Erlend Aune

Computation and modeling for high dimensional Gaussian distributions

Thesis for the degree of Philosophiae Doctor

Trondheim, October 1, 2012

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

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

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

ISBN 978-82-471-3884-7 (printed version)
ISBN 978-82-471-3885-4 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2012:285

Printed by Skipnes Kommunikasjon as

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree Philosophiae Doctor (PhD) at the Norwegian University of Science (NTNU) and Technology. The research has been funded by the Uncertainty in reservoir evaluation (URE) project. The project started September 2008, and ended August 2012.

When I first started working on this thesis, four years ago, I knew little about statistics. My master's degree was in harmonic analysis, dealing with the uncertainty principle and Gabor frames. It quickly became clear to me that my take on the challenges that appeared would be different from those having an a priori solid fundament in the statistical sciences. While it took a while to get used to the jargon and methods in statistics, I believe it was truly beneficial for me to change fields, learning new things and taking a different approach to thinking about what the real problems are.

First, thank you to all my collaborators and discussion partners for helpful input when doing the research found in the thesis. Special thanks is warranted to the following individuals:

I would like to thank my primary supervisor Jo Eidsvik – especially for being so open minded about the thesis' content, letting me follow my own ideas and pace in developing them. Without such an open minded supervisor, I believe the thesis would be something completely different, and that, in my opinion, for the worse. Of course, his scientific input has also been of great value for the work in this thesis.

Secondly, I thank my close friend and collaborator Daniel (Dan) Simpson for many fruitful discussions on the use of Krylov methods for matrix functions and for sharing his myriads of ideas on interesting topics worth exploring. In many ways, he and I were outsiders in the statistical group at NTNU these years, having quite a different background from the others.

I also thank my secondary supervisor, Bjørn Ursin, for being patient about the AVA inversion article and for giving sound advise on the topics therein.

The last 1.5 years at the office have been made more fun having my office mate Kjartan Rimstad around, especially with the introduction of the concept "Friday at 1026."

Lastly (but evidently not leastly), I thank my girlfriend, Sigrid, for being patient about my way of dealing with things and encouraging me when feeling exhausted after a hard day at the office.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The introductory chapter in this thesis contains basic results concerning the Gaussian distribution and fundamentals of the theory of matrix functions and how they relate to the Gaussian distribution, as well as some applications. We will refer to other parts of the thesis as required, but the introduction should for the most part be a self-contained piece of text. Throughout the thesis, necessary information is repeated as needed. We conclude the introductory part by giving a succinct account for the main contribution of the thesis.

Before getting into the details on the different aspects, we say a few words about the motivation for the research we have done: In many ways, spatial statistics and graphical models are in need of new computational frameworks. The dimensionality of the data that needs to be analysed in modern statistical problems increases faster than the computational power available on a single workstation, and in order to do inference in a particular framework, say, maximum likelihood, the individual procedures needed to compute a likelihood- or gradient estimate need to accommodate this. In particular, the algorithms needed to do inference should be cloud-computing friendly through interfaces such as MapReduce (Dean and Ghemawat, 2008). The first part of this thesis, namely Chapter 2 and 3 deals with parallel-friendly aspects of computing for the Gaussian distribution. The first one deals

with sampling from high-dimensional Gaussian distributions, needed for sampling based inference, and the second deals primarily with the hard part of likelihood-based inference in Gaussian models – computation of the log-determinant.

Another aspect that is of vital importance when designing prior distributions is flexibility of including prior information from different sources, and how to do that with ease. Multivariate spatial models, in particular, quickly become high-dimensional, and we need to specify priors that are possible to store and that treats different regions in space differently. In Chapter 5, we present a framework for dealing with different aspects of a Gaussian prior using systems of linear stochastic partial differential equations (SPDEs) driven by Gaussian white noise, which induces a Markovian model with nice properties.

For non-linear problems that are driven with Gaussian noise, many challenges appear, including multi-modality, fast convergence and performance gain over a linearised model. For seismic amplitude versus angle (AVA) inversion, such a problem is treated in Chapter 4, with heuristics to overcome some of these challenges.

1.1 The Gaussian distribution

The Gaussian distribution is probably the most known probability distribution in existence. The reason for this stems not only from its wide use in statistics, but also from aspects of different branches of analysis, where it pops up as parts of a solution to a PDE, or as being an eigenfunction of the Fourier transform. Every person who has studied mathematics is bound to have encountered this distribution in one form or another. A real-valued random variable, x, has a Gaussian distribution if

$$p(x|\sigma^2, \mu) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}. \tag{1.1}$$

The parameters μ, σ^2 are the expectation and the variance of the variable respectively. The definition is easily extended to $\mathbf{R}^d$-valued variables as follows: the variable $\mathbf{x}$ has a *multivariate* Gaussian distribution if

$$p(\mathbf{x}|\mathbf{Q}, \boldsymbol{\mu}) = \frac{\det(\mathbf{Q})^{1/2}}{(2\pi)^{d/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{Q}(\mathbf{x}-\boldsymbol{\mu})}. \tag{1.2}$$

Here, the precision matrix, $\mathbf{Q}$, is symmetric positive definite and $\boldsymbol{\mu}$ is the expectation vector. The precision matrix is the inverse of the covariance matrix, i.e. $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$. If $\mathbf{Q}$ has some eigenvalues that are zero, (1.2) does not define a proper density, since the "distribution" for $\mathbf{x}$ is invariant under addition of linear combination of the corresponding eigenvectors. We call variables having such a "distribution" intrinsic Gaussian variables. It should be possible to formulate a proper distribution for equivalence classes of variables that are shifted by linear combinations of eigenvectors, but we do not address this topic here.

It is possible to go farther than this, defining continuously indexed Gaussian *fields*, and even Gaussian measures. A random field $x : \Omega \to \mathbb{R}^T$ is a Gaussian field if for all finite subsets $\{k_i\}_{i=1}^N \subset T$, $\mathbf{x} = (x(k_1), \ldots, x(k_N))^T$ has a multivariate Gaussian distribution. Here, Ω is the probability space, and T is the index family (typically $\mathbb{R}^n$). Gaussian fields are uniquely identified through their expectation function $\mu(t) = \mathbb{E}(x(t))$ and covariance function $C(s,t) = \mathbb{E}([x(s) - \mu(s)][x(t) - \mu(t)])$. A thorough treatment of the theory of Gaussian fields is found in Adler and Taylor (2007), and the treatment of Gaussian measures is found in Bogachev (1998). In this thesis, we will only concern ourselves with finite dimensional Gaussian distributions and finite dimensional, discretized representations of Gaussian random fields.

1.1.1 Properties of the Gaussian distribution

There are several properties that inhabits the Gaussian distribution that makes it attractive for the prospective user, ranging from its asymptotic properties and the central limit theorem to its closure properties, and, of

course, its computational properties. We start by having a look at the closure properties of the multivariate Gaussian distribution. We give them without proof, but they are easily obtained and can be found in any introductory book on the multivariate Gaussian distribution. In particular, most of them are found in Rue and Held (2005), where they are formulated in terms of the precision matrix $\mathbf{Q}$ rather than the covariance matrix, wherever this is appropriate.

Theorem 1. *Suppose that*

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{pmatrix}^{-1} \right), \tag{1.3}$$

and let $\mathbf{A}$ *be any matrix compatible with* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)^T$*, then*

$$\mathbf{y} = \mathbf{A}\mathbf{x} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}, \mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^T) \tag{1.4}$$

$$\mathbf{x}_1 | \mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\mu}_1 - \mathbf{Q}_{11}^{-1}\mathbf{Q}_{12}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \mathbf{Q}_{11}^{-1}) \tag{1.5}$$

$$x^j \sim \mathcal{N}(\mu^j, (\mathbf{Q}^{-1})_{jj}), \tag{1.6}$$

where the superscript indicates the component of the corresponding vector. Moreover, if $\mathbf{H}\mathbf{H}^T = \mathbf{Q}$*, then if* $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$*,* $\mathbf{x} = \mathbf{H}^{-T}\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$*.*

In words, the Gaussian distribution is closed under linear combinations, conditioning and marginalisation. Moreover, the precision matrices remain fixed for any computation, only requiring multiplications and inversions of itself or another fixed matrix. A natural consequence of the preceding theorem is prediction in the Gauss-linear model, where observations are noisy versions of a linear combination – i.e.

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}, \tag{1.7}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_\epsilon)$. The goal in the Gauss-linear model is to infer $\mathbf{x}|\mathbf{y}$. Both the posterior expectation and precision are easily obtained by noting that

$$\mathrm{Cov}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}^{-1} & \mathbf{Q}^{-1}\mathbf{A}^T \\ \mathbf{A}\mathbf{Q}^{-1} & \mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^T + \mathbf{Q}_\epsilon^{-1} \end{pmatrix}, \tag{1.8}$$

and by block inversion of this covariance matrix, we obtain the corresponding precision matrix;

$$\begin{pmatrix} \mathbf{Q}^{-1} & \mathbf{Q}^{-1}\mathbf{A}^T \\ \mathbf{A}\mathbf{Q}^{-1} & \mathbf{A}\mathbf{Q}^{-1}\mathbf{A}^T + \mathbf{Q}_\epsilon^{-1} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{Q} + \mathbf{A}^T\mathbf{Q}_\epsilon\mathbf{A} & -\mathbf{A}^T\mathbf{Q}_\epsilon \\ -\mathbf{Q}_\epsilon\mathbf{A} & \mathbf{Q}_\epsilon \end{pmatrix}. \quad (1.9)$$

Hence we obtain the posterior expectation as

$$\mathbb{E}(\mathbf{x}|\mathbf{y}) = (\mathbf{Q} + \mathbf{A}^T\mathbf{Q}_\epsilon\mathbf{A})^{-1}\mathbf{x}_{\text{mod}}, \quad (1.10)$$

with $\mathbf{x}_{\text{mod}} = \mathbf{Q}\boldsymbol{\mu} + \mathbf{A}^T\mathbf{Q}_\epsilon\mathbf{y}$. This expectation is, in other words, some weighted mean of observations and the prior mean.

1.1.2 Introducing hyperparameters

If all precision matrices and observation processes did not depend on hyperparameters, the last part of the previous section would contain all parts needed for doing inference in the Gauss-linear model. This assumption is unrealistic, and the matrices $\mathbf{Q}, \mathbf{Q}_\epsilon, \mathbf{A}$, may very well depend on hyperparameters, say $\boldsymbol{\eta}$. In this situation, the entire likelihood must be considered in order to do maximum likelihood estimation of $\boldsymbol{\eta}$. The expression reads

$$p(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\mu}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\mu})\, p(\mathbf{x}|\boldsymbol{\eta}, \boldsymbol{\mu})}{p(\mathbf{x}|\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\mu})}, \quad (1.11)$$

and in the Bayesian setting, where we have a prior for $\boldsymbol{\eta}$ and $\boldsymbol{\mu}$,

$$p(\boldsymbol{\eta}|\mathbf{y}, \boldsymbol{\mu}) \propto \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\mu})\, p(\mathbf{x}|\boldsymbol{\eta}, \boldsymbol{\mu})\, p(\boldsymbol{\eta}, \boldsymbol{\mu})}{p(\mathbf{x}|\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\mu})}. \quad (1.12)$$

In order to find the maximum likelihood estimate for $\boldsymbol{\eta}$, we need to optimise the log-likelihood or posterior for $\boldsymbol{\eta}$. In the Gauss-linear case, the conditional distributions on the right hand side of the preceding case are Gaussian. The

log-posterior reads

$$\begin{aligned} 2\log p(\boldsymbol{\eta}|\mathbf{y},\boldsymbol{\mu}) &= C + \log p(\mathbf{y}|\mathbf{x},\boldsymbol{\eta},\boldsymbol{\mu}) + \log(p(\mathbf{x}|\boldsymbol{\eta},\boldsymbol{\mu}) + \\ &\quad \log p(\boldsymbol{\eta},\boldsymbol{\mu}) - \log p(\mathbf{x}|\mathbf{y},\boldsymbol{\eta},\boldsymbol{\mu}) \\ &= C + \log\det \mathbf{Q}_{\epsilon,\boldsymbol{\eta}} - (\mathbf{y} - \mathbf{A}_{\boldsymbol{\eta}}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}})^T \mathbf{Q}_{\epsilon,\boldsymbol{\eta}}(\mathbf{y} - \mathbf{A}_{\boldsymbol{\eta}}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}) + \\ &\quad \log\det \mathbf{Q}_{\boldsymbol{\eta}} - (\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} - \boldsymbol{\mu})^T \mathbf{Q}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} - \boldsymbol{\mu}) \\ &\quad - \log\det(\mathbf{Q}_{\boldsymbol{\eta}} + (\mathbf{A}_{\boldsymbol{\eta}})^T \mathbf{Q}_{\epsilon,\boldsymbol{\eta}}\mathbf{A}_{\boldsymbol{\eta}}) + 2\log p(\boldsymbol{\eta},\boldsymbol{\mu}), \end{aligned} \tag{1.13}$$

where the $\boldsymbol{\eta}$ subscript denotes dependence on the hyper-parameters. Sometimes the hyper-parameters comes in conjugate forms, leading to explicit distributions for $\boldsymbol{\eta}|\mathbf{y},\boldsymbol{\mu}$, but most of the time, we need to evaluate the expression (1.13). Often, this expression simplifies, and everything that simplifies is worth looking into, as optimising this expression is the most computationally intensive problem in solving the Gauss-linear model.

Finding the uncertainty for the parameters $\boldsymbol{\eta}$ can be done by using Taylor expansions, and fitting a density with respect to the derivatives at the optimum. Typically a Gaussian is used, requiring only the Hessian at the estimated $\boldsymbol{\eta}$, and in the literature, this is called the Laplace approximation or Laplace's method (see, e.g. Tierney and Kadane (1986)). It is also possible to do MCMC, but this may be very costly, as it requires the evaluation of the ratio of two likelihoods in each iteration. For small models, this may be feasible.

One of the main contribution of this thesis deals with the approximation of this likelihood or posterior in the case where all precision matrices are sparse, and the Gaussian distributions have large dimensions. This is the theme for Chapter 3. Here, we also present some alternative approximations that may be suitable in specific cases.

It may happen that the observation process is non-linear, i.e. $\mathbf{y} = \mathbf{A}(\mathbf{x}) + \boldsymbol{\epsilon}$, but the error- and prior process remains Gaussian. In this case, many of the properties of the computational parts for the Gaussian distribution may be retained with small modifications, and approximate inference is quite feasible through non-linear least squares methods, such as Gauss' method

(Kelley, 1999). Here approximate means that the distribution for $\mathbf{x}|\mathbf{y}$ is not known explicitly. In Chapter 4, one such model, dealing with seismic AVA inversion, is treated in detail.

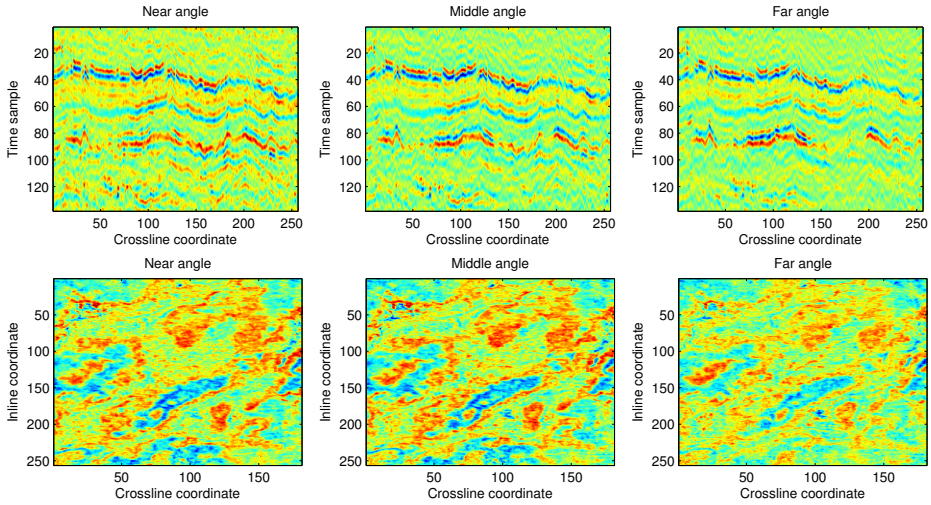
1.1.3 Applications of the Gauss-linear model

Applications of the Gauss-linear model is abundant in the literature, ranging from the most trivial linear regression found in any introductory statistics course to huge climate reconstruction models. We mention a few here, and focus on problems in high dimensions as this is keeping in line with the theme of the thesis.

An archetypical application of the Gauss-linear model in a really high dimensional setting is the seismic AVA inversion problem (Buland and Omre, 2003; Buland et al., 2003). Here, data is collected on 2-D grid or 3-D cube, and the number of data points can almost be arbitrarily large. In 2-D, the dimensionality ranges from the 10 000s to several 100 000s, and in 3-D from 100 000s up to about 10^9 data points, if all data is considered. The linear forward model takes elastic parameters, such as velocities and density, to reflection coefficients and observations. The prior from the elastic parameters can be specified through a covariance function, as in Buland and Omre (2003) and Buland et al. (2003), but when the model is non-stationary, this quickly becomes computationally infeasible, since fast Fourier transform-type (FFT-type) inversion models are not applicable. Possible remedies for this is explored in Chapter 5, where flexible Markovian models are studied specifically for this problem. A non-linear forward model of the top reservoir with Gaussian noise assumptions have been studied in Rabben and Ursin (2011), Rabben and Ursin (2009) and Rabben et al. (2008), and the numerically much more challenging 3-D problem is studied in Chapter 4. In Figure 1.1, typical inline-crossline and traveltime-crossline data is depicted. In this case, the dimensionality is about 10^6.

Reconstruction of global ozone levels from scattered satellite data is another high dimensional problem suited to be treated Gauss-linearly. Here, the

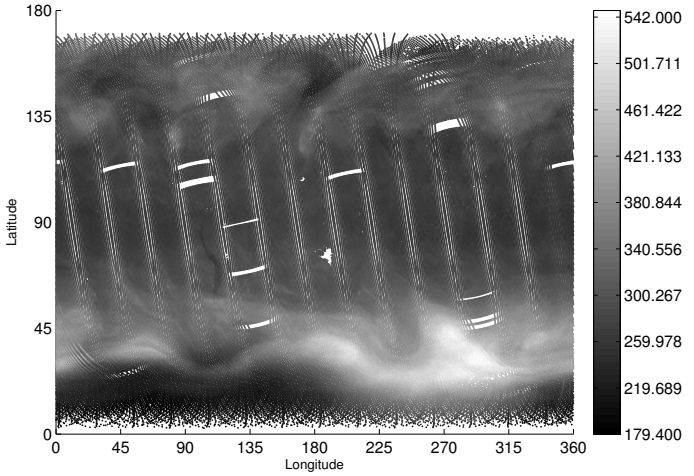
Figure 1.1: *Illustration of seismic data. Traveltime vs. crossline (top) for different angles, inline-crossline for different angles (bottom)*



satellite gathers date along its trajectory, yielding transects of ozone observations, and the goal is to reconstruct ozone level at arbitrary locations. This data, as it is publicly available, has become popular in the statistics literature for demonstrating methods for estimation, and has been treated in, e.g., Cressie and Johannesson (2008), Eidsvik et al. (2011) and Bolin and Lindgren (2011), where the two treats computational aspects and the last deals with sound statistical modeling. In Chapter 3, this example is used for illustrating the methodology presented therein. We note that the ozone data at one date, i.e. a snapshot, can easily be treated by usual methodology for estimating hyper-parameters. Proper reconstruction where the time dimension is also considered, is another story. In this case, alternative methodologies, like the one presented in Chapter 3 or in Eidsvik et al. (2011) is likely required to do proper inference. In Figure 1.2, a snapshot over ozone levels for one day is given.

In agricultural statistics, an important challenge is to determine how much

Figure 1.2: *Ozone observations for one day, given in Dobson units*



traits are influenced by inheritance, and how much by external properties, such as living conditions, seasonal effects, and so on. In Mrode and Thompson (2005), a detailed account for linear models for predicting breeding values is given, and in Gorjanc (2010) a graphical model representation for pedigree based mixed models is presented. The important aspect is that the "prior" precision matrix for inheritance can be constructed explicitly by using pedigree graph for the individuals in question. The number of individuals in these models can be extremely large, ranging from tens of thousands to tens of millions. The community treating such problems on a regular basis seems to lack methodologies for treating the largest models properly, and this thesis addresses some of the challenges they face.

1.2 Gaussian Markov random fields

In the previous section we discussed the general properties of the Gaussian distribution. Many models do, however, have a Markovian property, meaning that the distribution at some location is only dependent on its neighbourhood, where in this setting, neighbourhood must be understood in a rather wide sense. For Gaussian fields indexed by $\mathbb{R}^d$, there is a theorem classifying all Gaussian Markov random field (GMRF), and it reads as follows: A Gaussian random field has the continuous Markov property if its spectral density can be written as $\frac{1}{\sum_j a_j (2\pi i \xi)^j}$ – i.e. one divided by a polynomial, with the restriction that the polynomial is symmetric and positive. What this means is that all GMRFs in the continuous sense can essentially defined through stochastic differential equations. This theorem among others is proved in Rozanov (1977). Now, discretization of these operators typically leads to sparse precision matrices, but the Markovian property in the discrete sense can be described even simpler. Here, we define a GMRF as a multivariate Gaussian distribution having a sparse precision matrix, i.e. that most entries of the precision matrix $\mathbf{Q}$ are zero. For row i, let $\mathcal{I}$ denote the non-zero indices of $\mathbf{Q}_{i,:}$, and let $\mathbf{x}_{\mathcal{I}} \subset \mathbf{x}$ be the subvector containing these indices, then $x_i \perp x_j | \mathbf{x}_{\mathcal{I}} \iff Q_{ij} = 0$. This is the Markov property. One advantage of having a sparse precision matrix is that storing it takes much less space in memory using a sparse storage scheme, such as compressed column sorted rows, its transpose or coordinates of the non-zero entries stored in two vectors and its corresponding values in a third. Another advantage is that computation using sparse matrix techniques usually is much faster than the corresponding ones for dense matrices. One of the main foci of this thesis is this computational part.

There is some ambiguity of language when speaking of GMRFs, and we will try to clarify this here. First, we note that any sparse precision matrix defines a GMRF. There are, however, occasions where we will talk about a d-dimensional GMRF. In this situation, it is not the dimensionality of the precision matrix, but rather that we are dealing with a discretized version

of a d-dimensional GMRF in the continuous sense. The dimensionality of the precision matrix should be clear from context if it is important, or suppressed otherwise. We never use d-dimensional GMRF in the sense that we are dealing with a sparse $\mathbf{Q} \in \mathbb{R}^{d \times d}$. If the term high dimensional GMRF is used, however, it is usually the precision matrix that is considered.

A flexible method for constructing discrete Gaussian Markov random field is through discretization of linear stochastic PDEs,

$$Lx = \mathcal{W}, \tag{1.14}$$

with appropriate boundary conditions, where L is a linear differential operator satisfying some conditions ensuring non-degeneracy, and $\mathcal{W}$ is spatial white noise. In Lindgren et al. (2011), they explored the use of the fractional SPDE

$$(\kappa^2 - \triangle)^{\alpha/2} x = \mathcal{W} \tag{1.15}$$

using Neumann boundary conditions, noting the link between (1.15) and the Matérn fields (Matérn, 1960). This has been very successful due to its incorporation of the inference package INLA (Rue et al., 2009). In this thesis, we will say that any solution of (1.15) is a Matérn field, even if the induced covariance function do not satisfy the traditional stationary covariance function defined in Matérn (1960).

A particular class of SPDEs that is flexible and nice to work with are second order elliptic SPDEs, possibly extended to (fractional) powers of them. I.e.,

$$Lx = \sum_{i,j=1}^{n} (\partial_{s_i}(a^{ij}(\mathbf{s}))x)\partial_{s_j} x + \sum_{i=1}^{n} b^i(\mathbf{s})\partial_{s_i} x + c(\mathbf{s})x, \tag{1.16}$$

with the uniform ellipticity condition

$$\exists C > 0 \text{ s.t.} \forall \boldsymbol{\xi} \in \mathbb{R}^n, \ \sum_{i,j=1}^{n} a^{ij}(\mathbf{s})\xi_i \xi_j \geq C\|\xi\|_2^2, \tag{1.17}$$

for all $\mathbf{s}$ in the bounded domain of interest. This includes (1.15) and its extension to the anisotropic case

$$(\kappa^2 - \nabla \mathbf{H} \nabla)^{\alpha/2} x = \mathcal{W}. \tag{1.18}$$

Apart from the computational benefits arising from these equations generating sparse precision matrices, they also have more flexibility in treating boundaries in a physically sound way than covariance function models, and it is easy incorporate local behaviour that corresponds to nature. An extension of these ideas in the multivariate setting is treated in Chapter 5. Here the goal is to parametrise the local dependence of multivariate fields in an interpretable way that can facilitate discrete structures with different correlation structures between fields.

1.2.1 The Cholesky decomposition

The Cholesky decomposition of the precision matrix or the covariance matrix is one of the main tools for doing computation when using the Gaussian distribution, in particular in the Gauss-linear model and some extensions of it. For a positive definite matrix, $\mathbf{Q}$, the Cholesky decomposition is the unique lower triangular matrix, $\mathbf{L}$ with positive diagonal elements such that $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$. Such a factorisation also exists for positive semi-definite matrices. The reason why the Cholesky decomposition is such a popular tool for inference is that once you have a fast and robust implementation of an algorithm computing $\mathbf{L}$, it is usually easy to

- Sample $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$, by setting $\mathbf{x} = \mathbf{L}^{-T}\mathbf{z} + \boldsymbol{\mu}$
- Compute the marginal variances $(\mathbf{Q}^{-1})_{ii}$ for all i by recursion (Rue and Martino, 2007)
- Compute $\log\det \mathbf{Q} = 2 \cdot \sum_i \log L_{ii}$
- Compute $\frac{\partial}{\partial \eta_j} \log\det \mathbf{Q}(\boldsymbol{\eta})$ by combining analytic formulae and recursion

The last point obviously depends on how the parameter $\boldsymbol{\eta}$ influences $\mathbf{Q}$. Hence inference is, in principle, easy. For dense precision- or covariance matrices, the computation of $\mathbf{L}$ quickly becomes too expensive with growing dimensions, as computing it takes $n^3/3$ operations. Algorithm 1 is one straight-forward way to compute the Cholesky decomposition.

