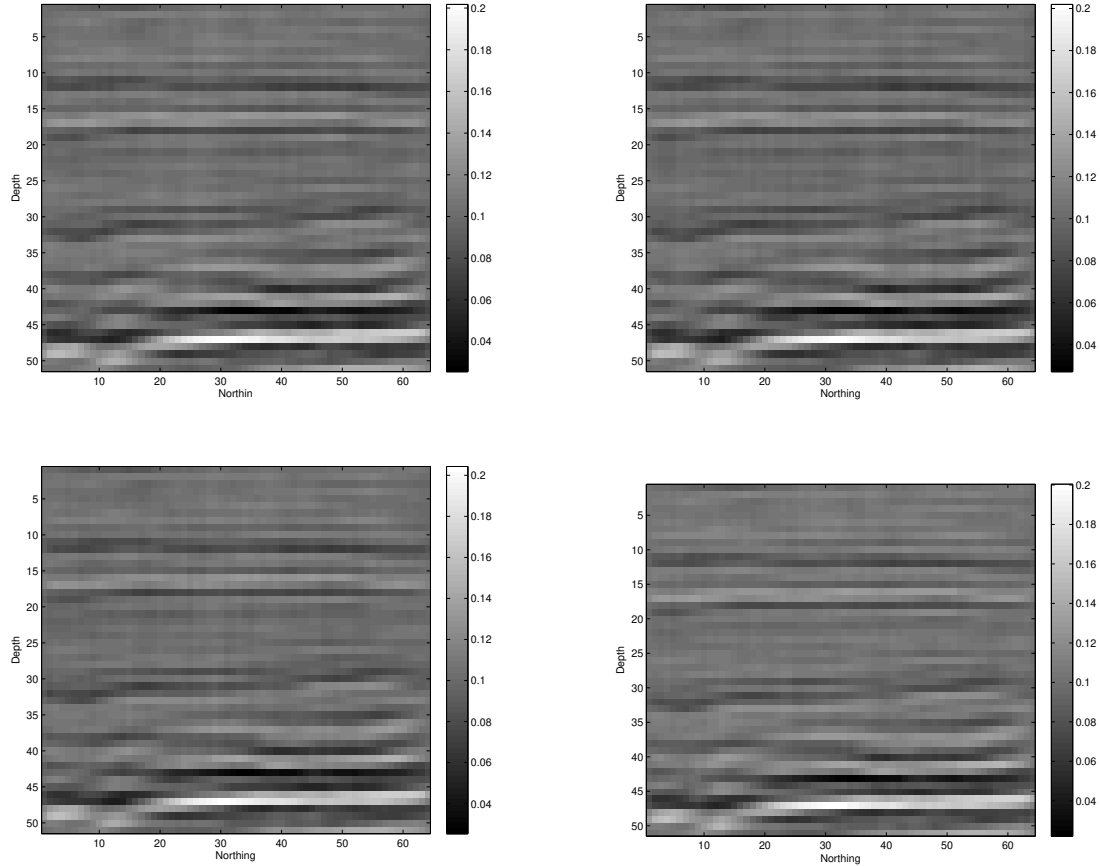
Figure 3: Samples from the posterior, vertical slice, Norne

Optimally, we could imagine a preconditionen RAG-CG-M, but this may be very difficult to obtain in practice. The D-2pLANC perform favourably for both 16^3 and 32^3 dimensions in this example.

For the GPU implementation, the story is a bit different from that of the random precision structure matrices. We do not have a clear improvement over all other methods before the 64^3 case. CU-CG-M remains competitive, however, with both CHOL and P-SEC-CG in all cases except in dimensions 8^3, and in dimensions 64^3 it gives a speedup of 12.1x over P-SEC-CG and a overwhelming speedup 33.1x over the similar CPU implementation, RAT-CG-M. In Table 7 for the posterior precision matrices, the condition numbers of the matrices are a bit better and we have more fill-in due to bandwith increase. Here Cholesky sampling quickly falls behind compared to the Krylov methods. We also see that the preconditioned version does not offer as much of an improvement as in the prior precision counterpart, and this is caused by the strong deviation from stationarity incurred by the likelihood. Additionally, the potential condition number improvement is not as huge as in the prior case.

One counter-intuitive result is that 2pLANC performs better that RAT-CG-M. This may be, as hinted in the introduction of Section 4, related to the different convergence criteria.

The CU-CG-M starts to outperform the other methods in dimensions 16^3, where it performs exactly as good as 2pLANC, and the speedup increases to 18.8x over 2pLANC. Not as good as for the prior precision matrices, but still a massive performance boost.

The question of whether we should deflate approximate eigenvectors (or other vectors) or not does not have an obvious answer. Comparing the results in Table 6 and Table 3, we see that in one case, deflating is really a good idea, while in the other, it hampers the performance of the sampling procedure. Heuristically speaking, there are two reasons to deflate vectors; one is increasing the performance of the sampling procedure, the other is the following: suppose we have a region of interest $U \subset D$, where D is the domain for the sampling, and we need more accurate sampling results in that region. Then we may deflate some orthogonal basis vectors pertaining to that region. The first of these two is the more natural, and in this one it is possible to address the question on whether we should deflate or not. In the article Saad et al. (1999) a detailed analysis on how deflating is related

18

Figure 4: Samples from the posterior, horisontal slice, Norne
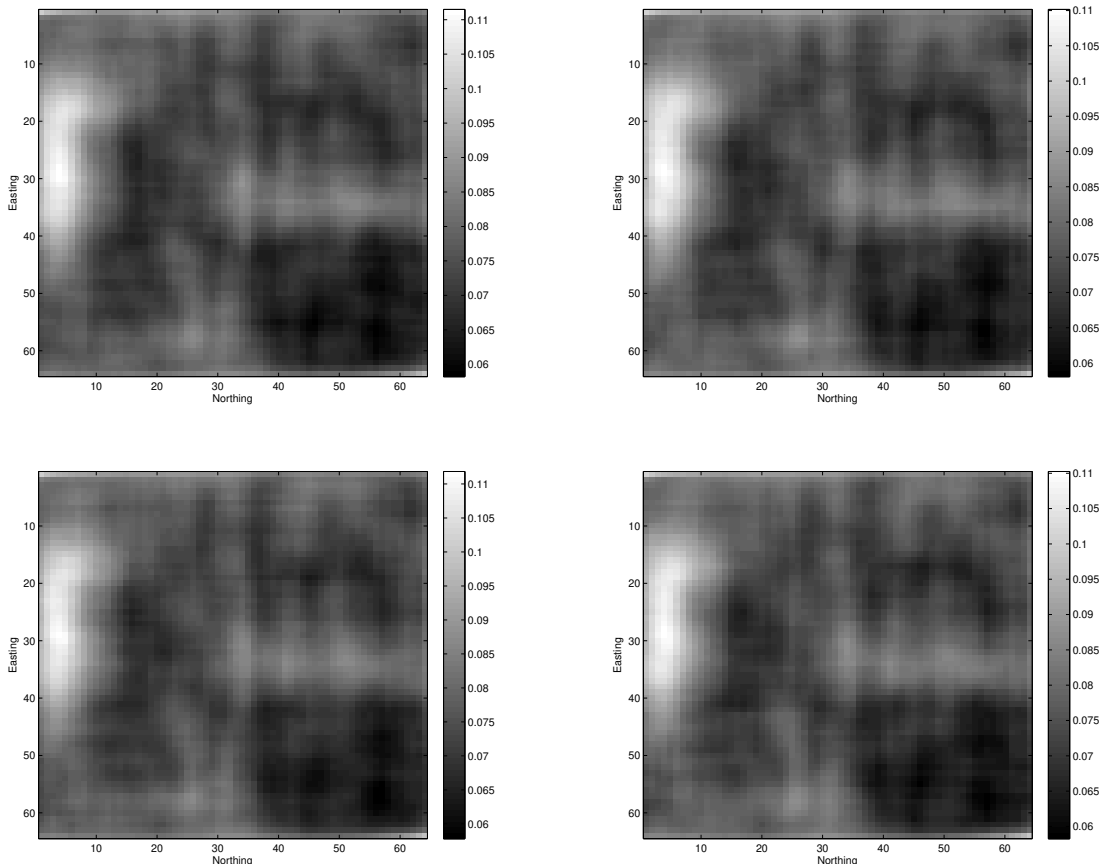
to preconditioning is presented. So suppose that $K_m(\boldsymbol{Q}, \boldsymbol{r}_0) \perp \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_s$ for eigenvectors $\boldsymbol{w}_i$. Then whether we should deflate a new vector, $\boldsymbol{w}_{(i+1)}$, depends on the associated fraction of eigenvalues $\kappa(i+1) = \lambda_{(i+1)}/\lambda_{(i)}$, where $\lambda_{(i)} = \min(\lambda_i | \lambda_i \in \sigma(\boldsymbol{Q}) - \{\bigcup_{k=1}^{i-1} \lambda_{(k)}\})$, where $\sigma(\boldsymbol{Q})$ is the set of eigenvalues of $\boldsymbol{Q}$. Note that $\kappa(i+1) \geq 1$, and if $\kappa(i+1)$ is large enough, we deflate $w_{(i+1)}$. "Large enough" is dependent on the implementation of the algorithm, specifically, the cost of dot products and populating vectors. In practice, it is easy to deflate more and more vectors, so we stop as soon as they are difficult to compute or the performance gain is trivial.