Algorithm 1 Computing the Cholesky factor of the precision matrix, $\mathbf{Q}$

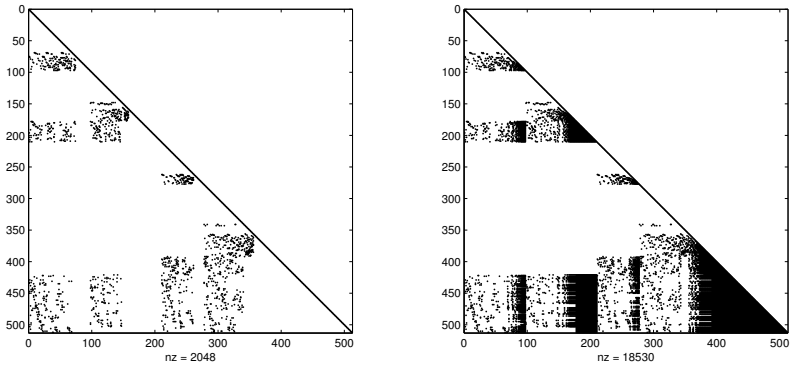
Input: $\mathbf{Q}$
Output: $\mathbf{L}$ such that $\mathbf{L}\mathbf{L}^T = \mathbf{Q}$
Set:
for $j = 1$ to n **do**
  $L_{jj} = \sqrt{Q_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}$
  $L_{ij} = \frac{1}{L_{jj}} \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right)$
end for

For GMRFs the precision matrix is sparse, and it is possible to exploit this when computing the Cholesky factors: if $\mathbf{Q}$ is sparse, $\mathbf{L}$ is often also sparse, and storage- and computational costs remain only a fraction of those required when using non-sparse matrices. In general, however, if $\mathbf{Q}$ is sparse, its Cholesky factor does not need to be sparse. An example is the matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $A_{11} = 2n + 1, A_{ii} = 3, i \neq 1$ and $\mathbf{A}_{2:n,1} = 1, \mathbf{A}_{1,2:n} = 1$. The lower Cholesky triangle of this matrix is full. Using the inverse ordering of this matrix, sending $i \mapsto n - i + 1$, its Cholesky factor becomes sparse, which suggests that good re-orderings may help in producing sparse Cholesky factors in some cases.

The most common library for computing these sparse Cholesky factors is CHOLMOD (Chen et al., 2008), and it is indeed extremely fast, and it serves as a black-box routine in many libraries used for statistical inference, e.g. INLA (Rue et al., 2009).

Now, suppose that we are dealing a d-dimensional GMRF and that $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a precision matrix defined by discretizing this field in some (discretely) Markovian way. Then, if $d = 1$, the fill-in is $\mathcal{O}(n)$, if $d = 2$, fill-in is

Figure 1.3: *Non-zero pattern of (the lower triangular part of)* $\mathbf{Q}$ *(left) and its sparse Cholesky factor (right)*



$\mathcal{O}(n\log(n))$ and computational effort $\mathcal{O}(n^{3/2})$. For $d = 3$, fill-in is approximately $\mathcal{O}(n^{3/2})$ and computational effort $\mathcal{O}(n^2)$ (Liu and George, 1981). Here the *fill-in* is defined as follows: Let $\text{nnz}(\mathbf{M})$ be the number of non-zero entries of a matrix $\mathbf{M}$. Then the fill-in is given by $(\text{nnz}(\mathbf{Q}) + n)/2 - \text{nnz}(\mathbf{L})$. The general trend is, the higher the d, the harder it is to reduce fill-in and computational costs when computing $\mathbf{L}$. For general graphs, it is hard to say anything, and in these situations, it is prudent to try different reordering types and use the one that produces least fill-in. We also note that the reordering typically needs to be computed only once, and hence it is worth spending the time to find a good one if n is large. The concept of fill-in is illustrated in Figure 1.3.

In the following chapters, we study what we can do if even after reordering the fill-in is so large that we cannot store the Cholesky factor in memory. It may also happen that it indeed is possible to store it, but the computational efforts required for solving linear systems are too large to remain efficient. In these cases, we present some alternatives that may be useful for some problems, but not necessarily for all.

1.3 Matrix functions

The theory of matrix functions is a huge area in mathematics, and it comes up in many applications. In particular, any linear ODE has a solution that can be expressed by matrix functions. And for PDEs, exponential integrators (Hochbruck et al., 1998) is a popular technique for approximating the solution. For Gaussian distributions, many matrix functions are also highly relevant, but before going into the details, we introduce the general definition of matrix functions and some of their properties. For an excellent treatise on matrix function and their properties, we recommend the book by Higham (2008).

Any square matrix, $\mathbf{M}$, can be decomposed into the Jordan canonical form as follows

$$\mathbf{M} = \mathbf{V} \begin{pmatrix} \mathbf{J}_1 & & & \\ & \mathbf{J}_2 & & \\ & & \ddots & \\ & & & \mathbf{J}_N \end{pmatrix} \mathbf{V}^{-1}, \tag{1.19}$$

where $\mathbf{V}$ is a matrix containing as columns the eigenvectors of $\mathbf{M}$ and $\mathbf{J}_i$ is the Jordan block associated with the eigenvalue λ_i defined by

$$\mathbf{J}_i = \begin{pmatrix} \lambda_i & 1 & & & \\ & \lambda_i & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & \lambda_i \end{pmatrix}. \tag{1.20}$$

If $\mathbf{J}_i \in \mathbb{R}^{1\times 1}$, the Jordan block consists simply of the eigenvalue λ_i.

For any $(m-1)$-times differentiable function, f, we define its variation on a

Jordan block of size m by

$$f(\mathbf{J}_i) = \begin{pmatrix} f(\lambda_i) & f'(\lambda_i) & \cdots & \cdots & \frac{f^{m-1}(\lambda_i)}{(m-1)!} \\ & f(\lambda_i) & f'(\lambda_i) & \cdots & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & f'(\lambda_i) \\ & & & & f(\lambda_i) \end{pmatrix} \tag{1.21}$$

This induces the definition for the function f of $\mathbf{M}$ as follows

$$f(\mathbf{M}) = \mathbf{V} \begin{pmatrix} f(\mathbf{J}_1) & & & \\ & f(\mathbf{J}_2) & & \\ & & \ddots & \\ & & & f(\mathbf{J}_N) \end{pmatrix} \mathbf{V}^{-1}. \tag{1.22}$$

If two Jordan blocks have equal eigenvalues, $\lambda_i = \lambda_j$, then we must have the equality $f(\lambda_i) = f(\lambda_j)$. This means that we deal with an equation for which $\mathrm{card} f^{-1}(\{\lambda_i\}) > 1$, we must choose one solution and stick with it. Using this restriction, we now have the definition of the primary matrix function. For some functions it is possible to make other definitions: for instance, for a square root of a matrix $\mathbf{M}$ can be defined as any matrix $\mathbf{N}$ such that $\mathbf{N}^2 = \mathbf{M}$, or even more generally, as any matrix such that $\mathbf{N}\mathbf{N}^T = \mathbf{M}$. When we mention a matrix function, we will refer to the standard matrix function defined by (1.22). A nice consequence of this definition is that eigenvectors are invariant under matrix functions for diagonalizable matrices.

To see that this definition makes sense, we consider two functions; the matrix power, $\mathbf{M}^n$ and the matrix exponential, $e^{\mathbf{M}}$, for the Jordan block

$$\mathbf{M} = \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix} \tag{1.23}$$

We can easily verify the power, by induction, we obtain

$$\mathbf{M}^n = \begin{pmatrix} a^{n-1} & (n-1)a^{n-2} \\ 0 & a^{n-1} \end{pmatrix} \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix} = \begin{pmatrix} a^n & na^{n-1} \\ 0 & a^n \end{pmatrix} \tag{1.24}$$

And since by verification

$$\mathbf{M}^2 = \begin{pmatrix} a^2 & 2a \\ 0 & a^2, \end{pmatrix} \tag{1.25}$$

the induction is obvious. Additionally,

$$\mathbf{M}^n = \begin{pmatrix} f(a) & f'(a) \\ 0 & f(a) \end{pmatrix} = \begin{pmatrix} a^n & na^{n-1} \\ 0 & a^n \end{pmatrix}, \tag{1.26}$$

and since both ways of computing $\mathbf{M}^n$ coincide, our definition makes sense.

Since the Taylor series for the exponential function converges everywhere, it should coincide with the definition above. Looking at the entries,

$$\begin{aligned} e^{\mathbf{M}} &= \mathbf{I} + \begin{pmatrix} a & 1 \\ 0 & a \end{pmatrix} + \frac{1}{2}\begin{pmatrix} a^2 & 2a \\ 0 & a^2 \end{pmatrix} + \cdots + \frac{1}{n!}\begin{pmatrix} a^n & na^{n-1} \\ 0 & a^n \end{pmatrix} + \cdots \\ &= \begin{pmatrix} \sum_{i=0}^{\infty} \frac{a^i}{i!} & \sum_{i=1}^{\infty} \frac{a^{i-1}}{(i-1)!} \\ 0 & \sum_{i=0}^{\infty} \frac{a^i}{i!} \end{pmatrix} = \begin{pmatrix} e^a & e^a \\ 0 & e^a \end{pmatrix}, \end{aligned} \tag{1.27}$$

and we see that the matrix function definition coincides with the Taylor series expansion.

An important property of matrix functions is that they commute under basis changes. For some of the applications we discuss in the suceeding chapters, this property is essential. Suppose that $\mathbf{M} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1} \in \mathbb{R}^{n\times n}$, and that $\mathbf{W} \in \mathbb{R}^{m\times n}$ is a (possibly redundant) basis for $\mathbb{R}^n$. Then,

$$f(\mathbf{W}\mathbf{M}\mathbf{W}^\dagger) = \mathbf{W} f(\mathbf{M}) \mathbf{W}^\dagger. \tag{1.28}$$

This follows simply from the definition, since if we expand $\mathbf{J}$ by a zero block, it still remains a Jordan block.

We conclude these generalities by outlining some applications of matrix functions. One of the most intuitive cases arises from generalising polynomial equations in one variable to where the variable is replaced with a matrix. Take, e.g. the quadratic matrix equation

$$\mathbf{A}\mathbf{X}^2 + \mathbf{B}\mathbf{X} + \mathbf{C} = \mathbf{O}, \tag{1.29}$$

where $\mathbf{O}$ is an all zero matrix. Then, in fact, the solution involves a matrix square root, but other than that resembles the usual formula

$$\mathbf{X} = -\frac{1}{2}\mathbf{B} + \frac{1}{2}\left(\mathbf{B}^2 - 4\mathbf{C}\right)^{1/2}. \tag{1.30}$$

A particular problem one encounters often in the literature when exploring high dimensional problems featuring Krylov methods and matrix functions is computations regarding Lattice quantum chromodynamics (QCD). We mention (van den Eshof et al., 2002; van den Eshof and Sleijpen, 2003), but these are only a few. The main problem here is to compute $\mathbf{y} = \text{sign}(\mathbf{M} - z\mathbf{I})\mathbf{x}$, for Hermitean $\mathbf{M}$, and $z \in \mathbb{C}$. Fortunately, this problem has close connections to principal component regression. We treat the principal component regression problem using the matrix sign function superficially in the next section.

We conclude the generalities by mentioning the topic of sensitivity analysis for matrix function. This is a big research area and is related to the Fréchet derivative of a matrix function. This has not been investigated in this thesis, but we note that all the chapters in Higham (2008) touch upon this subject for the respective matrix functions. Since we are dealing with positive definite matrices, we are in the best possible scenario for most relevant functions.

1.3.1 Matrix functions for precision matrices

If the matrix $\mathbf{Q}$ is positive semi-definite, the eigenvectors are orthogonal and the Jordan blocks have size 1, and hence,

$$f(\mathbf{Q}) = \mathbf{V}\begin{pmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{pmatrix}\mathbf{V}^T. \tag{1.31}$$

This is the situation that is relevant for Gaussian distributions, since all matrix function are to be computed on the precision matrix or a modification of it.

There are several matrix functions that are of particular interest for Gaussian computation. The most obvious one is the matrix inverse square root, since if $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{Q}^{-1/2}\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$, or alternatively, the matrix square root if one works in the covariance domain.

The second matrix function of clear interest is the matrix logarithm. This is the basis for computing the log-likelihood for GMRFs having high dimensional, sparse precision matrices. The reason why it is important is indirect, and relies on the identity

$$\log\det \mathbf{Q} = \mathrm{tr}\ \log \mathbf{Q}. \tag{1.32}$$

How to obtain efficient computational methods for the matrix inverse square root and the matrix logarithm is explored in Chapter 2 and 3. We also explore an alternative "log-precision" parametrisation, so that the matrix exponential becomes the matrix function of interest. This appears in Chapter 3. Note that it is possible to solve linear systems using the matrix exponential, as well (see e.g. Hasan et al. (2011)), and such methods can probably be facilitated for sampling as well.

These are, however, not the only matrix functions of interest. The matrix sign-function, for instance, may possibly play an important role in principal component regression (Faber and Kowalski, 1997), which is commonly used in chemical analyses. This is essentially a regularised regression estimate where instead of damping small spectral components, we disregard them completely. Take the ordinary regression model as in (1.7). Assuming no prior model for $\mathbf{x}$ yields the ordinary least squares regression,

$$\mathbb{E}(\mathbf{x}|\mathbf{y}) = (\mathbf{A}^T\mathbf{Q}_\epsilon\mathbf{A})^{-1}\mathbf{y}_{\mathrm{mod}}. \tag{1.33}$$

Principal component regression, however, uses the spectral decomposition of $\mathbf{A}^T\mathbf{Q}_\epsilon\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ and produces a regularised estimate by selecting some λs and their corresponding eigenvectors, so that

$$\mathbf{x}_{\mathrm{PCR}} = \sum_{i\in\mathcal{I}} \lambda_i^{-1}(\mathbf{v}_i\mathbf{v}_i^T)\mathbf{y}_{\mathrm{mod}}, \tag{1.34}$$

where typically eigenvectors below a threshold, $\{j|\alpha > \lambda_j\} \notin \mathcal{I}$. How this can be related to the matrix sign function can be seen as follows, if $f(x) = x^{-1}(\text{sign}(x-\alpha)+1)/2$, then $f(\mathbf{A}^T\mathbf{Q}_\epsilon^{-1}\mathbf{A})\mathbf{y}_{\text{mod}} = \mathbf{x}_{\text{PRC}}$. It is, of course, possible to generalise this, so that $f(x) = x^{-1}\left(\sum_j \text{sign}(x-\alpha_j)+1\right)/2$, which means that we only allow projections of eigenvectors with eigenvalues in certain intervals to appear in our solution.

One "drawback" is that we have no explicit control over the number of principal components to include, but it is still possible to easily estimate α or several α_js by a cross-validation or other predictive procedure. It may be possible to remedy this "drawback" by considering the following: If $f(\mathbf{Q}) = (\text{sign}(\mathbf{Q}-\alpha\mathbf{I})+\mathbf{I})/2$,

$$f(\mathbf{Q}) = \mathbf{V}\begin{pmatrix} \mathbf{I}_{\lambda_i>\alpha} & \\ & \mathbf{O} \end{pmatrix}\mathbf{V}^T, \tag{1.35}$$

and hence

$$\begin{aligned} \text{tr}\left(f(\mathbf{Q})\right) &= \text{tr}\left(\mathbf{V}f(\mathbf{\Lambda})\mathbf{V}^T\right) = \text{tr}\left(\mathbf{V}^T\mathbf{V}f(\mathbf{\Lambda})\right) \\ &= \text{card}(\{\lambda|\lambda_i \geq \alpha\}). \end{aligned} \tag{1.36}$$

So $f(\mathbf{Q})$ counts the number of eigenvalues larger than α. This quantity can be computed using probing vectors with techniques similar to those in Chapter 3.

The matrix sign function is essentially as easy to compute as the inverse matrix square root by rational approximations (Bloch et al., 2009), but a nested Krylov subspace approximation seems to be even better (Bloch and Heybrock, 2009) for computing $\text{sign}(\mathbf{Q})\mathbf{v}$ – that is, the sign function of a matrix times a vector. We do not explore this topic further, but mark it as an interesting venue for people who are interested in applying principal component regression in a high dimensional setting.

The reason that it is possible to treat the Gaussian distribution through matrix functions computationally, is that we only need the action $f(\mathbf{Q})\mathbf{v}$

for different **v**s. This leads to much less storage requirements than computing each entry of $f(\mathbf{Q})$. A particular technique we have used fruitfully throughout this thesis is rational approximations to a matrix function times a vector, i.e.

$$f(\mathbf{Q})\mathbf{v} \approx \sum_{j=0}^{N} \alpha_j (\mathbf{Q} - \sigma_j \mathbf{I})^{-1} \mathbf{v}, \tag{1.37}$$

where the approximation should hold throughout the spectrum of $\mathbf{Q}$, and, preferably, perturbations of $\mathbf{Q}$. By using rational approximations, we have reduced the problem of computing $f(\mathbf{Q})\mathbf{v}$ to solving a family of shifted linear systems.

Note that in the rational approximation for $f(x) = x^{-1}(\text{sign}(x - \alpha) + 1)/2$,

$$f(x) = \frac{1}{2x} + \sum_{i=1}^{N} \frac{\alpha_i}{x(x + \sigma_i)}, \tag{1.38}$$

we have the expansion

$$\frac{1}{x(x + \sigma_i)} = \frac{1}{\sigma_i x} - \frac{1}{\sigma_i (x + \sigma_i)}, \tag{1.39}$$

so the rational approximations are as easy to evaluate for $f(x) = x^{-1}(\text{sign}(x - \alpha) + 1)/2$ as for $\text{sign}(x - \alpha)$. Problems occur, however, if $\alpha = \lambda_j$, where λ_j is an eigenvalue of the matrix in question.

1.4 Krylov methods

There exists a myriad of Krylov methods in the literature, tailored to the most specific and most general problems imaginable, as long as they deal with matrices and you can form cheap matrix vector products with these matrices. A thorough introduction to Krylov methods for solving linear systems can be found in Saad (2003) or Golub and van Loan (1996), while

one concerning large eigenvalue problems is Saad (1992), but these are by no means exhaustive.

We are mainly concerned with Krylov methods for solving linear systems and computing matrix functions. Simpson (2008) explores the use of Krylov methods for computing matrix functions in some detail. In all the succeeding chapters, they are essential parts of the algorithms. Since we are dealing with precision matrices, we will only consider Krylov methods for positive (semi-)definite matrices. These methods are usually simpler than those for more general matrices and usually have better convergence properties. In this thesis, the focus is not theory of Krylov methods, but rather the application of a good Krylov method for the problem at hand. Performance is measured through computational examples, but it is still important to know the definition of Krylov subspaces and some properties that we use in order to obtain the so-called "good" Krylov methods for our problems.

We define the Krylov subspace of dimension m wrt. $\mathbf{Q}$ and a vector $\mathbf{r}$ as

$$\mathcal{K}_m = \mathcal{K}_m(\mathbf{Q}, \mathbf{r}) = \text{span}\{\mathbf{r}, \mathbf{Q}\mathbf{r}, \ldots, \mathbf{Q}^{m-1}\mathbf{r}\}. \tag{1.40}$$

The most readily available method for generating an orthonormal basis for $\mathcal{K}_m(\mathbf{Q}, \mathbf{r})$ in the positive definite case is the Lanczos algorithm, given in Algorithm 2. In this algorithm,

$$\mathbf{T} = \text{tridiag}(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & & & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}. \tag{1.41}$$

In order to solve linear systems, $\mathbf{Q}\mathbf{x} = \mathbf{b}$, using the Lanczos method, set $\mathbf{v}_1 = (\mathbf{b} - \mathbf{Q}\mathbf{x}_0)\frac{1}{\|\mathbf{b}-\mathbf{Q}\mathbf{x}_0\|_2}$, where $\mathbf{x}_0$ is an initial guess for the solution, and set the approximation to $\mathbf{x}_m = \mathbf{x}_0 + \frac{1}{\|\mathbf{b}-\mathbf{Q}\mathbf{x}_0\|_2}\|_2\mathbf{V}_m\mathbf{T}_m^{-1}\mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, \ldots, 0)^T$. For a matrix function f, we can, for initial guess $\mathbf{x}_0 = \mathbf{0}$

Algorithm 2 Lanczos algorithm

Input: $\mathbf{Q}$ and a vector $\mathbf{v}_1$.
Output: Ortonormal basis $\mathbf{V}_m$, tridiagonal matrix $\mathbf{T}_m = \text{tridiag}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\alpha})$ s.t. $\mathbf{T}_m = \mathbf{V}_m^T \mathbf{Q} \mathbf{V}_m$
Set: $\beta_1 = 0$, $\mathbf{v}_0 = \mathbf{0}$
for $j = 1$ to m **do**
  $\mathbf{w}_j = \mathbf{Q}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
  $\alpha_j = \langle \mathbf{w}_j, \mathbf{v}_j \rangle$
  $\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$
  $\beta_{j+1} = \|\mathbf{w}_j\|_2$, if $\|\mathbf{w}_j\|_2 = \mathbf{0}$, stop.
  $\mathbf{v}_{j+1} = \mathbf{w}_j / \beta_{j+1}$
end for

find the approximation $f(\mathbf{Q})\mathbf{b} = \mathbf{x}_0 + \frac{1}{\|\mathbf{b}-\mathbf{Q}\mathbf{x}_0\|_2} \mathbf{V}_m f(\mathbf{Q}) \mathbf{e}_1$. By construction, we build a polynomial approximation $p(\mathbf{Q})\mathbf{b} \approx f(\mathbf{Q})\mathbf{b}$ that has good properties, which is shown in Simpson (2008), where error bounds are given for the Lanczos- and full orthogonalisation methods (see Saad (2003), Chapter 6.4 for the definition of this method). Typically stopping criteria for the conjugate gradient (CG) method for solving linear systems can be used for matrix functions as well. The CG method is given in Algorithm 3, and one of the many relevant theorems from Simpson (2008) is

Theorem 2. (Simpson (2008), Theorem 3.3) *For any Stieltjes function f, the error in the Lanczos approximation to $f(\mathbf{Q})\mathbf{b}$ satisfies*

$$\|f(\mathbf{Q})\mathbf{b} - \mathbf{V}_m f(\mathbf{T}_m)\mathbf{V}_m^T \mathbf{b}\|_2 \leq (f(\lambda_{min}) - a)\|\mathbf{r}_m\|_2, \tag{1.42}$$

where λ_{min} is the smalles eigenvalue of $\mathbf{Q}$, and $\mathbf{r}_m$ is the residual after using m iterations of the CG algorithm so solve $\mathbf{Q}\mathbf{x} = \mathbf{b}$.

In Algorithm 3, explicit formulae relate the coefficients of $\mathbf{T}_m$ to the coefficients in the algorithm, so that we do not need to run both methods if we want to use aspects of either. With some abuse of notation, let

Algorithm 3 Conjugate gradient method for solving $\mathbf{Qx} = \mathbf{y}$

Input: $\mathbf{Q}$, an initial guess $\mathbf{x}_0$ and tolerance
Output: Approximate solution, $\mathbf{x}_m$
Set: $\mathbf{r}_0 = \mathbf{b} - \mathbf{Q}\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$
for $j = 0, 1, \ldots$ until convergence **do**
  $\mathbf{p}_\mathbf{Q} = \mathbf{Q}\mathbf{p}_j$
  $\alpha_j = \langle \mathbf{r}_j, \mathbf{r}_j / \langle \mathbf{p}_\mathbf{Q}, \mathbf{p}_j \rangle$
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{p}_\mathbf{Q}$
  $\beta_j = \langle \mathbf{r}_{j+1}, \mathbf{r}_{j+1} \rangle / \langle \mathbf{r}_j, \mathbf{r}_j \rangle$
  $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
end for

$\mathbf{T}_m = \text{tridiag}(\boldsymbol{\eta}, \boldsymbol{\delta}, \boldsymbol{\eta})$, and let $\boldsymbol{\eta}, \boldsymbol{\delta}$ be given by

$$\delta_{j+1} = \begin{cases} \frac{1}{\alpha_j} & \text{if } j = 0 \\ \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}} & \text{otherwise} \end{cases} \tag{1.43}$$

$$\eta_{j+1} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}} \tag{1.44}$$

where α_j, β_j are computed by Algorithm 3.

One of the main points is that for computing $f(\mathbf{Q})\mathbf{b}$ using the Lanczos approximation, the matrix function is only evaluated on the much smaller matrix $\mathbf{T}_m$, in addition to requiring relatively few iterations to converge. In Chapter 2 and 3, rational approximation are used. Rational approximations has the advantage that only systems of the form $\alpha_j(\mathbf{T}_m - \sigma_j \mathbf{I})^{-1}\mathbf{b}$ needs to be solved. Another benefit of using rational approximations is that $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}) = \mathcal{K}_m(\mathbf{Q} - \sigma\mathbf{I}, \mathbf{r})$ for any $\sigma \in \mathbb{C}$, and variant Krylov methods using this fact can be used. In Chapter 2 and 3, this is exploited with great success.

If m is very small, say, below 200, which may happen when we are dealing with preconditioned iterations, eigen decompositions are also possible, as

the $\mathcal{O}(m^3)$ cost is dwarfed by the matrix vector product calculations. In this situation, it is probably better to use particular methods for the matrix function in question, discussed in detail in Higham (2008). We give one such approach in Chapter 2 based on the polar decomposition.

1.5 Overview of contributions

This thesis is primarily based on the following four articles that have been submitted to scientific journals (Aune et al., 2012a,c; Aune and Simpson, 2012; Aune et al., 2012b)

- Aune et. al., *Iterative Numerical Methods for Sampling from High Dimensional Gaussian Distributions*, Statistics and Computing (2012). Accepted.
- Aune et. al., *Parameter estimation for high dimensional Gaussian distributions*. Statistics and Computing (2012). In revision.
- Aune et. al., *Three-dimensional non-stationary and non-linear AVA inversion*. Geophysical Journal International (2012). Submitted.
- Aune and Simpson, *The use of systems of stochastic PDEs as priors in seismic AVA inversion*. Scandinavian Journal of Statistics (2012). Submitted.

The content is not, however, limited to that found in these articles. Here, we give a brief account of the different chapters' contents.

In Chapter 2, the use of Krylov method for sampling from high dimensional Gaussian distribution is treated in detail. Specifically, the practical implementation and relative performance of some specific methods is discussed, and potential of using graphical processing units for these methods. In addition to the content in Aune et al. (2012a), another method for preconditioned sampling is proposed, and we discuss sampling from intrinsic Gaussian variables and conditioned sampling. We also briefly discuss alternative iterations for the matrix inverse square root, that has not been

considered in the literature in the context of sampling from Gaussian distributions. The methodology developed in this chapter is for Gaussian sampling only, but it is easily adapted to sampling from generalised Laplace fields (Bolin, 2012), and can be used for proposals for even more general distributions. The moral is that for high-dimensional Gaussian variables, and even for variables in moderate dimensions, the methods can be faster than traditional approaches, which is a boon in any computational setting.

Chapter 3 is devoted to the computation of the Gaussian likelihood when the precision matrices and/or observation process depends on hyperparameters. The approach is based on a determinantal identity, rational approximation to the matrix logarithm and cleverly chosen probing vectors, with appropriate variations for specific problems. This content is predominantly found in Aune et al. (2012c). In addition to this, two additional alternative modeling approaches for the prior distribution are presented. One is based on a sparse log-precision matrix parametrisation and the other is based on approximate algorithmic models using preconditioning techniques. The alternative methodologies come with computational benefits that can be utilised in a high-dimensional setting. The main benefit of the log-determinant approximation developed in the chapter is that it is possible to perform maximum likelihood estimation in settings in which the dimensionality of the variables proves prohibitive for traditional approaches.

Chapter 4 deals with the seismic AVA inversion problem, where we from seismic observations want to infer relative differences in elastic parameters. In particular, it deals with a non-linear approximate forward model that may improve inversion accuracy in some specific situations, and the complications this non-linear forward model introduces when doing inference. The results are here summarised by parameter sweeps over different noise levels for various inversion scenarios.

In Chapter 5, we present a way of constructing flexible Gaussian priors for multivariate inverse problems where there are discrete geometric structures that can be identified before inversion. We use systems of linear SPDEs for doing this. We also present a new way of specifying fuzzy interfaces by

using geodesic blending of local correlation matrices between the interfaces of the multivariate discrete structures. This is motivated by different layers in the subsurface in seismic AVA inversion.