## 4.3   Inversion of Norne-data and sampling from the posterior

In this section we will look at inversion of seismic data, and sampling from its posterior. The model is the same as in Section 4.2, but here we include the full version of the $\boldsymbol{A}$-matrix in (22), and we assume correlation between the elastic parameters in the prior model. It is straight forward to construct $\boldsymbol{Q}$ in this situation: we have $\boldsymbol{Q} := \boldsymbol{Q} \otimes \boldsymbol{Q}_0$, where $\boldsymbol{Q}_0$ is the 3×3 precision matrix for the elastic parameters. The relevant least squares problem, then becomes

$$(\boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{Q}_{lik} \boldsymbol{W} \boldsymbol{A} + c_1 \boldsymbol{Q}) E(\boldsymbol{x}|\boldsymbol{y}) = c_1 \boldsymbol{Q} \boldsymbol{\mu} + \boldsymbol{A}^T \boldsymbol{W}^T \boldsymbol{Q}_1 \boldsymbol{y}. \tag{24}$$

We solve for $E(\boldsymbol{x}|\boldsymbol{y})$ here, and we sample from the posterior precision matrix given on the left side of (24). The data we consider is from the Norne field in the North Sea. It consists of seismic reflection data gathered in three angles of resolution and on a 3D grid of size $111 \times 111 \times 510$. We take the slice $111 \times 111 \times 128$ and resample it to $64 \times 64 \times 64$ in order to fit the posterior matrix in memory. Alternatively, a routine for each matrix vector product may be constructed and applied in sequence. A typical vertical slice of this data y along with its inverted acoustic impedance can be visualised as in Figure 2.

The parameters in the prior mean $\boldsymbol{\mu}$, precision matrices $\boldsymbol{Q}_1$ and $\boldsymbol{Q}$, for the γ-parameter in the $\boldsymbol{A}$-matrix and the convolution model ($\boldsymbol{W}$) are typically assigned from auxiliary data. This consists of well log information from