Much of the work on this thesis has been of implementation type: To make sure that the models and algorithms work in practice and have potential in relevant industries, much testing and tweaking of many different implementations have been done. The amount of code needed to produce the desired results is perhaps not transparent in the exposition in the subsequent chapters, but it is, in fact, substantial. A fraction of the methodology in this thesis is implemented in the library KRYLSTAT, available at `http://www.math.ntnu.no/~erlenda/KRYLSTAT/`. Some care is needed to make the routines work with regards to compilation.

Chapter 2

Iterative Numerical Methods for Sampling from High Dimensional Gaussian Distributions

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 tools, such as satellite or seismic data, are influencing the focus of modeling and methods, see e.g. Buland et al. (2003), Banerjee et al. (2008) and Cressie and Johannesson (2008). The most common distribution for such high dimensional problems is the Gaussian distribution. The computational requirements for inference are then evaluation of a quadratic form and a determinant. For sampling based approaches we must be able to sample a variable with the right mean and covariance structure. Samples are useful for understanding high-level interactions. For instance, climate models, hydrological models, weather forecasting and petroleum reservoir prediction all rely on the propagation of samples (ensembles) over time. Moreover, a posterior model is often represented in canonical form, where the precision matrix $\mathbf{Q}$ enters in the quadratic part. The entries of $\mathbf{Q}$ are difficult to interpret, and sampling is a

natural way to assess the uncertainty and correlations.

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 a 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 chapter we explore iterative methods for sampling high dimensional Gaussian processes. The purpose of this is to show that there exist iterative methods that have comparable performance to Cholesky sampling in lower dimensions, and remain useful in the domain where Cholesky sampling gets impossible due to excessive memory requirements. The natural assumption in this setting is that the precision matrix $\mathbf{Q}$ has a sparse Markov structure, or that the matrix vector product $\mathbf{Qx}$ is available as a fast black-box procedure for any input $\mathbf{x}$. Here, 'iterative methods' mean iterative numerical linear algebra methods, see Golub and van Loan (1996), Trefethen and Bau (1997) and Saad (2003). We do not refer to random iterative sampling methods such as Markov chain Monte Carlo (MCMC). Our proposed sampling methods use numerical methods to compute $\mathbf{x} = \mathbf{Q}^{-1/2}\mathbf{z}$, where $\mathbf{z}$ is a vector of i.i.d. normal variables and $\mathbf{Q}^{-1/2}$ is the principal matrix inverse square root. The error of the sample, that is, how much it deviates from a sample from the true underlying distributions, 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 on that (Saad (2003)). The second is based on a quadrature representation formula arising from Cauchy's integral formula and conformal mappings of regions $U \in \mathbb{C}$, see Hale et al. (2008). For an accessible introduction to complex analysis, see Stein and Shakarchi (2003). The third procedure is a continuous deformation method based on a system of ODEs found in Allen et al. (2000).

Iterative methods are used for solving $\mathbf{Qx} = \mathbf{b}$ several times sequentially or concurrently in order to build the sample. 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 GPU, allowing fast parallel computing. We explore how using sparse matrix vector product on the GPU affect the performance of the iterative methods in this setting. A tutorial on using CUDA, a C++/C interface to the GPU, can be found on nVidias website (`http://www.nvidia.com/cuda/`). Statisticians are likely to 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 Parker and Fox (2011) study Krylov subspace approximations, where samples are in the directions obtained by the CG method. Their realizations are fast to generate, but they oversmooth the process. Moreover, they are impractical for large problems due to inherent instability in the

presence in round-off error in the orthogonal vectors generated by the Krylov method. Simpson et al. (2008) 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 chapter is organised as follows: Section 2.1 gives some background model assumptions and a review of direct sampling methods. In Section 2.2 we present the iterative sampling procedures, which are applied to examples in Section 2.3. In this example section, we compare timings for two different sparse models and sample the posterior for elastic earth properties given seismic 3D data from a North Sea reservoir.

2.1 Modelling assumptions and direct sampling methods

The distribution of a n-dimensional Gaussian random variable $\mathbf{x} = (x_1, \ldots, x_n)^T$, denoted $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$ for a symmetric positive definite $\mathbf{Q}$, is given by

$$\begin{aligned} p(\mathbf{x}) &= \frac{|\mathbf{Q}|^{1/2}}{(2\pi)^{n/2}} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{Q}(\mathbf{x}-\boldsymbol{\mu})] \\ &\propto \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{b}), \end{aligned} \tag{2.1}$$

where the covariance matrix is $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$, and $\mathbf{Q}$ is the precision matrix. The linear canonical parameter $\mathbf{b} = \mathbf{Q}\boldsymbol{\mu}$ links the mean $\boldsymbol{\mu}$ and the precision matrix. For GMRFs the precision matrix $\mathbf{Q}$ is sparse, with $Q_{i,j} = 0$ unless i, j are neighbours on a graph. On a 2D-grid, the first order neighbourhood is defined by the cells north, east, south and west. For a map of regions, the neighbours have a common border. The extension to second and higher

order neighbourhoods 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.

2.1.1 Gaussian linear model

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

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &\propto \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{Q}(\mathbf{x}-\boldsymbol{\mu}) - \\ &\quad \frac{1}{2}(\mathbf{y}-\mathbf{G}\mathbf{x})^T\mathbf{Q}_{lik}(\mathbf{y}-\mathbf{G}\mathbf{x})] \\ &\propto \exp[-\frac{1}{2}\mathbf{x}^T(\mathbf{Q}+\mathbf{G}^T\mathbf{Q}_{lik}\mathbf{G})\mathbf{x} + \\ &\quad \mathbf{x}^T(\mathbf{Q}\boldsymbol{\mu}+\mathbf{G}^T\mathbf{Q}_{lik}\mathbf{y})]. \end{aligned} \tag{2.2}$$

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

We consider sampling methods for both the prior process and the posterior process. Posterior sampling can be done either by sampling from the Gaussian process defined via the conditional $p(\mathbf{x}|\mathbf{y})$, 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 simplest of these strategies will depend on the situation.

2.1.2 Direct sampling of Gaussian processes

A key observation when sampling from a Gaussian is the following: If the precision matrix $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$, then the covariance is $\mathbf{\Sigma} = (\mathbf{L}\mathbf{L}^T)^{-1} = \mathbf{L}^{-T}\mathbf{L}^{-1}$, where $\mathbf{L}^{-T} = (\mathbf{L}^{-1})^T$. If we want a sample $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$, it is enough to compute $\mathbf{x} = \mathbf{L}^{-T}\mathbf{z}$ for $\mathbf{z} \sim \mathcal{N}(0, I)$, since $\mathrm{Cov}(\mathbf{L}^{-T}\mathbf{z}) = \mathbf{L}^{-T}\,\mathrm{Cov}(\mathbf{z})\mathbf{L}^{-1} = (\mathbf{L}\mathbf{L}^T)^{-1}$. This approach is called the Cholesky sampling from a GMRF (Rue (2001) and Rue and Held (2005)). The matrix $\mathbf{L}$ is the lower-triangular Cholesky factor. In an autoregressive graph with first order neighbourhood, $\mathbf{L}$ has non-zero entries only along the diagonal and the first sub-diagonal. The Cholesky factor may be fast to compute from $\mathbf{Q}$ depending on its non-zero structure; 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 $O(n^2)$. Here n denotes the number of discretization points of the underlying space. Moreover, the storage requirements for computing $\mathbf{L}$ become enormous in high dimensions because of the large fill-in between the non-zero structure of $\mathbf{Q}$ and the larger non-zero structure of $\mathbf{L}$. One can reduce the storage requirements by intelligent reordering of the n indices in the graph, but for a three dimensional grid, the reordering we tried did not prove particularly helpful in our study. A remedy is to apply Cholesky factorization 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 $\mathbf{Q}^{-1/2}$ as the principal

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

$$\begin{aligned} \mathrm{Cov}(\mathbf{V}\mathbf{D}^{-1/2}\mathbf{V}^T\mathbf{z}) &= \mathbf{V}\mathbf{D}^{-1/2}\mathbf{V}^T\mathbf{I}(\mathbf{V}\mathbf{D}^{-1/2}\mathbf{V}^T)^T \\ &= \mathbf{V}\mathbf{D}^{-1}\mathbf{V}^T = \mathbf{Q}^{-1}, \end{aligned} \tag{2.3}$$

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 $\mathbf{Q}$ 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 $\mathbf{Q}$ through clever discretization of the spatial domain, provided that the spatial domain is regular enough (it should at least be a convex subset of $\mathbb{R}^n$). For more complex domains, alternative approaches may be preferable - see e.g. Lindgren et al. (2011) for details on one such method.

We note that Cholesky and matrix function sampling give different samples $\mathbf{x}$, even though they use the same input i.i.d. variable $\mathbf{z}$. Still, both realizations are from the correct distribution, and correspond to one another through a unitary matrix given by the polar decomposition of $\mathbf{L}$ (Higham, 2008). Even though the Cholesky method and the eigen representation 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.

2.2 Iterative numerical methods for sampling

There are two major ingredients in our iterative methods for sampling from Gaussian distributions. The first one is the rational approximation

$f_N(\cdot)$ of the matrix inverse square root $\mathbf{Q}^{-1/2}$, i.e. we design a rational function $f_N(\cdot)$ such that $f_N(\mathbf{Q})\mathbf{z} \approx \mathbf{Q}^{-1/2}\mathbf{z}$ on the spectrum of $\mathbf{Q}$. To sample $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$ we set

$$\mathbf{x} = \mathbf{Q}^{-1/2}\mathbf{z} \approx f_N(\mathbf{Q})\mathbf{z} = \sum_{j=1}^{N} \alpha_j (\mathbf{Q} - \sigma_j \mathbf{I})^{-1}\mathbf{z}, \tag{2.4}$$

for $\sigma_j \in \mathbb{R}_-, \alpha_j \in \mathbb{R}$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The expected value in the Gaussian is added afterwards. The specification of α_j, σ_j is described in Section 2.2.3, except for the method described in Section 2.2.6, which is a bit different. The quantity N is typically less than 10.

The second ingredient is solving

$$(\mathbf{Q} - \sigma_j \mathbf{I})\mathbf{x} = \mathbf{z}, \tag{2.5}$$

for each shift σ_j in the rational approximation (2.4). We use iterative numerical linear algebra techniques (Krylov methods) to solve this equation. They are used throughout Section 2.2, but the basic idea is given in Section 2.2.1. Direct methods for solving (2.5) are based on Cholesky factorization of $(\mathbf{Q} - \sigma_j \mathbf{I})$ or a domain decomposition method (see e.g. Saad (2003)) followed by Cholesky factorizations. Obviously, the first of these is a not a good choice, as a sample can be obtained directly if we have the Cholesky factor of $\mathbf{Q}$, and in the situations we consider here, this is not possible. The second strategy without the shift may possibly provide an alternative to our methods, but this approach is not explored here.

The resulting sample $\mathbf{x}$ from equation (2.4) is in the span of the combined Krylov space constructed for different components $j = 1, \ldots, N$ of the rational approximation. The iterative Krylov methods are truncated using a numerical tolerance on the residual norm. Independent samples can be generated by using independent starting variable $\mathbf{z}$ in the construction.

2.2.1 Krylov methods

The most famous and readily available Krylov method for solving (2.5) is the CG method, first published by Hestenes and Stiefel (1952). For iteration number m it minimises the functional

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{x}^T\mathbf{z}. \tag{2.6}$$

on the subspace

$$\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{Q}\mathbf{r}_0, \mathbf{Q}^2\mathbf{r}_0, \ldots, \mathbf{Q}^m\mathbf{r}_0\},$$

which is referred to as the Krylov space. Here, $\mathbf{x}_0$ is the initial value, usually set to $\mathbf{0}$, and the associated residual is $\mathbf{r}_0 = \mathbf{x} - \mathbf{Q}\mathbf{x}_0$. Naturally, for a unique minimum to exist, and thus the CG method to be operational, $\mathbf{Q}$ must be symmetric positive definite. This is the case for all precision matrices which define Gaussian fields that are not intrinsic. A nice derivation of the CG method from this point of view can be found on Wikipedia (Conjugate gradient method) or in the electronic publication (Shewchuk (1994)). To solve (2.5), we use the CG method with shifts; i.e. we solve for $\mathbf{Q} \to \mathbf{Q} - \sigma_j\mathbf{I}$, $j = 1, \ldots, N$, in the Krylov space using Algorithm 4 for several σ_js.

An equivalent point of view is that the conjugate gradient method finds the best possible solution to $\mathbf{Q}\mathbf{x} = \mathbf{z}$ in the subspace $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0)$. This point of view is explored in detail in Saad (2003) together with a myriad of different Krylov methods which may be obtained by using different subspaces. It is possible to replace all the Krylov methods in what follows by another suited Krylov method and obtain similar methods that may be better suited for special problems.

There are many ways to control convergence, i.e. the number of iterations m required. Our criterion is based on a relative tolerance criterion (see Saad (2003)) on the 2-norm of the residual vector. The CG method is given in Algorithm 4. In addition to the matrix $\mathbf{Q}$, it requires as inputs the desired accuracy (relative 2-norm of residual error).

Algorithm 4 Conjugate gradient algorithm for computing $\mathbf{x} = \mathbf{Q}^{-1}\mathbf{z}$

Input: $\mathbf{Q}$, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, convergence tolerance.
Output: $\mathbf{x} \approx \mathbf{Q}^{-1}\mathbf{z}$
Set: $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_{cur} = \mathbf{z} - \mathbf{Q}\mathbf{x}_0$, $\mathbf{p} = \mathbf{r}_{cur}$
for $j = 1...$ to converged **do**
  $\mathbf{q}_p = \mathbf{Q}\mathbf{p}$
  $\alpha = \frac{\langle \mathbf{r}_{cur}, \mathbf{r}_{cur} \rangle}{\langle \mathbf{q}_p, \mathbf{p} \rangle}$
  $\mathbf{x} \to \mathbf{x} + \alpha\mathbf{p}$
  $\mathbf{r}_{new} = \mathbf{r}_{cur} - \alpha\, \mathbf{q}_p$
  $\beta = \frac{\langle \mathbf{r}_{new}, \mathbf{r}_{new} \rangle}{\langle \mathbf{r}_{cur}, \mathbf{r}_{cur} \rangle}$
  $\mathbf{p} = \mathbf{r}_{new} + \beta\mathbf{p}$
  $\mathbf{r}_{cur} = \mathbf{r}_{new}$
end for

The performance of Algorithm 4 essentially depends on two things: 1) The performance of the matrix vector product $\mathbf{Qp}$ and 2) the condition number of the matrix $\mathbf{Q}$, i.e. the ratio of its extremal eigenvalues $\kappa = \lambda_{max}/\lambda_{min}$. If the matrix vector product is fast, we are assured that each iteration in Algorithm 4 is fast. If κ is small, we are assured that the algorithm will require only a few iterations to converge (see Saad (2003) and Golub and van Loan (1996)). These are the essential aspects that immediately apply to the Krylov methods in the following sections as well. If $\mathbf{Q}$ is defined by a Markov random field, it is typically very sparse, and hence the matrix vector product $\mathbf{Qp}$ is fast. If $\mathbf{Qp}$ is fast by some other virtue, sampling using Krylov methods should be fast, provided the conditioning is not too bad. The routine for computing a sample using CG and (2.4) is given in Algorithm 5.

If a matrix has a particularly large (bad) condition number, κ, a possible remedy is using a preconditioner, $\mathbf{M}$. Now, we solve, for instance, the system $\mathbf{MQM}^T\mathbf{y} = \mathbf{Mb}$ followed by $\mathbf{M}^T\mathbf{y} = \mathbf{x}$ 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 $\mathbf{Mr}$ must be fast to

Algorithm 5 Sequential CG for computing $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$

Input: $\mathbf{Q}$, σ_j, α_j according to (2.4) for $j \in 1, \ldots, N$
Output: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$
Set: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x} = \mathbf{0}$
for $j = 1, \ldots, N$ **do**
  Compute $\mathbf{y}_j = (\mathbf{Q} - \sigma_j \mathbf{I})^{-1}\mathbf{z}$ using Algorithm 4
  $\mathbf{x} = \mathbf{x} + \alpha_j \mathbf{y}_j$
end for

compute, and also that the preconditioner may change for each shift; a way to compute an approximate inverse for each shift is described in Benzi and Bertaccini (2003). Typically, $\mathbf{M} \approx \mathbf{Q}^{-1}$, but is much faster to compute than $\mathbf{Q}^{-1}$. We mention also that apart from in the usual CG method, preconditioning can be difficult. It is for instance difficult to precondition the method CG-M described below. Moreover, preconditioners can be hard to parallelize which may be an important practical consideration.

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

In the Bayesian setting, the precision matrix $\mathbf{Q}$ often depends on hyper-parameters, $\boldsymbol{\eta}$, that require estimation. For optimization, this requires evaluation of the log-likelihood; that is the logarithm of an expression similar to (2.1). This again requires evaluation of a log-determinant. By using the identity $\log\det\mathbf{Q} = \operatorname{tr}\log\mathbf{Q}$, along with probing vectors and the matrix logarithm, Aune et al. (2012c) show that similar algorithms allow approximate evaluation of log-determinants using Krylov methods.

2.2.2 Lanczos methods for the inverse square root

The Lanczos method for sampling $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$ provides us with an alternative method to that of Algorithm 5. It is the building block for

self-adjoint Krylov methods and is perhaps the easiest ways of forming an orthonormal basis for $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0)$. In principle the method is simple; it builds an orthonormal basis for the Krylov subspace $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0)$, namely $\mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_m)$ as well as the tridiagonal Hessenberg matrix $\mathbf{T}_m = \mathbf{V}^T\mathbf{Q}\mathbf{V}$. Lastly, we project out the closest solution to $\mathbf{Q}^{-1/2}\mathbf{z}$ in $\mathcal{K}(\mathbf{Q}, \mathbf{r}_0)$, by $\mathbf{x} = \mathbf{x}_0 + \beta_0\mathbf{V}\mathbf{T}^{-1/2}\mathbf{e}_1$, as in Algorithm 6 given below.

Algorithm 6 Lanczos algorithm obtaining a sample $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$

Input: $\mathbf{Q}$, dimension of Krylov subspace, m
Output: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$
Set: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{z} - \mathbf{Q}\mathbf{x}_0$, $\beta_0 = \|\mathbf{r}_0\|$, $\mathbf{v}_1 = \mathbf{r}_0/\beta_0$, $\mathbf{v}_0 = \mathbf{0}$, $\mathbf{e}_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^m$
for $j = 1$ to m **do**
  $\mathbf{w}_j = \mathbf{Q}\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}$
  $\alpha_j = \langle \mathbf{w}_j, \mathbf{v}_j \rangle$
  $\mathbf{w}_j = \mathbf{w}_j - \alpha_j\mathbf{v}_j$
  $\beta_{j+1} = \|\mathbf{w}_j\|$
  $\mathbf{v}_{j+1} = \mathbf{w}_j/\beta_{j+1}$
end for
Set: $\mathbf{T}_m = \text{tridiag}(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$
Compute: $\mathbf{x} = \mathbf{x}_0 + \beta_0\mathbf{V}\mathbf{T}^{-1/2}\mathbf{e}_1$.

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 $\mathbf{T}$ and the resulting sample $\mathbf{x}$ in a reasonable amount of time. For this we have to compute $\mathbf{T}^{-1/2}\mathbf{e}_1 = \mathbf{V}_T\mathbf{\Lambda}_T^{-1/2}\mathbf{V}_T\mathbf{e}_1$, where $\mathbf{V}_T, \mathbf{\Lambda}_T, \mathbf{V}_T$ is the eigen decomposition of $\mathbf{T}$. An alternative approach comes from considering rational approximations, $\mathbf{T}^{-1/2}\mathbf{e}_1 \approx f_N(\mathbf{T})\mathbf{e}_1$, as in (2.4). This essentially requires a fast tridiagonal solver, and such solvers have computational complexity of $\mathcal{O}(n)$. The samples obtained using $f_N(\mathbf{T})\mathbf{e}_1$ are equivalent to those obtained by Algorithm 5 if the subspace dimensions are identical.

Simpson et al. (2008) present a theorem for the error of the Lanczos approx-

imation:

Theorem 3. *Let* $\mathbf{Q}$ *be symmetric positive definite with smallest eigenvalue* λ_{min}*. Then*

$$\|\mathbf{Q}^{-1/2}\mathbf{z} - \|\mathbf{z}\|_2\mathbf{V}\mathbf{T}^{-1/2}\mathbf{e}_1\|_2 \leq \lambda_{min}^{-1/2}\|\mathbf{r}\|, \tag{2.7}$$

where $\mathbf{r}$ *is the residual after* m *iterations of conjugate gradients to solve* $\mathbf{Q}\mathbf{x} = \mathbf{z}$*.*

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 accommodate this. Since we want to compute several samples, it is practically more efficient to pre-compute the number of Krylov dimensions required using the CG algorithm. We do this pre-computation on a number of samples $\mathcal{N}(\mathbf{0}, \mathbf{I}_n)$, and use the dimension of the largest Krylov subspace needed in the Lanczos approximation of the inverse square root times a vector. This also allows us to control the amount of memory used by the algorithm.

We note that it is difficult to interpret Theorem 3 in terms of the difference between a Lanczos approximation and an exact sample. However, the Lanczos process quickly identifies the needed components for the small scale variability defined by $\mathbf{Q}$. We tested this by visualizing various samples for different ms in Algorithm 6. The smallest m terms appear to give an almost independent process, but capture the small scale variability of the process. With increasing ms, the Lanczos process captures the smoothness of the sample. It is possible to obtain such a visualisation at `http://www.math.ntnu.no/~erlenda/misc`.

In general, loss of orthogonality of the basis $\mathbf{v}_1, \dots \mathbf{v}_m$ for $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0)$ may be a challenge. For our purposes this does not seem to be an issue. See, however Simon (1984) for an analysis of the Lanczos method with and without reorthogonalization of the basis vectors. For us, the more important aspect is storage of the basis; it is evident that we cannot store 1000 $\mathbf{v}_i \in \mathbb{R}^{10^7}$.

There are two ways to overcome this - the first is to use a restart strategy, essentially restarting when loss of orthogonality occurs or when the memory limit is reached. This approach is investigated in Ilic et al. (2010). Another approach is to use a so-called 2-pass strategy. This 2-pass strategy is mentioned in Frommer and Simoncini (2008) and we implement a version of it here.

In the 2-pass Lanczos algorithm we only need $\mathbf{v}_j, \mathbf{v}_{j-1}$ in each iteration to compute α_j, β_j. We exploit this, and compute $\mathbf{T}$ first, discarding the older $\mathbf{v}_i$s, so that we do not have to store them. In the next pass, we compute $\mathbf{x}_j = (\mathbf{T}^{-1/2}\mathbf{e}_1)_j \mathbf{v}_j$ and sum the $\mathbf{x}_j$s as we pass through the Lanczos iterations once more.

Another Lanczos-type algorithm we have implemented involves deflating some orthonormal vectors into the Lanczos procedure. That is, given some orthonormal vectors $\{\mathbf{w}_i\}_{i=1}^{s}$ $(s < n, m)$ in the eigenspace of $\mathbf{Q}$, we construct a Krylov space $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0) \perp \mathbf{w}_i \ \forall i \in 1, \ldots, s$. This has the effect of improving the conditioning of the system as per a preconditioner (Saad et al. (1999)). A similar procedure is explored in detail in Ilic et al. (2008) and it can be adapted to this setting. Specifically, let $\mathbf{W} = (\mathbf{w}_1 \mathbf{w}_2 \cdots \mathbf{w}_s)$ be a matrix with the given orthonormal eigenvectors, and $\lambda_i, i = 1, \ldots, s$ are the corresponding eigenvalues. In order to construct a Krylov basis which is orthogonal to $\mathbf{W}$, let $\mathbf{x}_0 = \mathbf{x}_{-1} + \mathbf{W}(\mathbf{W}^T\mathbf{Q}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{r}_{-1}$, with $\mathbf{x}_{-1}$ arbitrary (e.g. $\mathbf{x}_{-1} = \mathbf{0}$) and $\mathbf{r}_{-1} = \mathbf{z} - \mathbf{Q}\mathbf{x}_{-1}$. This is a projection of the solution of $\mathbf{Q}\mathbf{x} = \mathbf{z}$ onto the space spanned by the $\mathbf{w}_i$s. This initial value, $\mathbf{x}_0$, ensures that the Krylov vectors $\mathbf{v}_i$ are orthogonal to the $\mathbf{w}_i$s. Now, compute the Lanczos decomposition using a 2-pass version, compute the approximation $\mathbf{x}_{Krylov} = \mathbf{V}\mathbf{T}^{-1/2}\mathbf{e}_1$, and set $\mathbf{x}_{Proj} = \mathbf{W}\Lambda^{-1/2}\mathbf{W}^T\mathbf{z}$. Finally, the approximate solution is $\mathbf{x} = \mathbf{x}_{Krylov} + \mathbf{x}_{Proj}$. Note that if $\mathbf{W}$ has eigenvector columns we get $\mathbf{W}^T\mathbf{Q}\mathbf{W} = \mathrm{diag}(\lambda_1, \ldots, \lambda_s)$. The procedure is summarised in Algorithm 7 in which $\oslash$ is element-wise division and $\odot$ is element-wise multiplication.

An obvious drawback of algorithm 7 is the additional storage requirements of the approximated eigenvectors of $\mathbf{Q}$. Also, there is some overhead in

Algorithm 7 2-pass deflated eigenvector Lanczos algorithm for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$

Input: $\mathbf{W}$, $\mathbf{Q}$, $\mathbf{W}_Q = \mathbf{W}^T\mathbf{Q}$, $\mathbf{Q}_W = \mathbf{Q}\mathbf{W}$, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_s)^T$
Output: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$
Set: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ $\mathbf{r}_{-1} = \mathbf{z}$, $\mathbf{x}_0 = \mathbf{W}[(\mathbf{W}^T\mathbf{r}_{-1}) \oslash \boldsymbol{\lambda}]$, $\mathbf{r}_0 = \mathbf{z} - \mathbf{Q}\mathbf{x}_0$, $\beta_1 = \|\mathbf{r}_0\|$ $\mathbf{v}_{cur} = \mathbf{r}_0/\beta_1, \mathbf{v}_{old} = \mathbf{0}$
for $j = 1$ to m or converged (1st pass) **do**
  if $j = 1$ **then**
    $\mathbf{w} = \mathbf{Q}\mathbf{v}_{cur} - \mathbf{Q}_W(\mathbf{W}_Q\mathbf{v}_{cur} \oslash \boldsymbol{\lambda})$
  else
    $\mathbf{w} = \mathbf{Q}\mathbf{v}_{cur} - \mathbf{Q}_W(\mathbf{W}_Q\mathbf{v}_{cur} \oslash \boldsymbol{\lambda}) - \beta_j\mathbf{v}_{old}$
  end if
  $\alpha_j = \langle \mathbf{w}, \mathbf{v}_{cur} \rangle$
  $\mathbf{w} = \mathbf{w} - \alpha_j\mathbf{v}_{cur}$
  $\beta_{j+1} = \|\mathbf{w}\|$
  $\mathbf{v}_{old} = \mathbf{v}_{cur}$
  $\mathbf{v}_{cur} = \mathbf{w}/\beta_{j+1}$
end for
Set: $\mathbf{q} \approx \beta_1 \operatorname{trid}(\beta_2^m, \alpha_1^m, \alpha_2^m)^{-1/2}\mathbf{e}_1$ using (2.4) (here $\mathbf{e}_1 = (1, 0, \ldots, 0) \in \mathbb{R}^m$)
$\mathbf{v}_{cur} = \mathbf{r}_0/\beta_1$, $\mathbf{x}_{Krylov} = \mathbf{0}$
for $j = 1$ to m (2nd pass) **do**
  $\mathbf{x}_{Krylov} = \mathbf{x}_{Krylov} + q_j\mathbf{v}_{cur}$
  if $j = 1$ **then**
    $\mathbf{w} = \mathbf{Q}\mathbf{v}_{cur} - \mathbf{Q}_W(\mathbf{W}_Q v_{cur} \oslash \boldsymbol{\lambda})$
  else
    $\mathbf{w} = \mathbf{Q}\mathbf{v}_{cur} - \mathbf{Q}_W(\mathbf{W}_Q\mathbf{v}_{cur} \oslash \boldsymbol{\lambda}) - \beta_j\mathbf{v}_{old}$
  end if
  $\mathbf{w} = \mathbf{w} - \alpha_j\mathbf{v}_{cur}$
  $\mathbf{v}_{old} = \mathbf{v}_{cur}$
  $\mathbf{v}_{cur} = \mathbf{w}/\beta_{j+1}$
end for
$\mathbf{x} = \mathbf{W}(\boldsymbol{\lambda}^{-1/2} \odot (\mathbf{W}^T\mathbf{z})) + \mathbf{x}_{Krylov}$

the matrix vector, matrix-matrix computations in the algorithm, but most of this can be overcome by pre-computing $\mathbf{QW}, \mathbf{W}^T\mathbf{Q}$ and $(\mathbf{WQW}^T)^{-1}$. This comes at the cost of approximately tripling the initial vector storage requirements. However, this pre-computation 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.

The theory of Krylov methods is usually presented under the assumption of infinite precision arithmetic. On the computer, however, we need to deal with the fact that we only have access to finite precision arithmetic and representations of the quantities in question, usually 64 bits per real number. There are, however, good expositions on what happens to Krylov methods using finite precision arithmetic, and one such source is the long exposition of Meurant and Strakos (2006). For our methods, however, Theorem 3.4 in Simpson (2008) gives a rather definite answer. This also shows that we can use Theorem 3 to define a stopping criterion without further worries. The Cholesky method does have some small issues in finite precision arithmetic for very ill-conditioned matrices, but these are less severe than those for Krylov methods if they occur. An approach to address this for linear systems can be found in Riley (1955) and it generalises trivially to our setting.

2.2.3 Optimal rational approximations with linear solves

The rational approximation in (2.4) require careful choices of the α_js and σ_js, which in turn can give a small number N. One can achieve attractive solutions through numerical quadrature of a contour integral. More generally, for functions that are analytic in some domain containing the spectrum of

$\mathbf{Q}$, it is possible to compute $f(\mathbf{Q})\mathbf{z}$ by Cauchy's integral formula

$$f(\mathbf{Q})\mathbf{z} = \frac{1}{2\pi i}\oint_{\Gamma} f(\zeta)(\zeta\mathbf{I} - \mathbf{Q})^{-1}\mathbf{z}d\zeta, \tag{2.8}$$

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

$$\mathbf{Q}^{-1/2}\mathbf{z} = \frac{1}{2\pi i}\oint_{\Gamma} \zeta^{-1/2}(\zeta\mathbf{I} - \mathbf{Q})^{-1}\mathbf{z}d\zeta. \tag{2.9}$$

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 if we use uniformly sampled quadrature points on a circle enclosing the spectrum 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 (2.9) and obtain quadrature points which are optimal. A thorough description of this can be found in Hale et al. (2008). We next describe this approach briefly.

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

$$\begin{aligned}\mathbf{Q}^{-1/2}\mathbf{z} &= \frac{1}{2\pi i}\oint_{\Gamma_\omega} \omega^{-2}\omega(\omega^2\mathbf{I} - \mathbf{Q})^{-1}\mathbf{z}2\omega d\omega \\ &= \frac{1}{\pi i}\oint_{\Gamma_\omega} (\omega^2\mathbf{I} - \mathbf{Q})^{-1}\mathbf{z}d\omega,\end{aligned} \tag{2.10}$$

where Γ_ω is the curve resulting from the change of variables $\zeta \mapsto \omega$. Since $\mathbf{Q}$ 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 or angle preserving mapping of the Jacobi elliptic function $\omega = \lambda_{min}^{1/2}\mathrm{sn}(t|k^2)$. The mapping goes from the rectangle $(-K, K) \times (0, K')$ to $\mathbb{C}$. Here, $\mathrm{sn}(t|k^2)$ adheres to standard notation for elliptic functions (see e.g. Akhiezer (1990)) and the second co-ordinate is the

imaginary part. Moreover, $k = (\lambda_{min}/\lambda_{max})^{1/2}$, 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 uniformly on the line $0 \times (0, K')$, i.e. $0.5/n, 1 * 0.5K'/n, 2 * 0.5K'/n, \ldots (2n-1)0.5K'/n$, 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 (2.4):

$$\begin{aligned} \mathbf{Q}^{-1/2}\mathbf{z} &= -\frac{1}{\pi i} \oint_{R_-} (\mathbf{Q} - \omega^2 \mathbf{I})^{-1} \mathbf{z} d\omega \\ &\approx \sum_{j=1}^{N} \alpha_j (\mathbf{Q} - \sigma_j \mathbf{I})^{-1} \mathbf{z} \end{aligned} \tag{2.11}$$

The algorithm requires estimation of the extremal eigenvalues of Q. One should underestimate λ_{min} and overestimate λ_{max} to cover the spectrum appropriately in the quadrature. In practice, the rational approximations of the matrix inverse square root seem to be fairly robust in perturbing λ_{min}, λ_{max}. We tested several approximations, some really coarse, and we did not lose much in accuracy. Moreover, the number of terms, N, in the rational approximations (2.4) 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.

Theorem 4. *Let* $\mathbf{Q}$ *be a real or complex matrix with spectrum contained in* $[\lambda_{min}, \lambda_{max}]$. *Then the rational approximations* (2.4) *with coefficients computed by quadrature converge to* $\mathbf{Q}^{-1/2}$ *at the rate*

$$\|\mathbf{Q}^{-1/2} - f_N(\mathbf{Q})\| = \mathcal{O}(e^{\epsilon - 2\pi K N/K'}), \tag{2.12}$$

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*

$$\|\mathbf{Q}^{-1/2} - f_N(\mathbf{Q})\| = \mathcal{O}(e^{-2\pi^2 N/[\log(\lambda_{max}/\lambda_{min})+3]}). \tag{2.13}$$

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 $\|\mathbf{Q}^{-1/2}\mathbf{z} - f_N(\mathbf{Q})\mathbf{z}\| \leq \|\mathbf{Q}^{-1/2} - f_N(\mathbf{Q})\|\|\mathbf{z}\|$, and therefore the theorem holds for functions of a matrix times a vector as well.

For computing $(\omega^2\mathbf{I} - \mathbf{Q})^{-1}\mathbf{z}$ or equivalently $-(\mathbf{Q} - \omega^2\mathbf{I})^{-1}\mathbf{z}$ to identify ω and σ_i in (2.4), we employ versions of the conjugate gradient algorithms. Since ω^2 lies on the negative real axis, the condition number of the linear system gets smaller with decreasing ω^2 and this works as a stabilising agent for the sampling algorithm. For the tolerance of the iterative solvers, we use the approximation given by (2.13), divided by N. This can be regarded as a tuning parameter in the algorithms. Note that the α_j and σ_j values can be obtained before Krylov methods are employed. A succinct MATLAB implementation of the required conformal maps is method3.m in Hale et al. (2008).

2.2.4 Conjugate gradients for multiple shifts

The rational approximations have the property that all the systems that need to be solved are shifts of the initial system $\mathbf{Qx} = \mathbf{z}$. In relation to Krylov methods we have $\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0) = \mathcal{K}_m(\mathbf{Q} - \sigma_j, \mathbf{r}_0)$, 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 $\mathbf{Q} - \sigma_j\mathbf{I}$ 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 $f_N(\mathbf{Q})\mathbf{z} \approx \mathbf{Q}^{-1/2}\mathbf{z}$. The main advantage of using this strategy is that the

cost of producing a sample is essentially the cost of solving one linear system.

Note that the shift in Algorithm 8 only appears in the computation of ζ^{σ}_{new}, and the corresponding coefficients are those that would be obtained from running CG on the shifted systems. Algorithm 7 (with no deflated vectors) and Algorithm 8 are equivalent because of the invariance of Krylov subspaces under shifts. Algorithm 8 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 which can be seen in Section 2.3.

2.2.5 Factored preconditioned sampling and alternative inverse square root iterations

In this section we outline an extension that is not part of the timings presented in Section 2.3, using factored preconditioners. Additionally, we outline alternative iterations that may be useful when the Krylov subspace required for convergence is particularly small.

While preconditioning can be difficult in general, there exists a trick that may be useful. This trick requires a factored preconditioner, $\mathbf{M}\mathbf{M}^T \approx \mathbf{Q}^{-1}$. Now, suppose that $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{M}^{-T}\mathbf{Q}^{-1}\mathbf{M}^{-1})$. Then, $\mathbf{M}^T\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$, since $\mathrm{Cov}(\mathbf{M}^T\mathbf{x}) = \mathbf{M}^T\mathbf{M}^{-T}\mathbf{Q}^{-1}\mathbf{M}^{-1}\mathbf{M} = \mathbf{Q}^{-1}$. What this essentially means is that we replace the matrix vector product $\mathbf{Q}\mathbf{v}$ in the Lanczos process or CG for multiple shifts by $\mathbf{M}\mathbf{Q}\mathbf{M}^T\mathbf{v}$ and follow up the sample generated by this, $\mathbf{x}$, with $\mathbf{M}^T\mathbf{x}$.

Factored preconditioners are, however, a much smaller class than the one that only requires a matrix vector product for its application. All the ones relying on a spectral factorisation are, however, in this smaller class, including the FFT circulant preconditioner (Tyrtyshnikov, 1990), the discrete cosine transform types (Chan et al., 1999), and many others. Factored approximate

Algorithm 8 CG-M for $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$

Input: $\mathbf{Q}$, shifts $\sigma_k, k = 1, \ldots, N$, quadrature weights $w_k, k = 1, \ldots, N$ and convergence tolerance
Output: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$
Set: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{r} = \mathbf{z}, \mathbf{p} = \mathbf{r}, \beta_{old} = 1, \alpha = 0, c_{cur} = \langle \mathbf{r}, \mathbf{r} \rangle, \mathbf{x}^\sigma = \mathbf{0}, \zeta^\sigma_{cur} = 1, \zeta^\sigma_{old} = 1, p^\sigma = z$
for $j = 1$ to m or hardest system converged **do**
  $\mathbf{p}_Q = \mathbf{Q}\mathbf{p}$
  $\beta_{cur} = -\frac{c_{cur}}{\langle \mathbf{p}, \mathbf{p}_Q \rangle}$

  for $k = 1$ to n_σ **do**
    $\zeta^k_{new} = \beta_{old} \frac{\zeta^k_{cur} \zeta^k_{old}}{\beta_{cur}\alpha\left(\zeta^k_{old} - \zeta^k_{cur}\right) + \beta_{old}\zeta^k_{old}(1 - \sigma_k \beta_{cur})}$
    $\beta^k = \beta_{cur} \frac{\zeta^k_{new}}{\zeta^k_{cur}}$
    $\mathbf{x}^k = \mathbf{x}^k - \beta^k \mathbf{p}^k$
  end for

  $\mathbf{r} = \mathbf{r} + \beta_{cur} \mathbf{p}_Q$
  $c_{new} = \langle \mathbf{r}, \mathbf{r} \rangle$
  $\alpha = \frac{c_{new}}{c_{cur}}$
  $\mathbf{p} = \mathbf{r} + \alpha \mathbf{p}$

  for $k = 1$ to n_σ **do**
    $\alpha^k = \alpha \frac{\zeta^k_{new} \beta^k}{\beta_{cur} \zeta^k_{cur}}$
    $\mathbf{p}^k = \zeta^k_{new} \mathbf{r} + \alpha^k \mathbf{p}^k$
  end for

  Set: $\zeta^\sigma_{old} = \zeta^\sigma_{cur}, \zeta^\sigma_{cur} = \zeta^\sigma_{new}, \beta_{old} = \beta_{cur}$ and $c_{cur} = c_{new}$
end for

Set: $\mathbf{x} = \mathbf{0}$
for $k = 1$ to n_σ **do**
  $\mathbf{x} = \mathbf{x} + w_k \mathbf{x}^k$
end for

sparse inverse preconditioners (Benzi et al., 1996) are also in the admissible class, and can be combined with wavelet compression (Chan et al., 1997).

Alternative inverse square root iterations

Typically, when preconditioners are applied, the number of iterations needed for the Krylov method to converge is less than 200, and it may happen in other cases as well. When the the dimensionality of $\mathcal{K}_m$ is small, it is possible to exploit other methods for computing the matrix inverse square root for the tridiagonal matrix $\mathbf{T}_m$, containing coefficients obtained from the Lanczos process. This is possible since the computational time required for this inverse square root is small compared to the matrix-vector product, $\mathbf{Qv}$. We mention a particular scheme alluded to in Higham (2008).

First, note that any full-rank square matrix, $\mathbf{A}$ has a unique polar decomposition, $\mathbf{A} = \mathbf{UH}$, with $\mathbf{U}$ unitary and $\mathbf{H}$ positive definite, with $\mathbf{H} = (\mathbf{A}^*\mathbf{A})^{1/2}$. For the tridiagonal matrix $\mathbf{T}_m$, compute the bidiagonal Cholesky decomposition, $\mathbf{L}\mathbf{L}^T = \mathbf{T}_m$, and set $\mathbf{L}^T = \mathbf{UH}$. Now, compute $\mathbf{L}^{-T}\mathbf{L}^T = \mathbf{L}^{-T}\mathbf{UH}$, so that $\mathbf{H}^{-1} = \mathbf{L}^{-T}\mathbf{U}$. Since $\mathbf{H} = (\mathbf{L}\mathbf{L}^T)^{1/2}$, $\mathbf{L}^{-T}\mathbf{U} = \mathbf{T}_m^{-1/2}$. Computing the polar decomposition of $\mathbf{L}^T$ can be done by Algorithm 8.20 in Higham (2008), which is an accelerated Newton algorithm. In this algorithm, we need to approximate $\|\cdot\|_1$ and $\|\cdot\|_\infty$ for scaling the algorithm properly. Estimating these matrix norms is cheap, using for instance a block algorithm, such as the one found in Higham and Tisseur (2000).

Indeed, this observation leads to another method for computing $\mathbf{Q}^{-1/2}\mathbf{z}$ that we have not discovered in the literature. It is possible to compute the matrix-vector product $\mathbf{Uv}$ or $\mathbf{U}^T\mathbf{v}$ by noting that

$$\operatorname{sign}\begin{pmatrix} & \mathbf{L} \\ \mathbf{L}^T & \end{pmatrix} = \begin{pmatrix} & \mathbf{U} \\ \mathbf{U}^T & \end{pmatrix}, \tag{2.14}$$

where $\mathbf{L}$ is the Cholesky factor of $\mathbf{T}_m$. Using the nested Krylov subspace method in Bloch and Heybrock (2009), we have an efficient method for computing the required matrix-vector product. The method is now doubly

nested – one Krylov method for computing $\mathbf{T}_m$, and a nested one for computing $\mathbf{Uv}$, where $\mathbf{L}^{-T}\mathbf{U} = \mathbf{T}_m^{-1/2}$ as above. Note that the conditioning of the matrix in (2.14) is the square root of the original $\mathbf{T}_m$, effectively leading to an algorithm which converges in fewer than m Krylov iterations on this extended matrix.

It is also possible to use the matrix sign function directly, noting the identity

$$\text{sign}\begin{pmatrix} & \mathbf{Q} \\ \mathbf{I} & \end{pmatrix} = \begin{pmatrix} & \mathbf{Q}^{1/2} \\ \mathbf{Q}^{-1/2} & \end{pmatrix}. \tag{2.15}$$

Since $\mathbf{Qv}$ for some $\mathbf{v}$ needs to be computed only once in each Krylov iteration, and $\mathbf{Iv}$ is free, we may use the method in Bloch and Heybrock (2009). This may lead to a highly efficient sampling algorithm. It is also ameanable to factored preconditioning, replacing $\mathbf{Q}$ with $\mathbf{MQM}^T$ in (2.15). We need to double the vector storage cost over that of the Lanczos method to use this strategy.

2.2.6 Continuous deformation method

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

$$d\mathbf{z}/dt = r(\mathbf{Q}-\mathbf{I})[t(\mathbf{Q}-\mathbf{I})+\mathbf{I}]^{-1}\mathbf{z},\ t \in [0,1], \tag{2.16}$$

with $\mathbf{z}(0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $r = -1/2$. The solution at the endpoint, $\mathbf{z}(1) = \mathbf{x}$ is a sample from $\mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$.This continuous deformation approach is one of the methods for the (inverse) square-root mentioned in Higham (2008). One can show that this method gives correct sampling by considering the eigen decomposition of the system as in Allen et al. (2000). Let $\mathbf{V}, \mathbf{\Lambda}$ be the eigenvectors and eigenvalues of $\mathbf{Q}$ respectively, and let further $\mathbf{z}(t) = \sum_j \alpha_j(t)\mathbf{v}_j$, where $\mathbf{v}_i$ is the i'th column of $\mathbf{V}$. Inserting this representation in (2.16) and taking the $i'th$ component gives

$$\frac{d}{dt}\alpha_i(t)\mathbf{v}_i = r\alpha_i(t)(\mathbf{Q}-\mathbf{I})[t(\mathbf{Q}-\mathbf{I})+\mathbf{I}]^{-1}\mathbf{v}_i. \tag{2.17}$$

This corresponds to the equation

$$\frac{d}{dt}\alpha_i(t) = r\alpha_i(t)(\lambda_i - 1)[t(\lambda_i - 1) + 1]^{-1}, \tag{2.18}$$

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

$$\alpha_i(t) = (1 + t(\lambda_i - 1))^r \frac{\mathbf{z}(0)^T\mathbf{v}_i}{\mathbf{v}_i^T\mathbf{v}_i}. \tag{2.19}$$

Setting $t = 1, r = -1/2$, $\alpha_i(1) = \lambda_i^{-1/2}\mathbf{z}(0)^T\mathbf{v}_i/(\mathbf{v}_i^T\mathbf{v}_i)$ and summing over the $i's$, we get

$$\mathbf{z}(1) = \sum_i \lambda_i^{-1/2} \frac{\mathbf{z}(0)^T\mathbf{v}_i}{\mathbf{v}_i^T\mathbf{v}_i}\mathbf{v}_i = \mathbf{Q}^{-1/2}\mathbf{z}(0). \tag{2.20}$$

This ODE, when discretized, leads to rational approximations which are different from those of the previous section, but can be reduced to a form similar to that of (2.4). Equation (2.20) shows in an explicit way that we only need to interpolate the inverse square root on the spectrum of $\mathbf{Q}$.

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

$$\begin{aligned} \frac{d\boldsymbol{B}^{-1/2}}{dt} &= \frac{d}{dt}[(1-t)\mathbf{I} + t\mathbf{Q}]^{-1/2} \\ &= \frac{1}{2}(\mathbf{Q} - \mathbf{I})\boldsymbol{B}^{-1/2-1} \end{aligned} \tag{2.21}$$

Projecting the matrix equation onto the start vector, $\mathbf{z}(0) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ yields (2.16).

The critical points for implementing this ODE routine are: i) a good solver for $[t(\mathbf{Q} - \mathbf{I}) + \mathbf{I}]\boldsymbol{b} = \mathbf{z}$ for all $t \in [0, 1]$ and ii) an appropriate ODE solver.

Table 2.1: *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

We use Krylov methods (CG) to solve $[t(\mathbf{Q} - \mathbf{I}) + \mathbf{I}]\boldsymbol{b} = \mathbf{z}$ at every time step. The ODE is solved by MATLAB®s ODE45 discretization scheme (MATLAB (2010)). Natural tuning/accuracy parameters for this method are relative and absolute tolerances in the ODE-solver. If the matrix $\mathbf{Q}$ is badly conditioned, the ODE (2.16) is stiff (this can be taken as a definition in some settings). 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 time-stepping. An advantage of this implementation over that of the previous section is, however, that the extremal eigenvalues of $\mathbf{Q}$ need not be estimated.

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

$$\mathbf{z}'(t) = r(\mathbf{Q} - \mathbf{I})[\mathbf{I} + g(t)(\mathbf{Q} - \mathbf{I})]^{-1} g'(t)\mathbf{z}(t), \tag{2.22}$$

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 2.2: *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	$2\times$ 4	2×4	2×4
CONT-D	~ 100	~ 100	~ 80	~ 100	~ 100
CU-CG-M	7	7	8	8	8

2.2.7 Sampling from intrinsic fields

The methods we have discussed so far are predominantly suited for sampling from fields which are not intrinsic – i.e. they do not have some invariant subspace defined through eigenvectors having zero eigenvalues. Some of the methods are easy to modify in order for them to work for intrinsic fields, while others are harder to adjust. Before going into the case for iterative sampling methods, we mention that using direct methods, the most straightforward way of obtaining an intrinsic sample is through the pivoted $\mathbf{LDL}^T$-factorisation method. Let $\mathbf{Q} = \mathbf{PLDL}^T\mathbf{P}^T$, where $\mathbf{P}$ is permutation matrix, then $\mathbf{P}^T\mathbf{L}^{-T}\mathbf{D}^{\dagger/2}\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{-1})$ by direct verification, where $\mathbf{D}^{\dagger/2}$ is the the pseudo inverse square root given by $D_{ii}^{\dagger/2} = D_{ii}^{-1/2}$ if $D_{ii} > 0$ and $D_{ii}^{\dagger/2} = 0$ otherwise.

Iterative methods for computing a sample from an intrinsic field are essentially harder than ones for definite fields. This is closely related to finding suitable stopping criteria for the Krylov iterations. There is, however, a recent method developed for computing the minimal norm least squares solution for a semi-definite symmetric system. An analysis of the method, named minres-QLP, can be found in Choi et al. (2010) and Choi (2006). The strategy builds on using a pivoted QLP factorisation of the tridiagonal matrix $\mathbf{T}_m$. What we propose, is to use the minres-QLP algorithm for computing appropriate stopping conditions for the Krylov methods, and then

use rational approximations or eigen decompositions to form $\mathbf{T}_m^{-1/2}\mathbf{e}_1$. Using eigen decompositions are, naturally, more computationally demanding.

When the zero eigenvectors are known, however, it is possible to use the explicit deflation strategy in Section 2.2.2. The methods remains as it stands – simply deflate the singular vectors. It is also possible to deal with the problem by implicit deflation. More, specifically, if $\mathbf{u}_j$, $j = 1, \ldots, r$ are the eigenvectors of $\mathbf{Q}$ associated with zero eigenvalues, we orthogonalise the an i.i.d. sample $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to these eigenvectors by a Gram-Schmidt process and use the sample $\hat{\mathbf{z}} \perp \mathbf{u}_i$ in any of the algorithms described above. While this approach has sound theory, one has to be careful so that round-off errors due to loss of orthogonality do not start to dominate – this affects our stopping conditions adversely. One remedy is to orthogonalise current estimator of $\mathbf{Q}^{-1/2}\hat{\mathbf{z}}$ in the Krylov method to the known eigenvectors at regular intervals. The cost of this orthogonalisation is small.

2.2.8 GPU implementation and parallel CPU performance

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. (2010). 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, $\mathbf{Qz}$. 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.

In our view, the best candidate for GPU implementation is CG-M - both because its potential performance and it is a method that is easy for a possible end-user to utilise; no choice of deflation vectors is needed and it does not require the programming of a preconditioner.

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 (2.4) - 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.

Parallelization on a workstation with multiple CPUs is straightforward: Run the described algorithms on each core independently. Of course, using this approach, the precision matrix only need to be allocated once, but all auxiliary vectors that change need to be allocated separately for each core. There is minimal overhead using this approach, and we get a linear speedup in the number of cores with proportionality constant close to unity.

2.3 Examples

In Section 2.3.1 we present a random precision matrix model inspired by a space-time application of infectious disease count, see Paul et al. (2008). In Section 2.3.2 we consider an model for seismic data. Section 2.3.3 is a comparative study of the computation time, while Section 2.3.4 samples elastic model parameters given seismic 3D reflection data acquired at a North Sea reservoir.

The different algorithms we test are:

- **CHOL:** Cholesky sampling using approximate minimal degree reordering.
- **SEC-CG:** Sequential use of CG on each term in (2.4) using the coefficients developed in Section 2.2.3.

- **P-SEC-CG:** Preconditioned version of SEC-CG using a circulant preconditioner from a circulant approximation of the precision matrix.
- **RAT-CG-M:** Algorithm 8.
- **2pLANC:** 2-pass Lanczos sampling.
- **DEF-2pLANC:** The deflated version of 2-pass Lanczos, Algorithm 7.
- **CONT-D:** The continuous deformation method.
- **CU-CG-M:** The GPU implementation, using CUDA, of algorithm RAT-CG-M. We have used double precision arithmetic in our timings. It is a stand alone program compiled using `nvcc` with `g++-4.4` with appropriate optimization settings.

In all our comparisons, we have used a relative tolerance of 0.005 for the 2-norm of the residual vector, which also defines our stopping criterion.

There is one important aspect that must not be ignored when comparing the algorithms: are the comparisons fair? Is, for instance, one of the algorithms favoured in implementation compared to the others? We have tried to implement the algorithms on equal grounds. The deformation method (Section 2.2.6) may become faster by using a better ODE-solver for the problem at hand. We have chosen to use the fairly standard ODE45 solver (existing in MATLAB). All the methods, except the one implemented for the GPU, were implemented in MATLAB using serial matrix vector products. A point that is often mentioned for methods using loops in MATLAB is that the performance of such loops can be very bad. In these methods the cost of the loops are dominated by the matrix vector-product - a routine that is fairly well optimised in MATLAB. To test how well it is optimised, we used the linear algebra package Eigen, which actively uses lazy evaluation and expression templates to minimise overall cost of a particular set of operations. We used a discretized Laplacian on 100^3 grid points and ran 1000 matrix vector products in MATLAB and Eigen. The timings were 20 seconds using Eigen and 27 seconds using MATLAB, running the computations on a single

core, leading to a factor of 1.35. The conclusion is that there may be some overhead in the MATLAB matrix vector product.

The following hardware was used for computations: The CPU computations were done on a Core2 Duo @ 2.93 GHz with 8 Gb memory. The GPU computations were done on a nVidia Tesla c2050 with 3 Gb memory.

The timings reported in the following sections favour the GPU implementation. When deciding whether or not to use GPUs when having different hardware than what is reported here, one should check if this makes sense. A simple way of doing this is simply to time matrix-vector products for the appropriate matrix using both the CPUs in question and the GPUs. If one outperforms the other, it is probably wise to use the best performing one. If they perform similarly, it may be a good idea to use both simultaneously depending on how many samples that are required for the application at hand. Packages like Eigen and cusp make these comparisons easy to perform.