19

Figure 5: Posterior means with different prior levels, vertical (left) and horisontal (right)
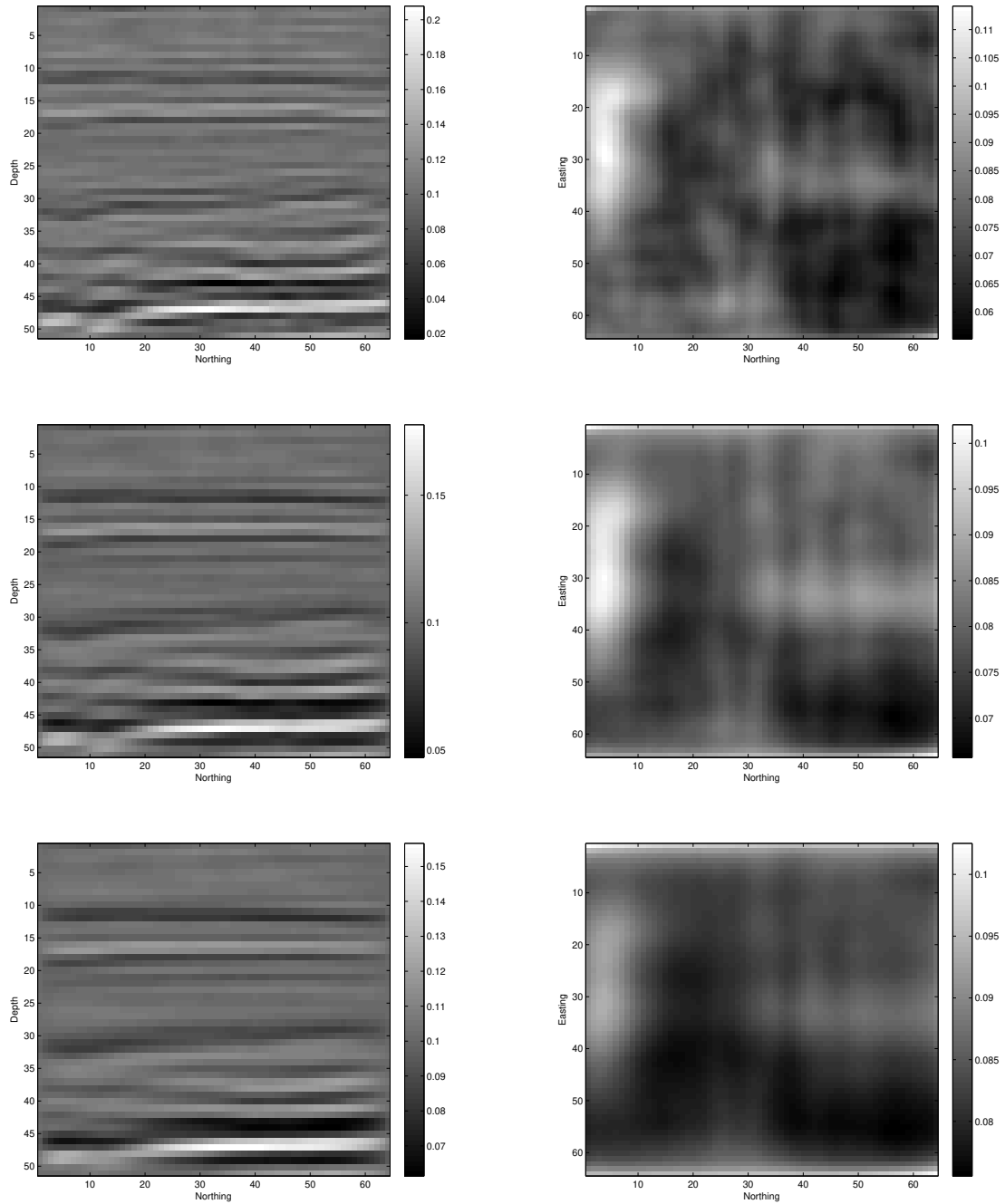
neighbouring reservoirs, lab measurements used to build geophysical relationships, and geological knowledge of the subsurface. Well logs are used to specify many of the prior parameters within the context of a geological depositional environment. Moreover, the well observations constitute relatively perfect observations of the reservoir properties as compared with the seismic data. Thus, a well log and seismic data at the same location are used to assess the seismic likelihood parameters, within the modeling assumptions defined from years of geophysical lab experiments. The large angle seismic data is noisier than that at small angles. Also, the noise level increases as a function of depth. The spatial correlation parameters are tuned from geological modeling.

For interpretation purposes, it can be argued that it is better to look at an ensemble of samples instead of only the inverted mean; what if there are regions with features that can be significantly perturbed for interpretation purposes that only show up in some of the samples from the posterior? This is valuable information for the contractor and should be present in evaluation of assets. Four samples from the posterior are given in Figure 3. In our example the horizontal slice in Figure 4 are a bit more perturbed than the lateral one, but there are no huge differences on the scale we are looking at here. To see the effect of the prior, we have included inversion results with different prior levels in Figure 5. This figure shows that as the prior level increases, we have more smoothing and hence more boundary effects, and less dependency on the data, as expected.

The timings for these samples are comparable to that of Section 4.2. Howewer, in the multivariate setting, the number of non-zero entries in the posterior precision matrix is approximaterly 10 times that of the posterior generated in the previous section. In our implementation the sampling takes about 5 minutes. This computation time is too large to attempt MCMC solutions, but is useful for visualizing an ensemble of seismic inversion results.

5   Discussion

In this article we have looked at several algorithms stemming from combinations of Krylov subspace methods combined with rational approximations for Gaussians in which it is easy to either compute the matrix vector product of its precision matrix or its covariance matrix. In particular, we have found that these methods are not prohibitively expensive in problems in which traditional Cholesky sampling is infeasible, and that the methods compare relatively well with Cholesky sampling in lower dimensions. In addition, we have seen that a considerable speedup in employing these methods is possible by utilising the GPU. This may lead to possibilities of sampling based inference in higher dimensional problems than have previously been considered for problems in which Gaussian proposal distributions are natural.

6   Acknowledgments

References

Akhiezer, N. I. (1990). *Elements of the Theory of Elliptic Functions*. American Mathematical Society.

Allen, E. J., Baglama, J., and Boyd, S. K. (2000). Numerical approximation of the product of the square root of a matrix with a vector. *Linear Algebra and its Applications*, 310:167–181.

Banerjee, S., Gelfand, A. E., Finley, A., and Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society, Series B*, 70:209–226.

Belabbas, M. and Wolfe, P. (2009). Spectral methods in machine learning and new strategies for very large datasets. *Proceedings of the National Academy of Sciences*, 106(2):369.