Table 2.3: *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 6.

The timings are in seconds, and represent the time required to get one sample. For all methods, the natural way to get multiple samples is to run the algorithms several times, either in parallel or sequentially. The abbreviations given in the list above are also used in the tables. The timing

starts when the i.i.d. Normal samples are generated, followed directly by the Krylov method. It therefore includes the allocation of the temporary scalars and vectors used in the method. The timing ends when is after the Krylov method finishes after it has looped over enough samples to get a good estimate for the time required to produce one sample.

On top of Table 2.1, 2.2, 2.4, 2.5, 2.7 and 2.8, $\kappa = \lambda_{max}/\lambda_{min}$ 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))

$$\|x - x_m\| \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^m \|x - x_0\| \tag{2.23}$$

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

2.3.1 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 realization of $\mathcal{N}(0, 1)$
3. Add this realization to the diagonal of Q
4. Loop until enough non-zero entries and $\mathbf{Q}$ is positive definite

This gives rise to an unstructured matrix. A similar unstructured pattern may emerge from using a non-standard spatio-temporal model (Paul et al. (2008)), where the spread of disease is simulated based on the neighbourhood pattern obtained from airline routes across the world. It has also come to our attention that this type of unstructured matrices play important roles in reconstruction in e.g. X-ray images (Björck (1996)). An illustration of

a matrix with such sparsity pattern can be found on the right in Figure 2.1. 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 2.3.2 and also with a comparable amount of non-zero entries. For these type of matrices, circulant preconditioners are inappropriate since the $\mathbf{Q}$ is completely non-stationary, and hence P-SEC-CG is not included in the comparison. For other preconditioners, P-SEC-CG could be a good choice for these type of matrices, but we have not researched such preconditioners. Note, however, that for other non-stationary processes, circulant preconditioners may work well despite of the non-stationary. See for instance Tyrtyshnikov (1990) for information on constructing potentially good preconditioners for any linear system. Incidentally, the matrices constructed by this method are extremely ill-suited for Cholesky factorizations 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 2.1 the timings of the different methods are displayed, while Table 2.2 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 summarised in Table 2.3. The number of matrix vector products is the same as for 2pLANC in Table 2.2.

The timings given in Table 2.1 show that in low dimensions, i.e. 8^3 and 16^3, the choice of 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 particularly 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

Table 2.4: *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 2.5: *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

2.1. 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 2.2, 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 $\mathbf{Q}$. 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.

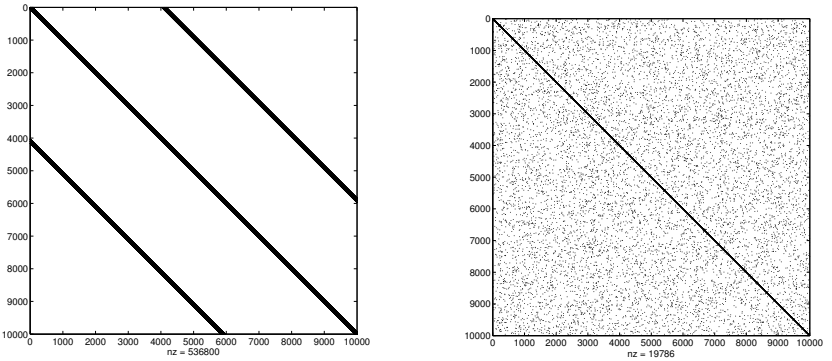
For deflation, the results are summarised in Table 2.3. It appears that deflation is not a good choice for this particular type of matrices, and the most natural explanation is that the small eigenvalues of $\mathbf{Q}$ cluster together in a relative sense.

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 (2.16).

Table 2.6: *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

Figure 2.1: *Structure of posterior seismic- (left) and random structure (right) precision matrices*



2.3.2 Seismic prior and 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 Zoepritz equations (Buland and Omre (2003) and Rabben et al. (2008)). More precisely, for reflection angle θ, and north, east and depth reference (i, j, k), we have the following model

$$\begin{aligned} y_p = f(\theta, x(i,j,k)) = \frac{1}{2\cos^2\theta_P}\frac{\triangle I_P}{\overline{I}_P}(i,j,k) - \\ 4\sin^2\theta_S\,\frac{\triangle I_S}{\overline{I}_S}(i,j,k) \\ -\frac{1}{2}\tan^2\theta_P[1-4\gamma^2(k)\cos^2\theta_P]\frac{\triangle\rho}{\overline{\rho}}(i,j,k), \end{aligned} \tag{2.24}$$

where $\frac{\triangle\cdot}{\overline{\cdot}}$ 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-

Table 2.7: *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 2.8: *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

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:

$$\mathbf{y} = \mathbf{W}(\mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}_1) + \boldsymbol{\epsilon}_2, \tag{2.25}$$

with $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathbf{0}, c_1\mathbf{Q}_1^{-1})$, $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathbf{0}, c_2\mathbf{Q}_2^{-1})$ and prior distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}^{-1})$. Here, the coefficients in $\mathbf{A}$ are obtained using (2.24), and the matrix $\mathbf{W}$ contains the convolution model. We assume that all precision matrices $\mathbf{Q}$, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are sparse Markov.

In this section, we will use a simplified version of (2.25) to compare the different sampling algorithms. We use prior mean $\mathbf{0}$, only one of the elastic parameters and 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. We do a full case posterior analysis on North Sea data in Section 2.3.4.

The matrices $\mathbf{Q}_1, \mathbf{Q}$ are constructed as follows: We use an exponential correlation function and optimise for parameters in a $3\times3\times3$-neighbourhood in the Markov graph for $\mathbf{Q}$. 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. $\mathbf{Q}_1$ 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 $\mathbf{Q}_1$. Choosing $\mathbf{Q}_2 = \mathbf{I}$ gives the posterior precision matrix for the simplified model

$$\mathbf{Q}_{post} = \mathbf{Q} + \mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}\mathbf{A}. \tag{2.26}$$

with $\mathbf{Q}_{lik} = (c_1\mathbf{W}\mathbf{Q}_1^{-1}\mathbf{W}^T + c_2\mathbf{I})^{-1}$.

Figure 2.2: *Relative differences in timings on a log-scale, random precision structure matrices*

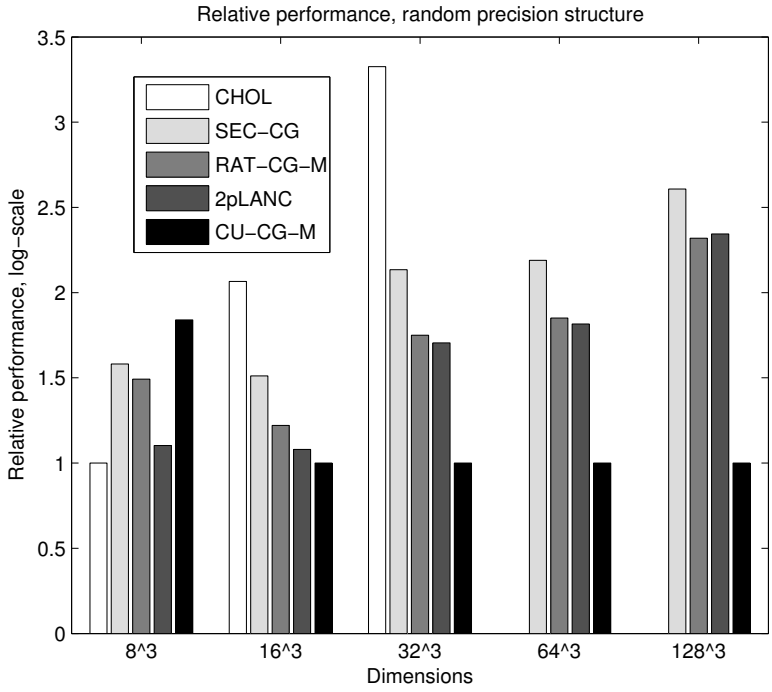


Figure 2.3: *Relative differences in timings on a log-scale, seismic prior matrix*

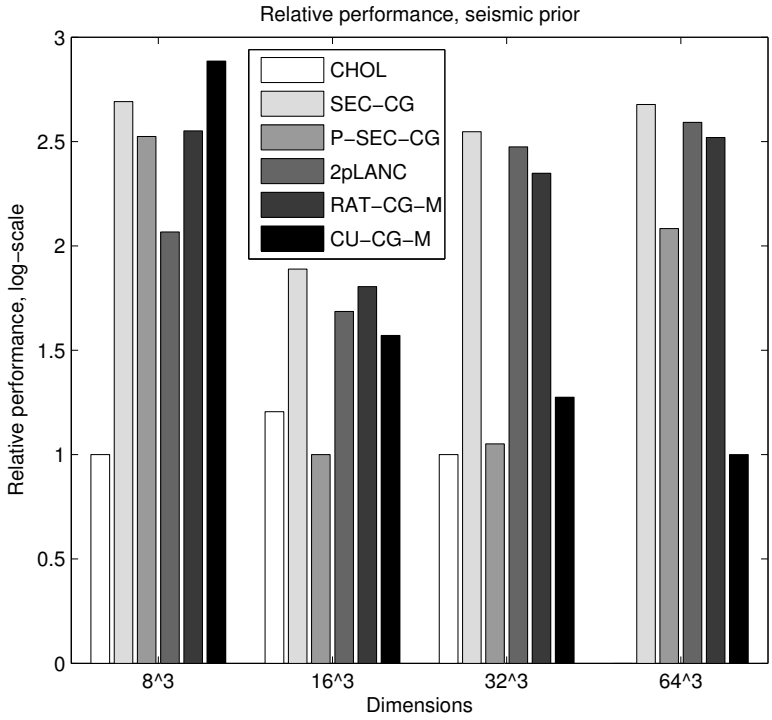
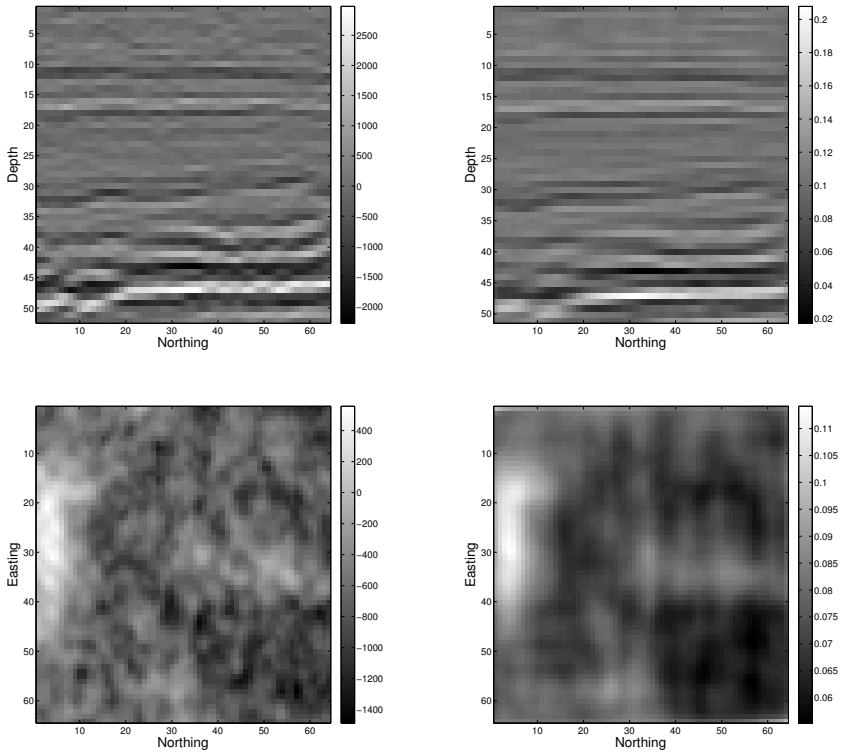


Figure 2.4: *Vertical (top) and horizontal (bottom) slice of Norne data (left) and inverted mean (right)*



Sampling results for sampling from $\mathbf{Q}$ and $\mathbf{Q}_{post}$ as in (2.26) are given in Table 2.4,2.6 and 2.7. 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 2.1. The prior precision matrix $\mathbf{Q}$ 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 2.4, 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 parallelisation can be major issues. Optimally, we could imagine a preconditioned RAT-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,

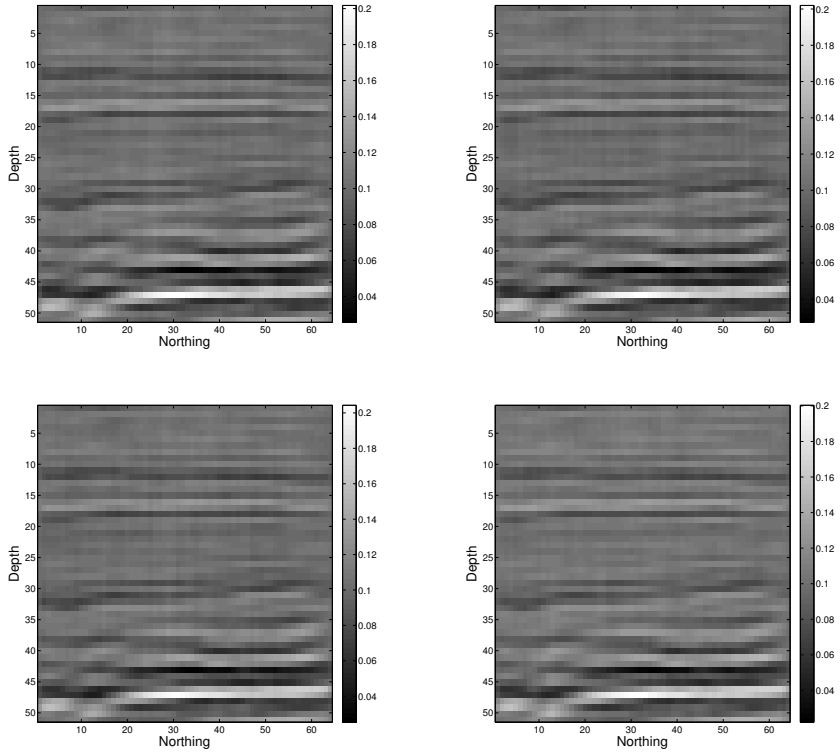
however, with both CHOL and P-SEC-CG in all cases except in dimensions 8^3, and in dimensions 64^3 it gives a speedup of 12.1x over P-SEC-CG and a overwhelming speedup 33.1x over the similar CPU implementation, RAT-CG-M. In Table 2.7 for the posterior precision matrices, the condition numbers of the matrices are a bit better and we have more fill-in due to bandwidth 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 2.3, 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 2.6 and Table 2.3, 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 to preconditioning is presented. So suppose that $K_m(\mathbf{Q}, \mathbf{r}_0) \perp \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_s$ for eigenvectors $\mathbf{w}_i$. Then whether we should deflate a new vector, $\mathbf{w}_{(i+1)}$,

Figure 2.5: *Samples from the posterior, vertical slice, Norne*



depends on the associated fraction of eigenvalues $\kappa(i+1) = \lambda_{(i+1)}/\lambda_{(i)}$, where $\lambda_{(i)} = \min(\lambda_i | \lambda_i \in \sigma(\mathbf{Q}) - \{\bigcup_{k=1}^{i-1} \lambda_{(k)}\})$, where $\sigma(\mathbf{Q})$ is the set of eigenvalues of $\mathbf{Q}$. 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 small.

2.3.3 Discussion of performance and application of the methods

While the performance of Cholesky sampling and the different Krylov methods differ between the applications, there is a basis for some general conclusions. Apart from CONT-D, the best Krylov methods have comparable performance on the CPU. Moreover, the best Krylov methods are comparable to CHOL in lower dimensions both on the CPU and the GPU. The question then is: When should we use CHOL and which Krylov method should we use if we can choose between two of comparable performance?

In 1-D applications, CHOL is most likely to outperform any of the Krylov methods mentioned here except if there are weird and non-standard couplings in the system. In 2-D, CHOL is should also outperform the Krylov methods. If one is unsure, however, if this is the case, one may run the standard CG algorithm to see how fast it may solve a linear system. If that is several times faster than doing a Cholesky sample (either on the CPU or the GPU), there is reason to investigate using one of the Krylov methods. In 3-D and higher dimensions, Krylov methods should be superior provided the number of discretization points, n, is high enough.

The dilemma one faces is illustrated on Figure 2.2 and 2.3. Here the relative differences between each of the methods are depicted on a log-scale, where a level of 1 is the fastest method for the specified dimensions and each integer above that denotes a doubling of the required time to perform a sample. These bar charts and the tables of timings and matrix vector products are considered to give guidance on what method to use for a

Figure 2.6: *Samples from the posterior, horizontal slice, Norne*

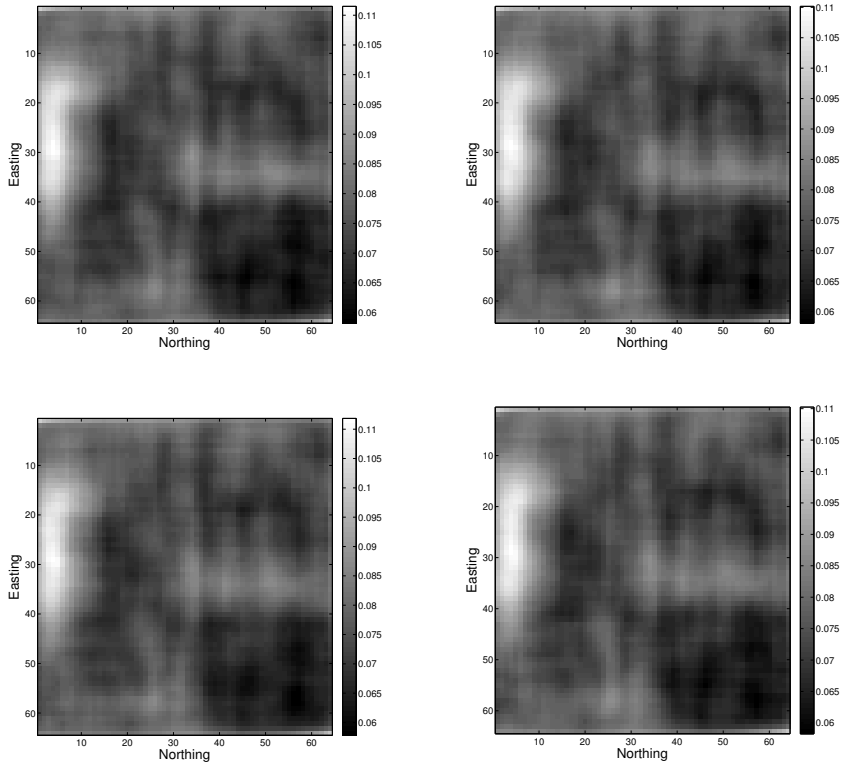
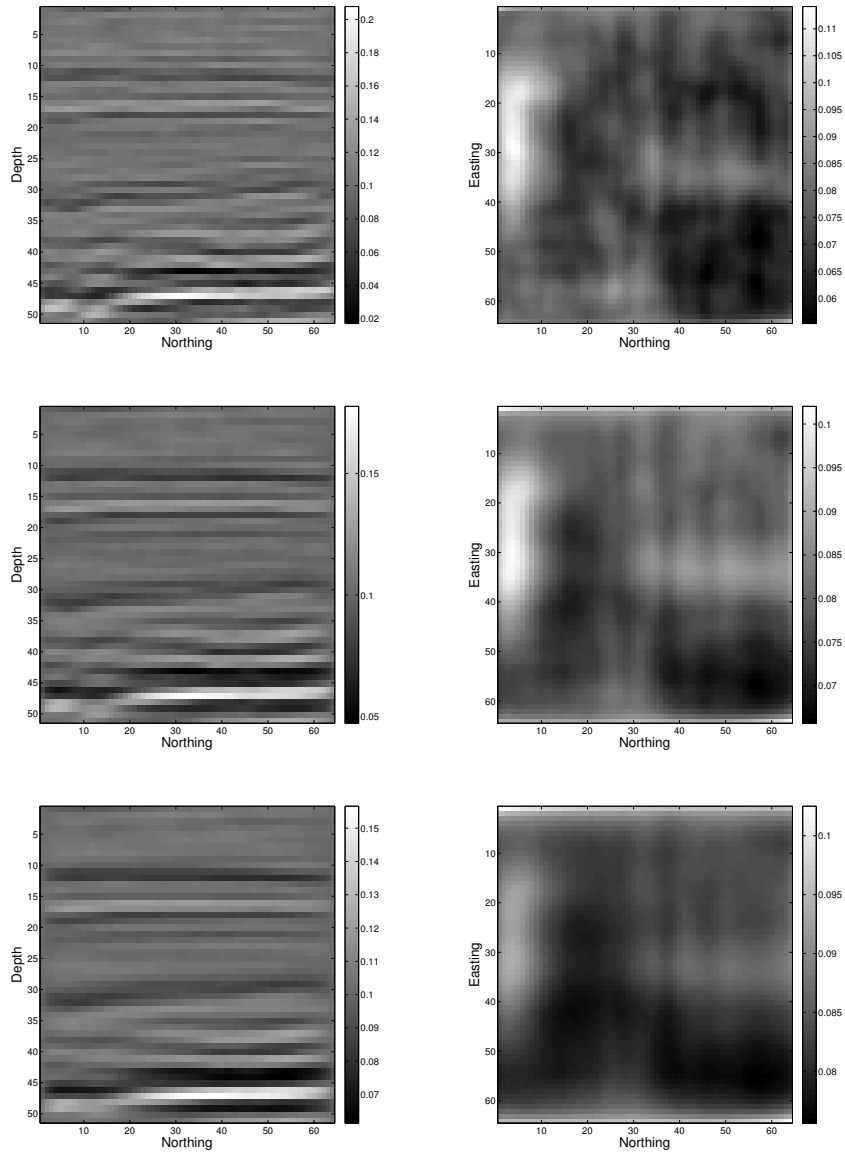


Figure 2.7: *Posterior means with different prior levels, vertical (left) and horizontal (right)*



specific application and are found below. In the bar charts, a missing bar for the CHOL means that it was impossible to perform the sampling due to high memory requirements.

The other question, mostly related to 2pLANC and RAT-CG-M (or CU-CG-M) is a matter of implementation considerations. If the Krylov method in question takes very few iterations to converge, 2pLANC or even just a pure Lanczos iteration (if one has enough memory to store some of the basis vectors) is comparable to RAT-CG-M in performance, but at the same time easier to implement; no estimation of extremal eigenvalues is required. In that case, we would recommend choosing the 2pLANC method for sampling, either on the CPU or GPU, depending on the requirements of the application and use eigen decomposition for the $\mathbf{V}\mathbf{T}^{-1/2}\mathbf{e}_1$ in 6. If on the other hand, the application requires a moderate amount of Krylov iterations, RAT-CG-M or CU-CG-M are likely to be your optimal pick. In the case where extremely many iterations are required for CG to converge, the P-SEC-CG with an appropriate preconditioner should be investigated. More research into good preconditioners for the application at hand must then be done, and that can be a huge task in itself.

The "tuning" parameters are dependent on the method in question. For the CONT-D method, the tolerance, ϵ_{CD}, for the ODE solver must be specified, and it should be set low enough for the application at hand and does not really need to be tuned. The Krylov method in CONT-D must however be tuned. An easy way to obtain a good enough relative tolerance is to run CONT-D once with very high accuracy in the Krylov method, see how many time steps, n_t it needs for convergence and use ϵ_{CD}/n_t as the relative tolerance. It can be advantageous to experiment with higher values for the relative tolerance. As of now, we see no reason to recommend the CONT-D method, but there is likely room for improvement here, using another ODE solver.

Finding the N in (2.4), can be done using Theorem 4. In practice, however, one may only need to use a fraction of that. Here it is beneficial to try to reduce the number of that obtained from Theorem 4 and see if the results

are accurate enough for a few samples. Additionally, finding N also requires estimates of $\lambda_{min}, \lambda_{max}$. Coarse under- and over estimation respectively is the recommended practice here. Specifying the relative tolerance of the Krylov method may be done exactly the same way as for CONT-D.

For 2pLANC the tuning parameter is the dimension, m, of the Krylov subspace to construct. Using the non-standard convergence criterion in Section 2.2.2 paragraph 3, this is easily done. For the deflated version, this together with deflating different numbers of vectors must be done, varying both the subspace dimension and the number of deflated vectors simultaneously.

2.3.4 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 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 $\mathbf{y}$ along with its inverted acoustic impedance can be visualised as in Figure 2.4. The model is the same as in Section 2.3.2, but here we include the full version of the $\mathbf{A}$-matrix in (2.25), and we assume correlation between the elastic parameters in the prior model. It is straight forward to construct $\mathbf{Q}$ in this situation: We have $\mathbf{Q} := \mathbf{Q} \otimes \mathbf{Q}_0$, where $\mathbf{Q}_0$ is the 3×3 precision matrix for the elastic parameters. The relevant least squares problem, then becomes

$$\begin{aligned} (\mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}\mathbf{A} + c_1\mathbf{Q})E(\mathbf{x}|\mathbf{y}) &= \boldsymbol{b}, \\ \boldsymbol{b} &= c_1\mathbf{Q}\boldsymbol{\mu} + \mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{y}, \end{aligned} \tag{2.27}$$

where $\mathbf{Q}_{lik}$ is similar to the one in equation (2.26). We solve for $E(\mathbf{x}|\mathbf{y})$, and we sample from the precision matrix given on the left side of (2.27).

The parameters in the prior mean $\boldsymbol{\mu}$, precision matrices $\mathbf{Q}_1$ and $\mathbf{Q}$, for the γ-parameter in the $\mathbf{A}$-matrix and the convolution model ($\mathbf{W}$) are typically assigned from auxiliary data. This consists of well log information from 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 2.5. In our example the horizontal slice in Figure 2.6 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 2.7. 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. However, in the multivariate setting, the number of non-zero entries in the posterior precision matrix is approximately 10 times that of the posterior generated in Section 2.3.2. In our implementation the sampling takes about 5 minutes. This computation time is too large to attempt MCMC solutions of a non-linear model, but is useful for visualizing an ensemble of seismic inversion results.

2.4 Conclusions

In this chapter we have looked at several algorithms using Krylov subspace methods combined with rational approximations for sampling Gaussians. The methods assume that matrix vector products are available at a low cost, using the sparse structure of the precision matrix or the covariance matrix. We believe that the recent numerical linear algebra techniques could be valuable in statistical applications. They provide interesting links between mathematical analysis, numerics and statistics.

The results show that our proposed methods are useful for high dimensional problems where traditional Cholesky sampling is infeasible, and that they are comparable to Cholesky sampling in lower dimensions. In addition, we have seen a considerable speedup in employing these methods in parallel on the Graphical Processing Unit (GPU). The results may lead to possibilities for sampling based inference in higher dimensional problems than have been considered previously.

Aune is currently developing a C++ library for sampling using the CG-M method and for log-determinant approximations. It is in its infancy, but can be found on `http://www.math.ntnu.no/~erlenda/KRYLSTAT/`. In this package, the CPU-implementation of CG-M is not done using parallel matrix vector product, but rather, each core produces samples independently, resulting in a linear speedup in the number of cores available.

Chapter 3

Parameter estimation in high-dimensional Gaussian distributions

In computational and, in particular, spatial statistics, increasing possibilities for observing large amounts of data leaves the statistician in want of computational techniques capable of extracting useful information from such data. Large datasets arise in many applications, such as modelling seismic data acquisition (Buland and Omre, 2003); analysing satellite data for ozone intensity, temperature and cloud formations (McPeters et al., 1996); or constructing global climate models (Lindgren et al., 2011). Most models in spatial statistics are based around multivariate Gaussian distributions, which means that random vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ has probability density function

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\eta}) =& (2\pi)^{-n/2} \det(\mathbf{Q}_{\boldsymbol{\eta}})^{1/2} \\ & \times \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{Q}_{\boldsymbol{\eta}} (\mathbf{x}-\boldsymbol{\mu})\right), \end{aligned}$$

where the mean vector is $\boldsymbol{\mu}$, and the precision matrix $\mathbf{Q}_{\boldsymbol{\eta}}$ is the inverse of the covariance matrix, which depends on the parameters $\boldsymbol{\eta}$. For short, we write $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}_{\boldsymbol{\eta}}^{-1})$. In this chapter, we assume that the precision matrix is sparse, that is, most of its entries are zero. For our purposes this

sparseness arises from a Markov property on the random vector $\mathbf{x}$, which gives computational advantages (Rue and Held, 2005). Moreover, the sparse structure also has strong physical and statistical motivations (Lindgren et al., 2011). We note that Rue and Tjelmeland (2002) showed that it is possible to approximate general Gaussian random fields on a lattice by multivariate Gaussians with sparse precision matrices.

Throughout this chapter, we will consider the common Gauss-linear model, in which our data $\mathbf{y} = (y_1, \ldots, y_{n_y})^T$ is a noisy observation of a linear transformation of a true random field, that is

$$\mathbf{y} = \mathbf{A}_{\boldsymbol{\theta}}\mathbf{x} + \boldsymbol{\epsilon}, \tag{3.1}$$

where the matrix $\mathbf{A}_{\boldsymbol{\theta}}$ connects the true underlying field $\mathbf{x}$ to observations and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}^{-1})$. We assume that $\mathbf{A}_{\boldsymbol{\theta}}$ and $\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}$ are sparse matrices. In the simplest case they are diagonal, or block diagonal. Under the Gauss-linear model assumption the conditional distribution of $\mathbf{x}$, given $\mathbf{y}$, is Gaussian with $\mathbf{x}|\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}, \mathbf{Q}_{\mathbf{x}|\mathbf{y}}^{-1})$, where $\mathbf{Q}_{\mathbf{x}|\mathbf{y}} = \mathbf{Q}_{\boldsymbol{\eta}} + \mathbf{A}_{\boldsymbol{\theta}}^T\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}\mathbf{A}_{\boldsymbol{\theta}}$ and $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} = \mathbf{Q}_{\mathbf{x}|\mathbf{y}}^{-1}(\mathbf{Q}_{\boldsymbol{\eta}}\boldsymbol{\mu} + \mathbf{A}_{\boldsymbol{\theta}}^T\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}\mathbf{y})$. Estimating the parameters, $\boldsymbol{\eta}, \boldsymbol{\theta}$, in the frequentist way amounts to maximising the following likelihood

$$p(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mu}) \propto \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mu})p(\mathbf{x}|\boldsymbol{\eta}, \boldsymbol{\mu})}{p(\mathbf{x}|\mathbf{y}, \boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mu})}. \tag{3.2}$$

In the Bayesian setting, we look at the posterior distribution of model parameters, $p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{y})$, which decomposes similarly, and we often need to compute the mode of this distribution. In both cases, we minimise the function $\Phi(\boldsymbol{\eta}, \boldsymbol{\theta}) = -2\log(f(\boldsymbol{\eta}, \boldsymbol{\theta}))$ for $f = p(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta})$ or $f = p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathbf{y})$. These expressions involve the log-determinant of matrices. When we evaluate (3.2) at the conditional mean $\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{\eta}, \boldsymbol{\theta})$, the likelihood is available as

$$\begin{aligned} 2\,\log p(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mu}) = {} & n_y \log(2\pi) + \log\det \mathbf{Q}_{\boldsymbol{\eta}} \\ & + \log\det \mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}} - \log\det(\mathbf{Q}_{\boldsymbol{\eta}} + \mathbf{A}_{\boldsymbol{\theta}}^T\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}\mathbf{A}_{\boldsymbol{\theta}}) \\ & - (\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} - \boldsymbol{\mu})^T\mathbf{Q}_{\boldsymbol{\eta}}(\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} - \boldsymbol{\mu}) \\ & - (\mathbf{y} - \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}})^T\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}(\mathbf{y} - \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}). \end{aligned} \tag{3.3}$$

Thus, the main computational requirement is the evaluation of three log-determinants, where $\log\det \mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}$ is usually trivial to compute because it is assumed to be diagonal.

We consider the situations when n and n_y are very large, say 10^6. In such high dimensions the direct determinant evaluations of the terms in (3.3) often become infeasible due to computational costs and storage limitations. For instance, the standard method of computing the determinant through the Cholesky factor is in most situations impossible due to enormous storage requirements. We suggest to use ideas from numerical linear algebra to overcome this problem, and present methods for likelihood evaluation or Bayesian computations that are useful for massive datasets. Our approach relies on fast evaluation of sparse matrix-vector products.

Previous approaches have tried to circumvent the determinant evaluation by constructing approximate likelihood models. A determinant-free approach is investigated in Fuentes (2007), based on estimated spectral densities. Pseudo-likelihood methods (Besag, 1974), composite likelihood and block composite likelihood (Eidsvik et al., 2011) combine subsets of the data to build an approximate likelihood expression. What these methods generally have in common is that they change the statistical model; i.e. they make simplifying assumptions about the model to reduce the computing dimensions. For models with long-range interactions or complex non-stationary, these approaches may be insufficient. Our approach differs from these in that we do not approximate the likelihood model, but rather approximate the log-determinant expressions directly.

In Section 3.1 we outline the main concepts behind our log-determinant evaluation and the different challenges involved. This is the methodology we have implemented for the examples in Section 3.4. In Section 3.2 we present possible solutions to these different challenges, using a number of results from numerical linear algebra, complex analysis and graph theory. Results are shown for real and synthetic datasets in Section 3.4.

3.1 Log-determinant evaluations

Precision and covariance matrices are characterised by being symmetric, positive definite; that is $\mathbf{Q} = \mathbf{Q}^T$ and for all $\mathbf{z} \in \mathbb{R}^n$, $\mathbf{z}^T\mathbf{Q}\mathbf{z} > 0$. For this class of matrices, the log-determinant can be found through the Cholesky factor of $\mathbf{Q}$ in the following manner: Let $\mathbf{Q} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is lower triangular. Then $\log\det\mathbf{Q} = 2\sum_i \log L_{ii}$. This is the most common way to compute the log-determinant. It takes only a few lines of code using a library for computing the Cholesky factor, such as CHOLMOD (Davis and Hager, 1999; Chen et al., 2008).

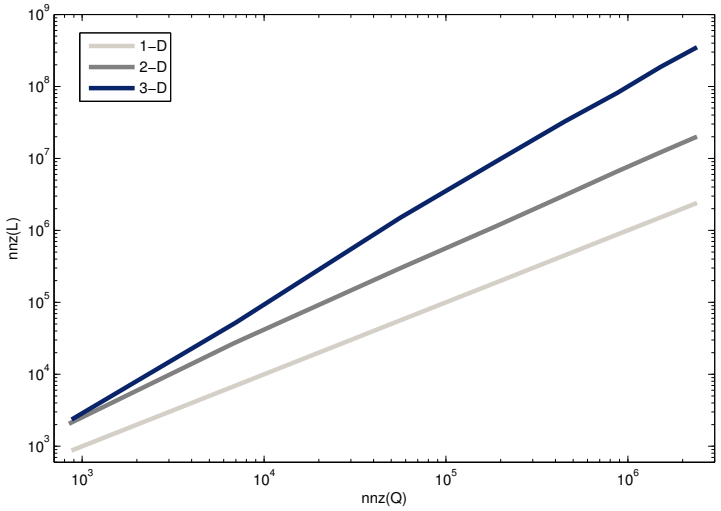
If $\mathbf{Q}$ is dense, computing $\mathbf{L}$ is an $\mathcal{O}(n^3)$ operation, and this quickly becomes infeasible for large n. If $\mathbf{Q}$ is sparse, much lower computational complexities may be obtained. In particular, if $\mathbf{x}$ is a one dimensional random field, such as a random walk or characterised through some stochastic differential equation, the computational complexity for computing $\mathbf{L}$ is $\mathcal{O}(n)$. Similarly, for a 2-D Markovian field, the complextiy is $\mathcal{O}(n^{3/2})$ and for a 3-D Markovian field $\mathcal{O}(n^2)$ (Rue and Held, 2005). These order terms are obtained after reordering the elements in the precision matrix. The fill-in is defined by the number of extra non-zero terms in $\mathbf{L}$, compared with $\mathbf{Q}$. This fill-in becomes large for higher dimensional processes. In Figure 3.1 we plot the number of non-zero entries of $\mathbf{L}$ versus that of $\mathbf{Q}$ on a log-scale. The Cholesky factor (second axis) grows quickly in 3-D, causing memory requirements to explode. The precision matrices used to display this figure come from a discretized Matérn field in 1-D, 2-D and 3-D.

We define a Matérn field to be solution to the following stochastic PDE,

$$(\kappa^2 - \triangle)^{\alpha/2} x(\mathbf{s}) = \mathcal{W}(\mathbf{s}) \tag{3.4}$$

with Neumann boundary conditions. Here, $\mathbf{s}$ is in a bounded domain, $\Omega \subset \mathbb{R}^d$, with $d = 1, 2, 3$ and $\mathcal{W}$ is Gaussian white noise. The connection of this representation and Matérn covariance functions, can be found in Lindgren et al. (2011). Our definition of a Matérn field differs from what may be found in the literature. It is usually defined as being a stationary field

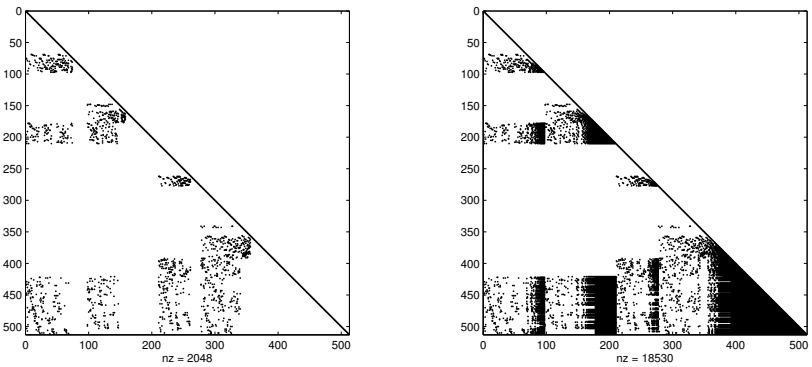
Figure 3.1: *Loglog-plot of nonzero elements in the precision matrix* $\mathbf{Q}$ *(first axis) versus nonzero elements in the Cholesky factor* $\mathbf{L}$ *(second axis). The precision matrices are constructed from a discretized Laplacian in* 1*-D,* 2*-D and* 3*-D.*



having the Matérn covariance function. Our definition does not induce a stationary covariance function, but it is locally similar to a Matérn function with appropriate parameters. Where it is appropriate in this section and Section 3.2, we use discretizations of the Matérn field in 1-D 2-D to illustrate properties of the method and its extensions. In Section 3.4, we use a 3-D Matérn field for parameter estimation. The precision matrix coming from discretizing (3.4) will be denoted $\mathbf{Q}_{\boldsymbol{\eta}} := \mathbf{Q}_{\kappa^2}$.

In Figure 3.2 an illustration of fill-in in the Cholesky factor is depicted. Here, the precision matrix $\mathbf{Q}$ of the 3-D Laplacian is used (lower triangular part of $\mathbf{Q}$ shown in left display). The lower triangular Cholesky factor $\mathbf{L}$ (right display) is obtained using METIS' nodal nested dissection reordering (Karypis and Kumar, 1999).

Figure 3.2: *Illustration of fill-in for a* 3*-D Laplacian. The black dots indicate the non-zero structure of matrices. The lower triangular part of the precision matrix (left) is very sparse, with* 2048 *non-zero elements. In contrast, the Cholesky factor (right) contains* 18530 *non-zero elements.*



In this chapter we suggest methods to overcome the prohibitive storage requirements of the Cholesky approach by using ideas from different areas of numerical mathematics, namely

- a matrix identity stating that log-determinants are equal to the tr $\log \mathbf{Q}$, where $\log \mathbf{Q}$ is the matrix logarithm,
- Cauchy's integral formula along with rational approximations for computing the logarithm of a matrix times a vector (Hale et al., 2008),
- Krylov subspace methods for solving linear systems (Saad, 2003),
- stochastic probing vectors (Hutchinson, 1989; Bekas et al., 2007; Tang and Saad, 2010).

We next outline these main concepts for evaluating log-determinants. Section 3 presents several useful extensions for practical use.

3.1.1 Determinant approximations

It appears that approximating the determinant of a large sparse matrix to sufficient accuracy is a hard problem. Nevertheless, several approximating techniques exist in the literature, the most useful of which is the approximation developed in Hutchinson (1989). The Hutchinson estimator was originally developed for calculating the trace of a matrix and it applies to our situation by the following observation:

$$\log\det\mathbf{Q} = \operatorname{tr}\log\mathbf{Q}. \tag{3.5}$$

This identity is proved using the Jordan- or eigen decomposition of the system and the cyclic property of the trace operator.

For practical implementation of this result we note the following;

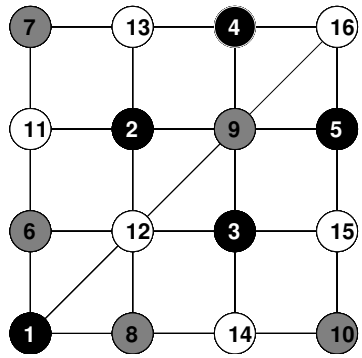
$$\operatorname{tr}\log\mathbf{Q} = \sum_{j=1}^{n}\mathbf{e}_j^T\log(\mathbf{Q})\mathbf{e}_j, \tag{3.6}$$

where $\mathbf{e}_j = (0,\ldots,1,\ldots,0)^T$ and the 1 entry is in position j. The unit vectors extract the diagonal of $\log\mathbf{Q}$ in (3.6). From this we can obtain a Monte Carlo estimator by introducing stochastic vectors $\mathbf{v}_j$ as follows: Let $\mathbf{v}_j$, $j = 1,\ldots,s$ be vectors with random entries. In position k the vector entry is defined by $P(v_j^k = 1) = 1/2$, $P(v_j^k = -1) = 1/2$, independently for all $k = 1,\ldots,n$. Next, let

$$\operatorname{tr}\log\mathbf{Q} \approx \frac{1}{s}\sum_{j=1}^{s}\mathbf{v}_j^T\log(\mathbf{Q})\mathbf{v}_j. \tag{3.7}$$

Using the Hutchinsons estimator theorem (Proposition 4.1 in Bai et al. (1996)), we recover a Hutchinson type estimator for the log-determinant. It is possible to compute confidence regions for the estimator in (3.7) since it is a Monte Carlo estimate, or we can use the Hoeffding inequality (Bai and Golub, 1997; Bai et al., 1996). This can give guidelines for choosing $s < n$. The memory requirements are low, but since this is a Monte Carlo method, the estimator requires a large s to be sufficiently accurate.

Figure 3.3: *Illustration of 1-distance colouring. Nodes sharing an edge cannot have the same colour.*



3.1.2 Probing vectors

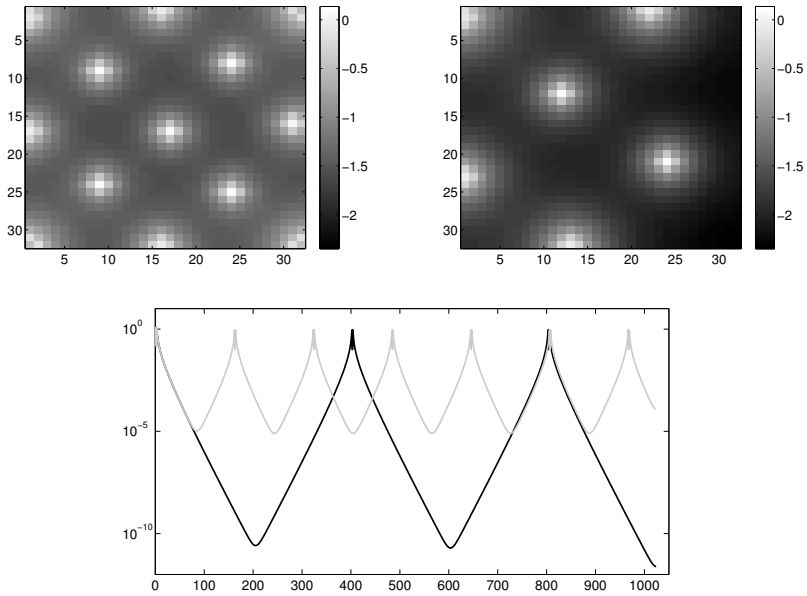
One method for keeping the number of vectors to a reasonable number is to choose the $\mathbf{v}_j$s in a clever way, so that we require far fewer vectors than a Monte Carlo method. These cleverly chosen vectors are called probing vectors. In recent publications, Bekas et al. (2007) and Tang and Saad (2010) explored the use of probing vectors for extracting the diagonal of a matrix or its inverse. Bekas et al. (2007) extract the diagonal of a sparse matrix under mild conditions. Tang and Saad (2010) relies on an approximate sparsity pattern of $\mathbf{Q}^{-1}$, determined by a power of the original matrix, i.e. $\mathbf{Q}^p$, $p = 2, 3, \ldots,$. That this is always true for large enough p can be seen using polynomial Hermite interpolation (Higham, 2008), although for large enough ps it is not necessarily practical. It turns out that if the sparsity structure of $\mathbf{Q}^{-1}$ can be approximated by that of $\mathbf{Q}^p$, then a set of probing vectors can be computed that takes this into account by using a colouring of the adjacency graph of $\mathbf{Q}^p$. If $\mathbf{Q}^{-1}$ has sufficient decay off the diagonal, say exponential, small ps are sufficient.

In this chapter, we are considering Gaussian random vectors that have an approximate *Markov property*, which, equivalently, means that their precision matrices are approximately sparse. By approximately sparse, we mean that by thresholding the matrix appropriately – i.e. setting non-zero entries below a certain magnitude to zero – it becomes sparse. We can therefore for each precision matrix associate a graph, such as the one shown in Figure 3.3. We can use this graph structure, and the idea that $\mathbf{Q}^{-1}$ or $\log(\mathbf{Q})$ can be well approximated by a matrix with the same sparsity structure as $\mathbf{Q}^p$ to design a good set of probing vectors. Ideally, we would choose $\mathbf{v}_j \equiv \mathbf{e}_j$. This is not practical due to the computational costs induced by using n vectors. We will therefore relax our requirements and chose a set of probing vectors that are sums of $\mathbf{e}_j$s. In order to not lose too much accuracy with this approximation, we need to make sure that the non-zero elements of $\mathbf{v}_j$ are sufficiently separated in some appropriate sense. Using the fact that our desired matrix function is well approximated by $\mathbf{Q}^p$, Tang and Saad (2010) suggested that a good choice of probing vectors would have the property that if both the kth and ℓth element of $\mathbf{v}_j$ were non-zero, then the (k, ℓ)-entry of $\mathbf{Q}^p$ is zero. A set of probing vectors with this property can be constructed using a graph colouring of $\mathbf{Q}^p$.

A neighbourhood colouring of the graph induced by $\mathbf{Q}^p$ associates with each node a colour, c, such that no adjacent nodes have the same colour. While constructing the optimal graph colouring is generally a difficult problem, sufficiently good colourings can often be generated easily using greedy algorithms (Culberson, 1992). Figure 3.3 illustrates the concept with three colours inducing three probing vectors. Here, the probing vectors are defined by $v^1_{1,2,3,4,5} = 1, v^2_{6,7,8,9,10} = 1, v^3_{11,12,13,14,16} = 1$, with the remaining entries equal to zero.

A heuristic method suggested in Tang and Saad (2010) is to find the power, p in $\mathbf{Q}^p$ by solving $\mathbf{Q}\mathbf{x} = \mathbf{e}_j$ and setting $p = \min\{d(l, j)||x_l| < \epsilon\}$ where $d(\cdot, \cdot)$ defines the graph distance. In our case, we may compute $\log(\mathbf{Q})\mathbf{e}_j$ and apply the same heuristic. Figure 3.4 illustrates how ones in a probing vector influence neighbors. This is illustrated on a grid, where the size is 32×32, i.e. $n = 1024$. We discuss some issues with using this kind of

Figure 3.4: *Illustration of* $\log(\mathbf{Q})\mathbf{v}_i$ *for different probing vectors using* $(\kappa^2 - \triangle)x = \mathcal{W}$. *Top right:* $\log(\mathbf{Q})\mathbf{v}_i$ *in a situation with few probing vectors in 2-D. Left: Situation with more probing vectors in 2-D. The vectors have been reshaped to fit its corresponding 2-D grid. Bottom: The same computation for the 1-D problem.*



probing vectors in Section 3.1.3, and propose a potential remedy. Note that the probing vectors need not be stored, but may be computed cheaply on the fly. If we pre-compute them, they are sparse, and do not need much storage. Since what we need for each probing vector is $\mathbf{v}_j^T \log(\mathbf{Q})\mathbf{v}_j$, we observe that the computation is highly parallel with low communication costs. On a computing cluster, each node gets one probing vector, and computes $\mathbf{v}_j^T \log(\mathbf{Q})\mathbf{v}_j$ and sends back the result. In essence, this leads to linear speed-up in the amount of processors available with proportionality close to unity.

3.1.3 Random sign flipping in probing vectors

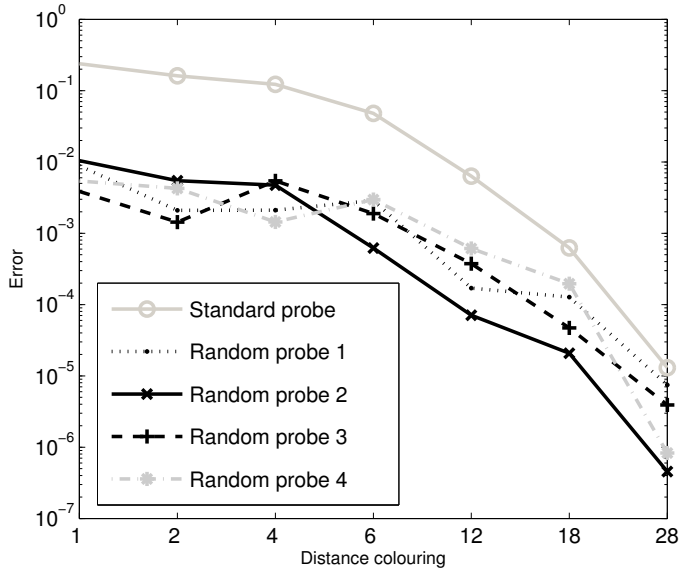
For the computed probing vectors, setting the non-zero entries to $+1$ as in Tang and Saad (2010) is not necessarily the optimal choice. Indeed, in spatial modeling, it is common to know in advance if the precision matrix induces a monotone, decreasing function off the diagonal of the covariance matrix. This is the case for Matérn type covariance functions and many other used in a wide range of spatial models (see e.g. Cressie (1993)). It may also be known in graphical models that the correlation of nodes remain positive throughout the graph. In this particular setting, it is possible to refine the probing vectors in order to achieve greater accuracy with fewer vectors. To see this, note that if $\mathbf{u}_k = -\mathbf{e}_k$,

$$\mathbf{u}_k^T f_N(\mathbf{Q})\mathbf{u}_k = \mathbf{e}_k^T f_N(\mathbf{Q})\mathbf{e}_k, \tag{3.8}$$

and we could have replaced all probing vectors with their negatives and recovered the same approximation. Now, let $\mathbf{v}_k, k = 1, \ldots, s$ be the probing vectors computed with the graph colouring approach, and let some of the entries of $\mathbf{v}_k$ be flipped to -1. We propose the following approach: If $\mathbf{v}_j(i) = 1$, set $\mathbf{v}_j(i) = -1$ with probability $1/2$. We motivate this heuristic as follows: Given a non-zero entry in a probing vector, e.g. $\mathbf{v}_j(i) = 1$, then surrounding ones in the same probing vector will all contribute positively or negatively to the entry so that $(f_N(\mathbf{Q})\mathbf{v}_j)(i) = f_N(\mathbf{Q})_{ii} + \epsilon$, where ϵ denotes errors accumulating from nearby ones. If, however, some of the surrounding ones are flipped to minus one, some of this error will cancel locally. Moreover, since we are interested in the sum of many quadratic forms $\mathbf{v}_j^T f_N(\mathbf{Q})\mathbf{v}_j$, a global cancellation also occurs. One can see this approach as a synthesis of the original Hutchinson estimator (Hutchinson (1989)), in which the vectors have entries $+1$ or -1 with probability $1/2$ and the basic probing approach in Tang and Saad (2010). It appears that this synthesis greatly improves upon the accuracy of the log-determinant approximations, which can be seen in Table 3.2.

Even though the heuristic suggested above does not immediately carry over to precision matrices inducing oscillating covariance functions, it appears

Figure 3.5: *Error of standard versus random probing vectors with flips for an oscillating covariance function.*



that using this randomised approach still gives better approximations than not using it. We illustrate this by using a stationary covariance function that oscillates and induces a sparse precision matrix. In Figure 3.5, we see the effect of using randomised probing vectors versus the standard ones. Considering these observations, it becomes quite clear that randomly flipping entries in the probing vectors should be the default behaviour for computing these log-determinant approximations. It may be that in some cases, it is possible to compute the optimal distribution of $+1$ and -1 in the probing vectors, but how to do this is not obviously clear in all situations. The randomised version is therefore a good default choice. The observations made here also suggests that randomised probing vectors is compatible with the wavelet approach discussed in Section 3.2.1.

3.1.4 Computing $\log(\mathbf{Q})\mathbf{v}_j$

The procedure described above requires the evaluation of $\log(\mathbf{Q})\mathbf{v}_j$. The matrices we consider have real positive spectrum, and it is possible to evaluate $\log(\mathbf{Q})\mathbf{v}_j$ through Cauchy's integral formula,

$$\log(\mathbf{Q})\mathbf{v}_j = \frac{1}{2\pi i}\oint_{\Gamma} \log(z)(z\mathbf{I} - \mathbf{Q})^{-1}\mathbf{v}_j dz, \tag{3.9}$$

where Γ is a suitable curve enclosing the spectrum of $\mathbf{Q}$ and avoiding branch cuts of the logarithm. Discretizing this integral leads to a rational approximation of $\log(\mathbf{Q})\mathbf{v}$ of the following form

$$\log(\mathbf{Q})\mathbf{v}_j \approx f_N(\mathbf{Q})\mathbf{v}_j = \sum_{l=1}^{N} \alpha_l(\mathbf{Q} - \sigma_l\mathbf{I})^{-1}\mathbf{v}_j, \qquad \alpha_l, \sigma_l \in \mathbb{C}, \tag{3.10}$$

where typically $N < 20$ in our case, and α_l, σ_l, $l = 1, \ldots, N$ are integration weights and shifts respectively.