Bell, N. and Garland, M. (2009). Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–11, New York, NY, USA. ACM.

Besag, J., York, J., and Mollie, A. (1991). Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43:1–59.

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

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

Cressie, N. and Johannesson, G. (2008). Fixed rank kriging for large spatial datasets. *Journal of the Royal Statistical Society, Series B*, 70:209–226.

Davies, P. I. and Higham, N. J. (2005). *QCD and Numerical Analysis III*, chapter Computing f(A)b for Matrix Functions f, pages 15–24. Springer-Verlag.

Fox, C. (2008). Conjugate direction sampling. Technical report, University of Otago, New Zealand.

Frommer, A. and Simoncini, V. (2008). *Model Order Reduction: Theory, Research Aspects and Applications*, chapter Matrix functions, pages 275–304. Springer.

Golub, G. H. and van Loan, C. F. (1996). *Matrix Computations, 3rd Ed.* John Hopkins University Press.

Gray, R. (2006). *Toeplitz and circulant matrices: A review.* E-book.

Hale, N., Higham, N. J., and Trefethen, L. N. (2008). Computing A^{α}, $\log(A)$ and related matrix functions by contour integrals. *SIAM Journal of Numerical Analysis*, 46:2505–2523.

Held, L. and Paul, M. (2009). Predictive validation of a non-linear model with random effects for infectious disease counts. Technical report, Institute of Social and Preventive Medicine, University of Zurich, Switzerland.

Hestenes, M. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436.

Higham, N. J. (2008). *Functions of Matrices: Theory and Computation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Ilic, M., Turner, I. W., and Simpson, D. P. (2009). A restarted lanczos approximation to functions of a symmetric matrix. *IMA Journal of Numerical Analysis Advance Access.*

Jegerlehner, B. (1996). Krylov space solvers for shifted linear systems. *arXiv.org, arXiv:hep-lat/9612014v1*, NA:NA.

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2009). On the utility of graphics cards to perform massively parallel simulation of advanced monte carlo methods. *arXiv.org, arXiv:0905.2441v3*, NA:NA.

Lehoucq, R. B., Sorensen, D. C., and Yang, C. (1998). *ARPACK, user's guide.* SIAM.

Rabben, T. E., Ursin, B., and Tjelmeland, H. (2008). Non-linear bayesian joint inversion of seismic reflection coefficients. *Geophysical journal international*, 173:265–280.

Rasmussen, C. and Wiliams, C. (2006). *Gaussian processes for machine learning.* MIT Press, MA.

Roberts, G. and Sahu, S. (1997). Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(2):291–317.

Rue, H. (2001). Fast sampling of gaussian markov random fields. *Journal of the Royal Statistical Society, Series B*, 63:325–338.

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields.* Chapman & Hall.

Rue, H. and Tjelmeland, H. (2002). Fitting gaussian markov random fields to gaussian fields. *Scandinavian Journal of Statistics*, 29:31–49.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems, 2nd Ed.* SIAM.

Saad, Y., Yeung, M., Erhel, J., and Guyomarc'h, F. (1999). A deflated version of the conjugate gradient algorithm. *SIAM Journal of Scientific Computing*, 21:1909–1926.

Schneider, M. and Willsky, A. (2003). A krylov subspace method for covariance approximation and simulation of random processes and fields. *Multidimensional systems and signal processing*, 14:295–318.

Simpson, D. (2008). *Krylov subspace methods for approximating functions of symmetric positive definite matrices with applications to applied statistics and anomalous diffusion.* PhD thesis, School of Mathematical Sciences, Queensland Univ of Tech.

Simpson, D., Turner, I., and Pettitt, A. (2007). Fast sampling from a gaussian markov random field using krylov suspace approaches. Technical report, School of Mathematical Sciences, Queensland Univ of Tech.

Stein, E. M. and Shakarchi, R. (2003). *Complex Analysis.* Princeton University Press.

Stovas, A. and Ursin, B. (2003). Reflection and transmission responses of layered transversely isotropic viscoelastic media. *Geophysical Prospecting*, 51:447–477.

Trefethen, L. and Bau, D. (1997). *Numerical linear algebra.* SIAM Publications, Philadelphia, PA.

van den Eshof, J. and Sleijpen, G. (2003). Accurate conjugate gradients methods for families of shifted systems. *Applied Numerical Mathematics*, 49:17–37.

Zolotarev, E. I. (1877). Applications of elliptic functions to questions of functions deviating least and most from zero. *Zap. Imp. Nauk St. Petersburg*, 30:xx.