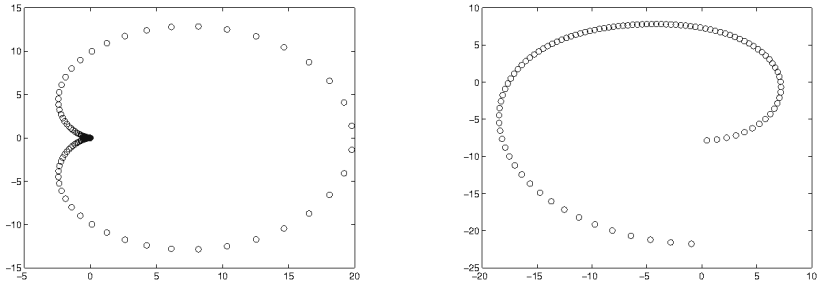
Davies and Higham (2005) show that direct quadrature on (3.9) can be extremely inefficient, but through clever conformal mappings, Hale et al. (2008) developed midpoint quadrature rules that converge rapidly for increasing number of quadrature points. The maps needed depend on the extremal eigenvalues of the matrix $\mathbf{Q}$ and therefore need to be estimated. An example of the contour and shift produced by this method is illustrated in Figure 3.6. For the quadrature rules resulting from these mappings, the following theorem holds

Theorem 5. *(Hale et al., 2008) Let* $\mathbf{Q}$ *be a positive definite matrix with eigenvalues in* $[\lambda_{min}, \lambda_{max}]$*, then the* N*-point discretization formula developed in Hale et al. (2008) (equation* 3.2*) converges at the following rate*

$$\|\log \mathbf{Q} - f_N(\mathbf{Q})\| = \mathcal{O}(e^{-2\pi N/(\log(\lambda_{max}/\lambda_{min})+6)}) \tag{3.11}$$

with f_N *as in* (3.10)*.*

Figure 3.6: *Contours, the α_js, (left) and shifts, the σ_js, (right) for f_N in* (3.10)*, with $N = 200$ (note that typically $N \leq 20$).*



By the inequality $\| \log(\mathbf{Q})\mathbf{v} - f_N(\mathbf{Q})\mathbf{v}\| \leq \| \log \mathbf{Q} - f_N(\mathbf{Q})\| \|\mathbf{v}\|$, the theorem holds for functions of a matrix times a vector as well. This theorem can be used to determine the needed number of terms, N, in (3.10) required to achieve a certain accuracy. The conformal maps required for computing this quadrature rule, require the evaluation of the Jacobi elliptic functions. These functions are in general difficult to compute. We use an approach similar to that in Driscoll (2009) to compute them.

The approximation of $\log(\mathbf{Q})\mathbf{v}$ in (3.10) is based on solving a family of shifted linear systems. The method of choice for computing $f_N(\mathbf{Q})\mathbf{v}_j$ is problem dependent, but in high dimensions, we usually have to rely on iterative methods, such as Krylov methods. Conjugate gradients (CG) is the most famous such method for solving $\mathbf{Q}\mathbf{x} = \mathbf{v}$, for a sparse $\mathbf{Q}$. This method solves for $\mathbf{x}$ by iteratively computing $\mathbf{Q}\mathbf{w}$, many times, for different $\mathbf{w}$. Generally, a Krylov subspace, $\mathcal{K}_k(\mathbf{Q}, \mathbf{v})$ is defined by $\mathcal{K}_k(\mathbf{Q}, \mathbf{v}) = \text{span}\{\mathbf{v}, \mathbf{Q}\mathbf{v}, \mathbf{Q}^2\mathbf{v}, \ldots, \mathbf{Q}^{k-1}\mathbf{v}\}$, see e.g. Saad (2003). The Krylov method of choice is highly dependent on the condition number $\mathfrak{K}(\mathbf{Q}) = \lambda_{max}/\lambda_{min}$ of $\mathbf{Q}$, and the performance can often be improved by preconditioning the matrix $\mathbf{Q}$. The convergence of Krylov methods depends explicitly on the condition number of the matrix, with large values having an adverse effect on the the iterations needed for convergence, and small values – the closer

to 1 the better – are essentially good (Saad, 2003).

If the condition number $\mathfrak{K}(\mathbf{Q})$ is relatively small, there are Krylov methods that are particularly well suited to compute the approximation in (3.10). These methods are based on the fact that $\mathcal{K}_k(\mathbf{Q}, \mathbf{v}) = \mathcal{K}_k(\mathbf{Q} - \sigma_l \mathbf{I}, \mathbf{v})$ for any $\sigma_l \in \mathbb{C}$. This means that we can obtain the coefficients for the shifted systems in (3.10) without computing new matrix-vector products, see Jegerlehner (1996) and Frommer (2003) for details. We have employed the method CG-M in Jegerlehner (1996) for our implementation. One possible difficulty in employing the method is that we have complex shifts - this is remedied by using a variant, Conjugate Orthogonal CG-M (COCG-M), which entails using the conjugate symmetric form $(\overline{\mathbf{v}}, \mathbf{y}) = \mathbf{v}^T\mathbf{y}$ instead of the usual inner product $(\mathbf{v}, \mathbf{y}) = \overline{\mathbf{v}}^T\mathbf{y}$ in the Krylov iterations. See van der Vorst and Melissen (1990) for a description of the COCG method. In practice, little complex arithmetic is needed, since the complex, shifted coefficients are computed from the real ones obtained by the CG method used to solve $\mathbf{Q}\mathbf{v} = \mathbf{y}$. Note that for large $\mathfrak{K}$, this particular method may have poor convergence behaviour and it is difficult to precondition the COCG-M method. In these cases, one is better of by solving the shifted systems in (3.10) in sequence using good preconditioners for $\mathbf{Q} - \sigma_l \mathbf{I}$.

3.1.5 Subtractive cancellation for log-determinants

Suppose that the quantity of computational interest is given by

$$f(\boldsymbol{\eta}, \lambda^2) = \log\det(\mathbf{Q}_{\boldsymbol{\eta}}) - \log\det(\mathbf{Q}_{\boldsymbol{\eta}} + \lambda^2\mathbf{K}) + q(\boldsymbol{\eta}, \lambda^2) \tag{3.12}$$

for some well conditioned matrix $\mathbf{K}$, and where $q(\boldsymbol{\eta}, \lambda^2)$ is shorthand for the quadratic forms and potential prior distributions involving model parameters $\boldsymbol{\eta}, \lambda^2$. This happens when we have noisy observations of a Gaussian field and we want to find the posterior distribution or compute the maximum likelihood estimate for $\boldsymbol{\eta}, \lambda^2$, as in (3.3). When computing $f(\boldsymbol{\eta})$, it appears that $\log\det(\mathbf{Q}_{\boldsymbol{\eta}})$ is over-/underestimated while $\log\det(\mathbf{Q}_{\boldsymbol{\eta}} + \lambda^2\mathbf{K})$ is under- or overestimated comparatively. As a result, the relative error in

$\log\det(\mathbf{Q}_{\boldsymbol{\eta}}) - \log\det(\mathbf{Q}_{\boldsymbol{\eta}} + \lambda^2\mathbf{K})$ is greater than for each of the quantities individually. In the numerical litterature, this is known as subtractive cancellation. This effect may lead to problems in optimisation procedures where this difference needs to be computed for several and different parameters, $\boldsymbol{\eta}, \lambda^2$. In computational terms, it essentially means that we will need more probing vectors to accurately compute this difference than to accurately compute its individual parts.

To illustrate this effect, we utilise a 2-D Matérn field defined by (3.4) with indirect observation using iid Gaussian noise on each discretization point. If $\mathbf{Q}_{\kappa^2}$ denotes the discretized precision obtained from (3.4), the perturbed matrix becomes $\mathbf{Q}_{\kappa^2}^2 + \lambda^2\mathbf{I}$. In Table 3.1 we can see this effect, and this is typical for these type of models. The column captions in Table 3.1 and 3.2 are defined as follows

$$
\begin{aligned}
r^A_{\text{pri}} &= \frac{(\log\det\mathbf{Q}_{\kappa^2})_{\text{est}}}{(\log\det\mathbf{Q})_{\text{est}}}, \\
r^A_{\text{post}} &= \frac{\left(\log\det(\mathbf{Q}_{\kappa^2} + \lambda^2\mathbf{I})\right)_{\text{est}}}{(\log\det(\mathbf{Q}_{\kappa^2} + \lambda^2\mathbf{I}))_{\text{true}}}, \\
r^A_{\text{diff}} &= \frac{(\log\det\mathbf{Q}_{\kappa^2})_{\text{est}} - \left(\log\det(\mathbf{Q}_{\kappa^2} + \lambda^2\mathbf{I})\right)_{\text{est}}}{(\log\det((\log\det\mathbf{Q}_{\kappa^2})_{\text{est}} - \mathbf{Q}_{\kappa^2} + \lambda^2\mathbf{I}))_{\text{true}}},
\end{aligned}
\tag{3.13}
$$

where $(\cdot)_{\text{est}}$ denotes the log-determinant computed by the approximation technique, and $(\cdot)_{\text{true}}$ is the true determinant. The $(\cdot)_{\text{true}}$ can be computed exactly, since the eigenvalues for the Matérn field on a 2-D grid is known analytically, when using a standard finite-difference discretization of the Laplacian. Better conditioning of both matrices (corresponding to higher κ^2 and λ^2) leads to less subtractive cancellation and worsening of the conditioning leads to greater amounts of subtractive cancellation. Specifically, when $\kappa^2 = 0.001, \lambda^2 = 0.05$, the differences of the log-determinants are too inaccurate to perform optimisation on the parameters, and we need to have sufficient accuracy in the entire range of possible values for the parameters for an optimisation routine to stably find the correct optimum.

One possiblity for removing the subtractive cancellation effect is to use the

Table 3.1: *Relative accuracy for log-determinants of precision matrices, perturbations of these and their differences. Here we used a 14-distance colouring.*

	r^A_{pri}	r^A_{post}	r^A_{diff}
$\kappa = 0.001, \lambda^2 = 0.1$	1.01411	0.99997	0.75638
$\kappa = 0.005, \lambda^2 = 0.1$	1.00714	0.99996	0.85185
$\kappa = 0.01, \lambda^2 = 0.1$	1.00468	0.99996	0.89247
$\kappa = 0.05, \lambda^2 = 0.1$	1.00098	0.99995	0.96623
$\kappa = 0.001, \lambda^2 = 0.05$	1.01410	0.99980	0.68790
$\kappa = 0.005, \lambda^2 = 0.05$	1.00714	0.99980	0.79878
$\kappa = 0.01, \lambda^2 = 0.05$	1.00468	0.99980	0.84818
$\kappa = 0.05, \lambda^2 = 0.05$	1.00098	0.99984	0.94338
$\kappa = 0.001, \lambda^2 = 0.5$	1.01411	1.00001	0.87264
$\kappa = 0.005, \lambda^2 = 0.5$	1.00714	1.00001	0.92890
$\kappa = 0.01, \lambda^2 = 0.5$	1.00468	1.00001	0.95090
$\kappa = 0.05, \lambda^2 = 0.5$	1.00098	1.00001	0.98751

following identity

$$\begin{aligned} &\log\det(\mathbf{Q}) - \log\det(\mathbf{Q} + \lambda^2\mathbf{A}^T\mathbf{A}) \\ &\qquad = \log(\det \mathbf{Q}(\mathbf{Q} + \lambda^2\mathbf{A}^T\mathbf{A})^{-1}) \\ &\qquad = \operatorname{tr} \log(\mathbf{Q}(\mathbf{Q} + \lambda^2\mathbf{A}^T\mathbf{A})^{-1}). \end{aligned} \tag{3.14}$$

Here, we have to use an inner Krylov method to compute $(\mathbf{Q}+\lambda^2\mathbf{A}^T\mathbf{A})^{-1}\mathbf{v}_i$ for each shift in (3.10). Another advantage of using this identity, is that the off-diagonal decay of $\log(\mathbf{Q}(\mathbf{Q} + \lambda^2\mathbf{A}^T\mathbf{A})^{-1})$ may be better than for those of their respective components.

Another approach that seems to partially remove the effect of subtractive cancellation is the random sign-flipping approach discussed in Section 3.1.2. To illustrate this, we use the same model as in Section 3.1.5, and reproduce Table 3.1 and give them in Table 3.2. We also note that producing this table requires a 4-distance colouring, whereas the previous one required a 14-distance colouring, so using randomised entries in the probing vectors

Table 3.2: *Relative accuracy for log-determinants of precision matrices, perturbations of these and their differences. Now using random flipping in probing vectors. Here we used a 4-distance colouring.*

	r^A_{pri}	r^A_{post}	r^A_{diff}
$\kappa = 0.001, \lambda^2 = 0.1$	1.00262	1.00008	0.95061
$\kappa = 0.005, \lambda^2 = 0.1$	1.00227	1.00008	0.95446
$\kappa = 0.01, \lambda^2 = 0.1$	1.00200	1.00009	0.95766
$\kappa = 0.05, \lambda^2 = 0.1$	1.00113	1.00011	0.96962
$\kappa = 0.001, \lambda^2 = 0.05$	1.00262	1.00020	0.93674
$\kappa = 0.005, \lambda^2 = 0.05$	1.00227	1.00021	0.94097
$\kappa = 0.01, \lambda^2 = 0.05$	1.00200	1.00021	0.94470
$\kappa = 0.05, \lambda^2 = 0.05$	1.00113	1.00024	0.95969
$\kappa = 0.001, \lambda^2 = 0.5$	1.00262	0.99989	0.97432
$\kappa = 0.005, \lambda^2 = 0.5$	1.00227	0.99989	0.97679
$\kappa = 0.01, \lambda^2 = 0.5$	1.00200	0.99990	0.97871
$\kappa = 0.05, \lambda^2 = 0.5$	1.00113	0.99991	0.98539

both reduces the number of probing vectors required and eliminates some of the subtractive cancellation.

3.2 Potential tools for improving the log-determinant approximation

The method outlined in Section 3.1 can be used as a black-box procedure for well-conditioned matrices, for which COCG-M only requires a few iterations to converge. For poorly conditioned matrices, such as ones that require more than 500 iterations for the Krylov method to converge, the method is slow; solving hundreds of linear systems for computing one determinant approximation can be very time consuming, and therefore we should make the effort of tuning the method to the application at hand. Indeed, if it is possible to solve one set of shifted systems using fewer Krylov iterations, we should do so. Additionally, if we are able to shave off some probing vectors

for a sufficiently accurate approximation, we should do so as well.

In the following subsections we propose various extensions of the basic methodology presented in Section 3.1 to facilitate special problems that may arise. These tricks are useful both for evaluating the potential of the approach and in practical implementations. First, we give some general advice on using the proposed log-determinant approximations. This advice also applies when using the numerous extensions proposed.

The most obvious way to reduce the number of Krylov iterations for convergence, is if $\mathbf{Q}$ is in product form, $\mathbf{Q} = \prod_{i=1}^{K} \mathbf{Q}_i$. If there are repeated factors in the product, i.e. for $i_j, j = 1, \ldots, J$, $\mathbf{Q}_{i_j} = \mathbf{Q}_{i_k}$, we note that $\log\det \prod_{j=1}^{J} \mathbf{Q}_{i_j} = J \log\det \mathbf{Q}_{i_1}$, and the conditioning of $\mathbf{Q}_{j_1}$ is better than that of the product. Additionally, some matrices may have determinants that are easy to compute, such as diagonal or tridiagonal matrices and can be separated from the approximation.

To reduce the number of probing vectors, start using the approach above, looking at $\log(\mathbf{Q})\mathbf{e}_j$ for some js to find a k-distance colouring that is sufficient. Then compute the log-determinant using a $(k-1)$-distance colouring for finding the probing vectors and see if the resulting determinant is (almost) the same as for the k-distance version, in a scenario where the parameter $\boldsymbol{\eta}$ creates the largest possible condition number $\mathfrak{K}(\mathbf{Q})$. If they are, use the $(k-1)$-distance colouring instead, which should decrease the number of probing vectors by a significant amount.

If $\mathbf{Q}$ does not depend on parameters, one should obviously pre-compute it and use it in each step of the optimisation routine. This reduces the total number of log-determinant evaluations with one third for each matrix that is fixed.

While we do not do explicit parameter estimation using the extensions discussed in the following subsections, they have been tested on the issues they are meant to partially resolve.

Section 3.2.1 and 3.2.2 deal with general procedures for improving ap-

proximations that may have potential for any model, while Section 3.2.3 treats computational properties for precision matrices that come in a special factored form. Section 3.2.4 treats the case where we have an intrinsic prior.

3.2.1 Off-diagonal compression using time-frequency transforms

The most common matrix functions have the property for many precision matrices, $\mathbf{Q}$, the elements of $f(\mathbf{Q})$ decay quickly (polynomially or better) as they get farther from the diagonal (Benzi and Golub, 1999; Benzi and Razouk, 2007). However, the rate of decay often depends on the basis - that is, the elements of $f(\mathbf{WQW}^{-1}) = \mathbf{W}f(\mathbf{Q})\mathbf{W}^{-1}$ may decay faster than those of $f(\mathbf{Q})$. In our context the rate of decay is very important: the faster the diagonal elements decay, the smaller we can take p. Therefore, the efficiency of our method is intimately tied to the decay properties of $f(\mathbf{Q})$ and, in this section, we consider some options for finding a good basis $\mathbf{W}$. We remark that this can be regarded as a pre-processing step that is executed in full before applying the approximation discussed in Section 3.1.

In particular, we can change the basis through a wavelet transform. The continuous wavelet transform of a function $g \in L^2(\mathbb{R})$ is defined by through shifts and scalings of a mother wavelet $\phi \in L^2(\mathbb{R})$, namely $\phi_{u,s}(t) = \frac{1}{\sqrt{s}}\phi((t-u)/s)$, by

$$Wg(u,s) = \int_{\mathbb{R}} g(t)\overline{\phi}_{u,s}(t)dt, \tag{3.15}$$

provided that $\int_{\mathbb{R}} \phi(t)dt = 0$ and that $\int_{\mathbb{R}} |\hat{\phi}(\omega)|^2/\omega \; d\omega < \infty$. This transform can be discretized and has a fast version if g has compact support, called the fast wavelet transform. In the discretized setting, this corresponds to a basis change with another orthonormal basis (i.e. $\mathbf{W}^{-1} = \mathbf{W}^T$). This can also be generalised to multiple dimensions and on general manifolds. An introduction to wavelets can be found in Mallat (1998). Now, the properties of this transform that are interesting for our setting is exactly this: if the underlying field inducing $\mathbf{Q}$ possesses some smoothness, which

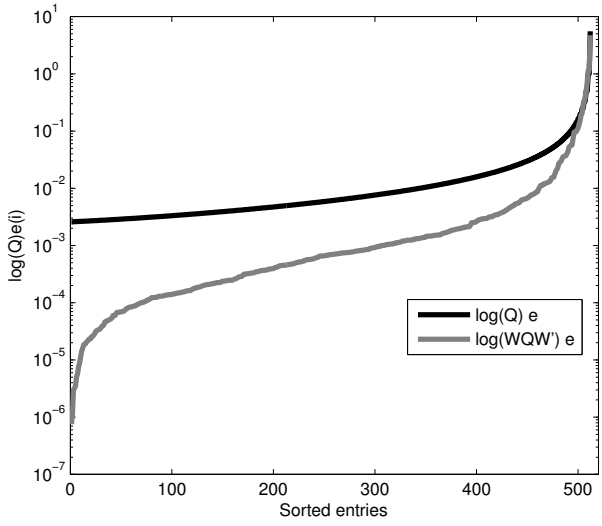
it almost always will when $\mathbf{Q}$ corresponds to a spatial prior, the entries in the transformed basis will have good decay. What we here mean by smoothness is that if its continuous operator equation (for instance an SPDE) induces local differentiability of the solution, the discretized one is also "smooth," meaning slowly varying. This also happens for the logarithm. This is essentially the compression property of wavelet bases. While we consider wavelets here, the approach naturally extends to other transforms which may compress the off-diagonal entries and at the same time has a fast transform and inverse transform. Examples include curvelet transforms and Gabor frames (see Gröchenig (2001) and Candès et al. (2006)).

In our setting, we are not interested in using this as a preconditioner for solving linear systems, as is done in e.g. Chan et al. (1997), but rather to find a basis in which we need fewer probing vectors to make a sufficiently good log-determinant approximation. Since $\mathbf{W}(\mathbf{Q} - \sigma\mathbf{I})^{-1}\mathbf{W}^T = (\mathbf{W}\mathbf{Q}\mathbf{W}^T - \sigma\mathbf{I})^{-1}$ we do not need to modify our rational approximations to accommodate this new basis. The probing vectors do, however, need to be computed with respect to the new basis, which may be difficult to facilitate in a computationally efficient way. An empirical observation, however, suggests that it may be possible to use the probing vectors computed from the original precision matrix.

To illustrate how the decay behaviour may change, we compute $\log(\mathbf{Q})\mathbf{e}_{256}$ and $\log(\mathbf{W}\mathbf{Q}\mathbf{W}^T)\mathbf{e}_{256}$ for a 1-D Matérn model defined by (3.4) using the Daubechies 2 wavelet (the compact wavelet with fewest non-zero entries with vanishing moments of order 0, 1 and 2). The result is illustrated in Figure 3.7. In this Matérn model, the $\log\det\mathbf{Q} = 2.8347 \cdot 10^3$, the 1-distance colouring in the wavelet approximation, corresponding to 27 colours gives $\log\det\mathbf{Q} \approx 2.8318 \cdot 10^3$, while the 17-distance colouring (30 colours) in the original basis gives $\log\det\mathbf{Q} \approx 2.6893 \cdot 10^3$. In the original basis, we require a 169-distance colouring, corresponding to 172 colours in order to match the approximation accuracy in the wavelet basis.

Now, computing $\mathbf{W}\mathbf{Q}\mathbf{W}^T\mathbf{v}$ for arbitrary $\mathbf{v}$s can be done without forming the matrix $\mathbf{W}\mathbf{Q}\mathbf{W}^T$ by using the fast wavelet transform, and while we need

Figure 3.7: *Decay behaviour of wavelet basis versus normal basis.* $\log(\mathbf{Q})e_{256}$ *and* $\log(\mathbf{WQW}^T)e_{256}$ *are sorted ascendingly*



to form an approximation to $\mathbf{WQW}^T$ in order to compute the probing vectors, it will be faster to use the fast wavelet transform in the matrix vector product case. This certainly suggests that for specific problems, where underlying field has some smoothness this may be an approach to pursue.

3.2.2 Nodal nested dissection reordering and recursive computation

When computing the log-determinant of a precision matrix using the Cholesky approach, we should always do a fill-in reducing reordering of the precision matrix before computing the Cholesky factor. In effect, we then compute chol($\mathbf{PQP}^T$), where $\mathbf{P}$ is a permutation matrix. A particularly well-suited reordering is the METIS nodal nested dissection reordering (Karypis and Kumar (1999)). Figure 3.8 illustrates a sparsity structure coming from employing this reordering coming from a pedigree of cows (see Gorjanc (2010) for an exposition of the model type). Pedigree matrices appears to be especially well suited for this type of reordering.

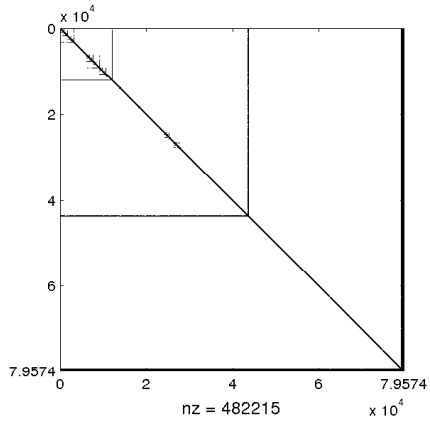
While this type of reordering allows for fill-in that is close to minimal, it also allows for recursive computation of the log-determinant via a nested Schur-complement technique. Take the following block matrix, corresponding to the general block form of a matrix that has undergone nodal nested dissection reordering,

$$\mathbf{Q} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{F}_{1,1} & \mathbf{O} & \mathbf{F}_{2,1} \\ \mathbf{F}_{1,1}^T & \mathbf{B}_1 & \mathbf{O} & \mathbf{F}_{2,2} \\ \mathbf{O} & \mathbf{O} & \mathbf{A}_2 & \mathbf{F}_{2,3} \\ \mathbf{F}_{2,1}^T & \mathbf{F}_{2,2}^T & \mathbf{F}_{2,3}^T & \mathbf{B}_2 \end{pmatrix}, \tag{3.16}$$

and let

$$\mathbf{F}_2 = \begin{pmatrix} \mathbf{F}_{2,1}^T & \mathbf{F}_{2,2}^T & \mathbf{F}_{2,3}^T \end{pmatrix}^T,$$

Figure 3.8: *Example of nested dissection reordering for a precision matrix defined by a pedigree model. The non-zero elements are very centered near the diagonal, except for a small number of variables that are coupled with almost all predecessors.*



$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{A}_1 & \mathbf{F}_{1,1} & \mathbf{O} \\ \mathbf{F}_{1,1}^T & \mathbf{B}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{A}_2 \end{pmatrix},$$

and let the block Schur complements be $\mathbf{S}_1 = \mathbf{B}_1 - \mathbf{F}_{1,1}^T \mathbf{A}_1^{-1} \mathbf{F}_{1,1}$ and $\mathbf{S}_2 = \mathbf{B}_2 - \mathbf{F}_2^T \mathbf{Q}_1^{-1} \mathbf{F}_2$. Then we can compute the log-determinant of $\mathbf{Q}$ in the following recursive manner,

$$\begin{aligned} \log\det \mathbf{Q} &= \log\det \mathbf{Q}_1 + \log\det \mathbf{S}_2 \\ &= \log\det \mathbf{A}_1 + \log\det \mathbf{S}_1 \\ &\quad + \log\det \mathbf{A}_2 + \log\det \mathbf{S}_2. \end{aligned} \tag{3.17}$$

This obviously extends to arbitrary levels of recursion, say k. The key elements in this recursive way of computing the log-determinant are 1) we can use Krylov methods to compute $\mathbf{F}_2^T \mathbf{Q}_1^{-1} \mathbf{F}_2$ and its upper level counterparts. This requires the solution of some linear systems that do not need to be stored. 2) $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_k$ are typically low dimensional, and we can use direct methods for computing their log-determinants, and 3) we can use the determinant approximations of the previous section for computing $\log\det \mathbf{A}_i$, and the condition numbers and the distance colourings required for the $\mathbf{A}_i$s are typically much smaller than for the original system.

The question then is: when do we need to use this recursive approach rather than using the matrix function approach directly? The obvious situation in which to apply this extension is when, after reordering the matrix $\mathbf{Q}$, the last block matrix $\mathbf{B}_k$ is very small, and the conditioning of $\mathbf{Q}_1$ is much better than the original $\mathbf{Q}$. Then this approach should be orders of magnitude faster than using the direct approximation on $\mathbf{Q}$, depending on how much the conditioning is improved. Another situation for which the nested dissection strategy may be prudent is when it is very hard to compute $\log\det \mathbf{Q}$ or $\log\det \mathbf{Q} - \log\det(\mathbf{Q} + \lambda^2 \mathbf{A}^T \mathbf{A})$. In this case the goal is to increase the accuracy of the log-determinant approximations without them taking much more time.

3.2.3 Model variants with particular factored precision matrices

When doing optimisation using high-dimensional determinant approximations, it is important to use whatever structure that is available in order to speed up computations. The approach outlined in this chapter is not always fast, and if it is possible optimise some aspects of computation for special models, we should do so.

In particular, for precision matrices of the kind

$$\mathbf{Q} = (\mathbf{K} + \kappa^2\mathbf{C})\mathbf{B}_1^{-1}(\mathbf{K} + \kappa^2\mathbf{C})\cdots\mathbf{B}_k^{-1}(\mathbf{K} + \kappa^2\mathbf{C}), \tag{3.18}$$

which for instance arises in the SPDE approach in Lindgren et al. (2011), it is possible to compute the partial derivative with respect to κ^2 at almost no extra cost. To see this, note the following calculations

$$\begin{aligned}
&\frac{\partial}{\partial\kappa^2}\log\det\Big((\mathbf{K} + \kappa^2\mathbf{C})\mathbf{B}_1^{-1}(\mathbf{K} + \kappa^2\mathbf{C}) \\
&\qquad\qquad \cdots\mathbf{B}_k^{-1}(\mathbf{K} + \kappa^2\mathbf{C})\Big) \\
&= \frac{\partial}{\partial\kappa^2}\Bigg((k+1)\log\det(\mathbf{K} + \kappa^2\mathbf{C}) + \\
&\qquad\qquad \sum_{j=1}^{k}\log\det\mathbf{B}_j\Bigg) \\
&= (k+1)\ \mathrm{tr}\left((\mathbf{K} + \kappa^2\mathbf{C})^{-1}\mathbf{C}\right).
\end{aligned} \tag{3.19}$$

First note that the matrix vector products, $(\mathbf{K} + \kappa^2\mathbf{C})\mathbf{v}_i$ are exactly those needed to compute the log-determinant. The trace approximation then follows directly from the diagonal inverse approximation in Tang and Saad (2010). If $\mathbf{C}\mathbf{v}_i$ is relatively cheap to compute, as happens if $\mathbf{C}$ is, e.g. diagonal, this partial derivative comes at essentially no extra cost.

Similarly, if we have an observation matrix, $\mathbf{A}$ after which i.i.d. noise is added, we need to compute $\log\det(\mathbf{Q} + \eta^2\mathbf{A}^T\mathbf{A})$. Then we obtain the

following partial derivatives

$$\frac{\partial}{\partial \kappa^2} \log\det(\mathbf{Q} + \eta^2 \mathbf{A}^T \mathbf{A}) = \operatorname{tr}\left((\mathbf{Q} + \eta^2 \mathbf{A}^T \mathbf{A})^{-1} \right.$$
$$\left. \times (k+1)\mathbf{C}(\mathbf{K} + \kappa^2 \mathbf{C})^k \prod_{i=1}^{k} \mathbf{B}_i^{-1} \right) \tag{3.20}$$

by the cyclic property of the trace operator for symmetric matrices, and

$$\frac{\partial}{\partial \eta^2} \log\det(\mathbf{Q} + \eta^2 \mathbf{A}^T \mathbf{A}) = \operatorname{tr}\left((\mathbf{Q} + \eta^2 \mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A} \right). \tag{3.21}$$

In computations, the matrix vector product $(\mathbf{Q} + \eta^2 \mathbf{A}^T \mathbf{A})\mathbf{v}$ needs to be computed for the determinant approximation. Hence $\mathbf{A}^T \mathbf{A}\mathbf{v}$ needs to be cheap if the second expression is to compute at low costs. The first of these, however, is a bit more complicated, but observe that if $k = 1$ and $\mathbf{B}_1^{-1}$ is diagonal, we can have $(\mathbf{K} + \kappa^2 \mathbf{C})\mathbf{v}$ from (3.19), provided that the probing vectors are equal, and by definition the $\mathbf{B}_1^{-1}\mathbf{v}$ is cheap to compute. Hence it is possible to compute the gradient in an optimisation routine on the fly while computing the objective function at little extra cost, and the computational requirements for a Newton-type algorithm is easily decreased to a fraction between 1/2 and 1/3 compared to the one where finite differences are used for gradient computations. We also note that these observations are compatible with the wavelet compression approach discussed in Section 3.2.1.

3.2.4 Deflation for generalised determinants

The determinant approximations described in Section 3.1 also allows for an elegant way of computing the generalised log-determinant. The need for computing this arises, for instance, if we have an essentially intrinsic (singular) precision matrix. That is, the evaluation of

$$\widetilde{\log\det}\, \mathbf{Q} = \sum_{\substack{\lambda_i \in \sigma(\mathbf{Q}) \\ \lambda_i > 0}} \log \lambda_i. \tag{3.22}$$

To do this, we will need to implicitly deflate the eigenvectors corresponding to the zero eigenvalues. More, specifically, if $\mathbf{u}_j$, $j = 1, \ldots, r$ are the eigenvectors of $\mathbf{Q}$ associated with zero eigenvalues, we orthogonalise the probing vectors $\mathbf{v}_i$ to these eigenvectors by a Gram-Schmidt process and use these new probing vectors for computing $\widetilde{\log\det}\,\mathbf{Q}$. While we need accurate approximations to these eigenvectors for this procedure to work, they are often known from the modelling assumptions (see, for example, Chapter 3 in Rue and Held (2005)).

It is also possible to use this technique if we have a small cluster of eigenvalues that are very different (on a relative scale) to the other eigenvalues. Then we use the same approach as above, but we include the eigenvalues in our determinant evaluation, which leads to $\log\det\mathbf{Q} = (\log\det\mathbf{Q})_{probe} + \sum_{j=1}^{r}\log\lambda_j$. While this approach has sound theory, one has to be careful so that round-off errors due to loss of orthogonality do not start to dominate. One remedy is to orthogonalise current estimator of $f_N(\mathbf{Q})\mathbf{v}_j$ in the Krylov method to the known eigenvectors at regular intervals. The cost of this orthogonalisation is small.

3.3 Alternative modeling for efficient log-determinant evaluations

As was mentioned in the introduction of this chapter, there are several strategies for parameter estimation that changes the model in some way in order to obtain efficient procedures. In this section, we present two alternatives that we have not seen in the literature that may be good alternatives in some cases. The first one uses a log-covariance parametrisation in order to transform computations to matrix exponentials. The particular parametrisation explored quickly recovers the Gaussian correlation function, which is inherently hard to compute with using traditional approaches. The second deals with using preconditioning techniques for generating approximate models.

3.3.1 Modeling in the logarithmic domain

A particular avenue that has not been investigated yet is the prospect of doing the covariance modeling sparsely in the logarithmic domain. What we mean by this is essentially that

$$\mathbf{G} = \log(\mathbf{\Sigma}) \tag{3.23}$$

is a sparse matrix, so that $e^{\mathbf{G}} = \mathbf{\Sigma}$. If doing the modeling by using a sparse $\mathbf{G}$ is possible, we may be in for huge computational benefits in the cases where our stochastic fields are very large and direct observations are available. For indirect observations, the savings are more moderate.

In order to make this intuitive, we start by giving one example of a class of $\mathbf{G}$s that may turn into useful models. Suppose that $\mathbf{D}_{xx}$ defines the usual finite difference approximation to the second derivative wrt. x, and that $\mathbf{D}_{yy}$ is the same wrt. y. Then define the log-domain model by

$$\mathbf{G} = \mathbf{D}_{xx} \oplus \mathbf{D}_{yy}. \tag{3.24}$$

This can be facilitated for more complicated domains in the usual fashion. A realisation from this particular model is then given by

$$\mathbf{x} = e^{0.5 \cdot \mathbf{G}} \mathbf{z}, \tag{3.25}$$

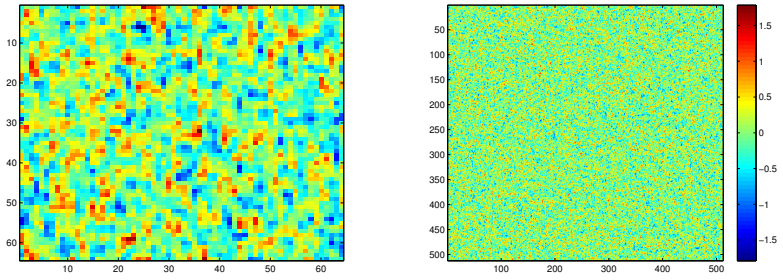
where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To see this, note simply that $\mathrm{Cov}(e^{0.5 \cdot \mathbf{G}} \mathbf{z}) = e^{0.5 \cdot \mathbf{G}} (e^{0.5 \cdot \mathbf{G}})^T = e^{\mathbf{G}} = \mathbf{\Sigma}$. In Figure 3.9, we see realisations from this particular model. The realisations may not inspire confidence, but this simple model is only the building block for ones including different correlation lengths that may depend on position and scales that may also be position dependent.

The range and scale parameters

We propose a simple parametrisation of a parameter that is both a range and differentiability parameter. This parametrisation is

$$\mathbf{G}(\alpha) = \alpha \mathbf{G}. \tag{3.26}$$

Figure 3.9: *Realisation defined by* $e^{0.5 \cdot \mathbf{G}}\mathbf{z}$. 128^2*-grid (left),* 512^2*-grid (right)*



The scale parameter has the usual interpretation, and the joint parametrisation is given by

$$\mathbf{G}(\alpha, \tau) = \alpha \mathbf{G}_0 + \tau \mathbf{I}. \tag{3.27}$$

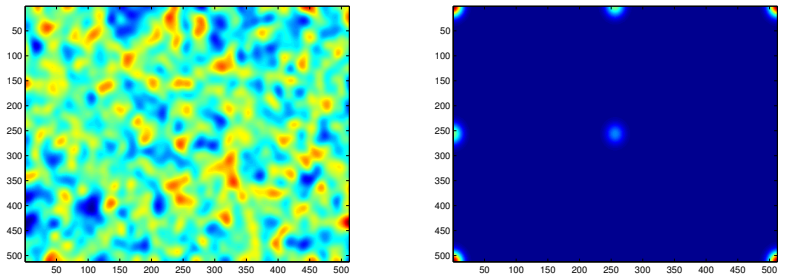
This parametrisation is for an essentially stationary field. A more general parametrisation where correlation range and scale is allowed to depend on position, $\mathbf{s}$, is given by

$$\mathbf{G}(\alpha, \tau, \mathbf{T}, \mathbf{s}) = \alpha \mathbf{K}(\mathbf{s})^{1/2} \mathbf{G}_0 \mathbf{K}(\mathbf{s})^{1/2} + \tau \mathbf{T}(\mathbf{s}). \tag{3.28}$$

The factorisation $\mathbf{K}(\mathbf{s})^{1/2}\mathbf{G}\mathbf{K}(\mathbf{s})^{1/2}$ is made for preserving symmetry of the log-covariance. A realisation from $\mathbf{G}(\alpha = 50)$ is given in Figure 3.10 on the left, with its induced covariance function in some locations on the right. Looking at the induced covariance function at some location, we see that the scalings vary similarly to using Neumann boundary conditions in the SPDE-approach in Lindgren et al. (2011). If one desires uniform marginal variances depending on location, this can be scaled in the usual way – only the implementation here is a bit simpler as you only need to add a diagonal scaling matrix.

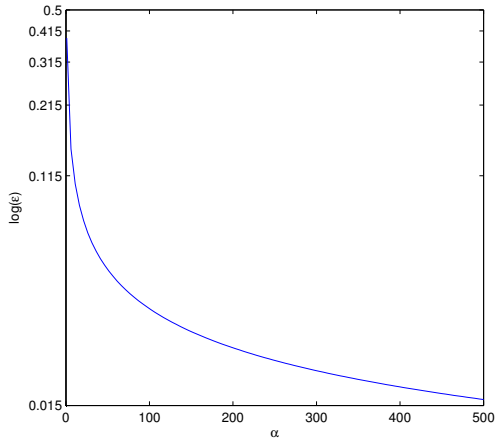
The scaling parameter has the same interpretation as in the usual setting. The range parameter, α, has a slightly different interpretation in this model

Figure 3.10: *Realisation defined by* $e^{0.5 \cdot \mathbf{G}(\alpha=100)}\mathbf{z}$ *(left), induced covariances (right)*



than, for instance, usual correlation function approaches (Cressie, 1993) or SPDE-models (Lindgren et al., 2011). In this log-covariance parametrisation, α is a fractional convolution approach for parametrising correlation length. To see this, just notice that the precision matrix $\mathbf{Q} = e^{-\alpha \mathbf{G}_0} = \left(e^{-\mathbf{G}_0}\right)^{\alpha}$, and hence, if the process generated by $\mathbf{G}(\alpha = 1)$ is k times differentiable, the one generated by $\alpha = h$ is hk times differentiable, by the convolution theorem for distributions (see e.g. Grafakos (2004)). Now, as α increases, more and more of the usual interpretation becomes valid – the correlation function approximates the Gaussian correlation function better, and increasing α then only increases the range using this correlation function. This is a consequence of the fact that if $f \propto e^{-x^2}$, $f \star f \propto e^{-kx^2}$, a fix point for convolutions – using the Fourier transform, we may get fractional entries. How $e^{\alpha \mathbf{G}_0}$ approximates the Gaussian correlation function with increasing α on a 512^2-grid is Given in Figure 3.11. Small αs really have small ranges in this parametrisation – using αs above 100 is very common, so it really approximates the Gaussian correlation function quickly. In both the machine learning and statistical literature, it has been known for a long time that computation using this type of correlation function is very unstable due to the bad conditioning of the covariance matrix. The approach outlined here makes these claims void – computation using the log-covariance parametrisation is extremely stable. Note that in the SPDE approach of

Figure 3.11: *How well $e^{\alpha \mathbf{G}_0}$ approximates the Gaussian correlation function with increasing α*



Lindgren et al. (2011), increasing the exponent of $(\kappa - \triangle)^k x = \mathcal{W}$ increases its resemblance of the Gaussian correlation function, but as k increases, the model quickly becomes too unmarkovian for it to be practical.

Computation for direct observations

While this is quite uncommon in statistics, we include derivations for this case, as it has the remarkable property that Jacobian evaluations are free. The log-likelihood for direct observations is simply

$$\begin{aligned} \log p(\mathbf{x}) &= C + \mathrm{tr}\mathbf{G}(\alpha, \tau) - (\mathbf{x} - \boldsymbol{\mu})^T e^{-\mathbf{G}(\alpha,\tau)} (\mathbf{x} - \boldsymbol{\mu}) \\ &= C + \mathrm{tr}(\alpha \mathbf{G}_0 + \tau \mathbf{I}) - (\mathbf{x} - \boldsymbol{\mu})^T e^{-\alpha \mathbf{G}_0 - \tau \mathbf{I}} (\mathbf{x} - \boldsymbol{\mu}) \\ &= C + \alpha \mathrm{tr}\, \mathbf{G}_0 + \tau n - e^{-\tau} (\mathbf{x} - \boldsymbol{\mu})^T e^{-\alpha \mathbf{G}_0} (\mathbf{x} - \boldsymbol{\mu}) \end{aligned} \tag{3.29}$$

Differentiating wrt. α and τ yields

$$\frac{\partial}{\partial \alpha} \log p(\mathbf{x}) = \operatorname{tr} \mathbf{G}_0 + e^{-\tau}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{G}_0 e^{-\alpha \mathbf{G}_0}(\mathbf{x}-\boldsymbol{\mu}) \tag{3.30}$$

$$\frac{\partial}{\partial \tau} \log p(\mathbf{x}) = n + e^{-\tau}(\mathbf{x}-\boldsymbol{\mu})^T e^{-\alpha \mathbf{G}_0}(\mathbf{x}-\boldsymbol{\mu}) \tag{3.31}$$

Since $e^{-\alpha \mathbf{G}_0}(\mathbf{x}-\boldsymbol{\mu})$ needs to be computed for the likelihood evaluation, the Jacobian is then free. The action $e^{-\mathbf{G}}\mathbf{v}$ are computed by the use of methods developed for exponential integrators (Hochbruck et al., 1998; Al-Mohy and Higham, 2011). These methods usually converge extremely fast, requiring only a few matrix-vector products.

Computation for noisy observations

Unfortunately, these nice computations do not carry over to indirect observations. For observations given by

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon} \tag{3.32}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$, we need to compute the following log-determinants and quadratic forms in the likelihood:

$$\log \det \left(e^{-\mathbf{G}} + \frac{1}{\lambda^2} \mathbf{A}^T \mathbf{A} \right) \tag{3.33}$$

$$\log \det(e^{-\mathbf{G}}) \tag{3.34}$$

$$(\boldsymbol{\mu}_{x|y} - \boldsymbol{\mu})^T e^{-\mathbf{G}} (\boldsymbol{\mu}_{x|y} - \boldsymbol{\mu}) \tag{3.35}$$

$$\sigma^{-2} (\mathbf{y} - \mathbf{A}\boldsymbol{\mu}_{x|y})^T (\mathbf{y} - \mathbf{A}\boldsymbol{\mu}_{x|y}) \tag{3.36}$$

The one expression that is not covered yet is the first of these. To compute this log-determinant, we must use the machinery developed in the earlier sections. Fortunately, this is the most well-behaved of all the log-determinants, and should be fast to approximate in usual situations. It has the drawback that the action of $e^{-\mathbf{G}}\mathbf{v}$ must be computed one more time in the inner iteration of the scheme.

The bottom line is that we essentially save one determinant evaluation, a lot of effort on the Jacobian calculations at the cost of including even more complex inner iteration computation in our method.

3.3.2 Algorithmic approximate prior models through preconditioning techniques

An aspect that, to our knowledge, has not been touched upon in the statistical literature is the use of purely algorithmic techniques for approximating the prior precision matrix. In spatial statistics, the prior precision is often not modelled through physically exact systems, but rather some heuristic about the spatial relationship between points in space. Many statisticians adopt some covariance function, often in the Matérn class, and estimates parameters for interpolation through, e.g., Kriging.

Since this prior precision or covariance matrix does not arise from physics or theory alone, but rather is some vague idea about how variable should interrelate, there should a priori be no harm in using algorithmic approximates of the models in question. Suppose, for instance, that we adopt the usual SPDE framework for defining a prior,

$$(\kappa^2 - \triangle)^{\alpha/2} x = \mathcal{W}. \tag{3.37}$$

For this there exist many preconditioning techniques that is usually used for Krylov subspace solvers. Let $\mathbf{Q}$ be the precision matrix associated with this model, and recall that the main idea for preconditioners usually is to design a matrix $\mathbf{M}$ that approximates $\mathbf{Q}^{-1}$. The matrix $\mathbf{M}$ should have favourable computational properties.

In the statistical modeling setting, the aim for $\mathbf{M}$ is different from in the literature on solving sparse linear systems. In this traditional setting, the matrix $\mathbf{M}$ is used in a Krylov subspace solver to accelerate and/or stabilise the algorithm so solve $\mathbf{Qx} = \mathbf{y}$. In our case, there are two other characteristics that are more important. Firstly, the covariance function induced by $\mathbf{M}$ should have essentially the same behaviour as the one induced by $\mathbf{Q}$,

secondly, the determinants should be good enough for optimisation purposes. For these specific purposes, it is natural to look at factored preconditioners, $\mathbf{M} = \mathbf{B}\mathbf{B}^T$, which include some incomplete Cholesky preconditioning, spectral preconditioning (e.g Tyrtyshnikov (1990) and Chan et al. (1999)) and some cases of sparse approximate inverse preconditioners (Benzi et al., 1996).

As is always true for approximate models for high-dimensional problems, it is difficult to lay forth a general procedure that will work for all observations processes and classes of priors. At the same time, however, the idea is general enough to be adapted to the individual settings.

To illustrate the use of this approach, we use the simplest model available: An SPDE model with $\alpha = 2$ in 2-D with identity observations. For this particular model, the matrices in question are $\mathbf{Q}_{\text{base}} = \mathbf{Q}_0^2 + \kappa_0^2\mathbf{Q} + \kappa_0^4\mathbf{I}$ and $\mathbf{Q} = \mathbf{Q}_{\text{base}} + \kappa^2\mathbf{I} + (1/\lambda^2)\mathbf{I}$, where κ_0^2 is the smallest range parameter admissible. We also use the modification that κ^2 defines the the range parameter only through shifts. For $\mathbf{Q}_0 + \kappa_0^2\mathbf{I}$, we define an incomplete root-free Cholesky preconditioner, $\mathbf{Q}_{\text{base}} \approx \mathbf{L}\mathbf{D}\mathbf{L}^T$, that recovers the covariance model at key locations (e.g. corners and centres) quite well, and that defines a determinant that is comparable to that of $\mathbf{Q}_0 + \kappa_0^2\mathbf{I}$. Now, instead of using the normal model, we define one that is composed of updates of the root-free decomposition for shifts, see e.g. Benzi and Bertaccini (2003) and Bellavia et al. (2011), and use wavelet compression so that we have fast decay off the diagonals in the original model. What this essentially means is that in the incomplete factorisation, we need less non-zero elements than when using the uncompressed one for approximating the model. The decomposition is as follows

$$\mathbf{W}\mathbf{Q}\mathbf{W}^T \approx \mathbf{P}\mathbf{L}\mathbf{D}\mathbf{L}^T\mathbf{P}^T, \tag{3.38}$$

where $\mathbf{W}$ denotes the wavelet transform matrix, $\mathbf{P}$ a fill-in reducing permutation matrix, $\mathbf{L}$ a lower triangular incomplete factorisation and $\mathbf{D}$ its diagonal scaling matrix. An alternative is to use an incomplete factorisation with sparsity pattern defined by the lower triangular part of $\mathbf{W}\mathbf{Q}\mathbf{W}^T$. This sparsity pattern is easy to compute without the use of transforms, but may have

quite a large number of non-zero entries compared to that of $\mathbf{Q}$. The incomplete factorisation, on the other hand, may be computed using matrix vector products only, and by using the fast wavelet transform on sparse vectors, this is quite efficient. The fact that $\mathbf{W}(\mathbf{Q}+\kappa^2\mathbf{I})\mathbf{W}^T = \mathbf{W}\mathbf{Q}\mathbf{W}^T + \kappa^2\mathbf{I}$ makes updating strategies based on $\mathbf{Q}$ equal to those based on $\mathbf{W}\mathbf{Q}\mathbf{W}^T$. Another alternative is to use an incomplete Cholesky factorisation on $\mathbf{W}\mathbf{Q}\mathbf{W}^T + \kappa_0\mathbf{I}$ with standard thresholding strategies. The challenge here is to form the matrix $\mathbf{W}\mathbf{Q}\mathbf{W}^T$, which may have very many non-zero entries, and thereafter compute its incomplete factorisation. Fortunately, this factorisation may be regarded as a preprocessing step, and we may use out-of-core strategies to form it. Once it is formed, subsequent computations may be carried out efficiently and parallelised on nodes in a cluster.

Using a thresholding parameter of 0.013 in the incomplete factorisation on a 64×64 grid using $\kappa_0^2 = 0.01$, we get $\text{nnz}(L) = 79\,977$ whereas the number of non-zero entries for the true Cholesky factorisation is $\text{nnz}(\text{Chol}(\mathbf{W}\mathbf{Q}\mathbf{W}^T))$ is 967 900, and $\text{nnz}(Q) = 51\,972$. In Figure 3.12, the true covariance functions at key locations are depicted, as well as using the approximate model. They resemble each other quite a lot. We use the shifting strategy in Bellavia et al. (2011) for our approximate model. This is given by updating the diagonal of $\mathbf{L}$ by

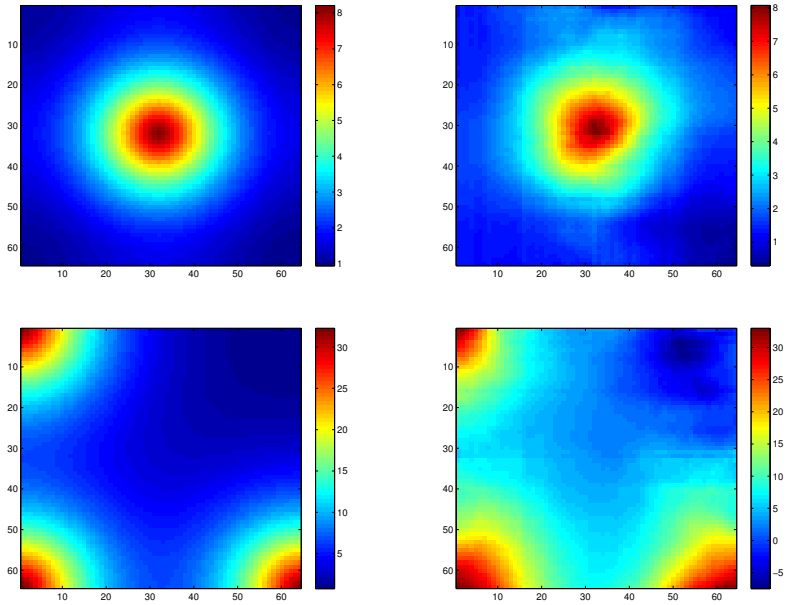
$$L_{ii} \to L_{ii} + \sqrt{1 + \frac{\kappa^2}{D_{ii}}} - 1 \tag{3.39}$$

and its off-diagonal by

$$L_{ij} \to L_{ij} + L_{ij}\left(\frac{1}{\sqrt{1 + \frac{\kappa^2}{D_{jj}}}} - 1\right) \tag{3.40}$$

The effect of using this updating strategy instead of simply adding a constant on the diagonal can be seen in Figure 3.13 to 3.15. Here we see that for small and large values of κ^2, the covariance models are very close to one another. For intermediate values, however, they differ, and we have some

Figure 3.12: *Covariance function at the center and corners for the seed model and its approximation*



spurious effect at certain locations in the field. These effects also depend on the sparsifying transform used. Here, a 1-D wavelet transform has been used – for a 2-D one, the effects are different, of course, they also depend on the wavelet itself.

While the prior fields defined through this updating strategy are different from the ones defined through adding a constant to the diagonal, they are not necessarily worse, and they can be used on problems of much higher dimensionality than the other approach. Between $\kappa^2 = 10^{-2}$ and $\kappa^2 = 1$ there are some adverse effects, like large negative correlations at specific places depending on the probing vector, but in a prediction setting, they need not be too bad.

Note that while we have used a stationary field as an example here, the approach is trivially valid in the non-stationary anisotropic setting, as well as in domains with complex geometry. Computing a basic incomplete $\mathbf{LDL}^T$ decomposition is only done once and the procedure for doing so is independent of the seed precision matrix. Having κ^2 depending on position is also not a problem.

The computational benefits of using this model are many. Recall that for inference, the following quantities and their derivatives need to be computed:

- The log-determinant of the prior precision, $\log\det(\mathbf{LDL}^T)$
- The log-determinant of the posterior precision, $\log\det(\mathbf{LDL}^T + (1/\lambda^2)\mathbf{A}^T\mathbf{A})$
- The log-determinant of the error model
- Solution of a linear system $(\mathbf{LDL}^T + (1/\lambda^2)\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{y}$
- Some quadratic forms

Computing the log-determinant of the prior precision is trivial by construction of the model, as $\log\det(\mathbf{LDL}^T) = 2\sum_i(\log L_{ii} + (1/2)D_{ii})$, is essentially free. Derivatives are cheap by construction. Solving the linear system $(\mathbf{LDL}^T + (1/\lambda^2)\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{y}$ is easy, since a good preconditioner is available. Quadratic forms are trivial, since we are dealing with preci-

Figure 3.13: *Effect of updating strategy vs. adding a constant diagonal term,* $\kappa^2 = 10^{-4}$ *and* $\kappa^2 = 10^{-3}$

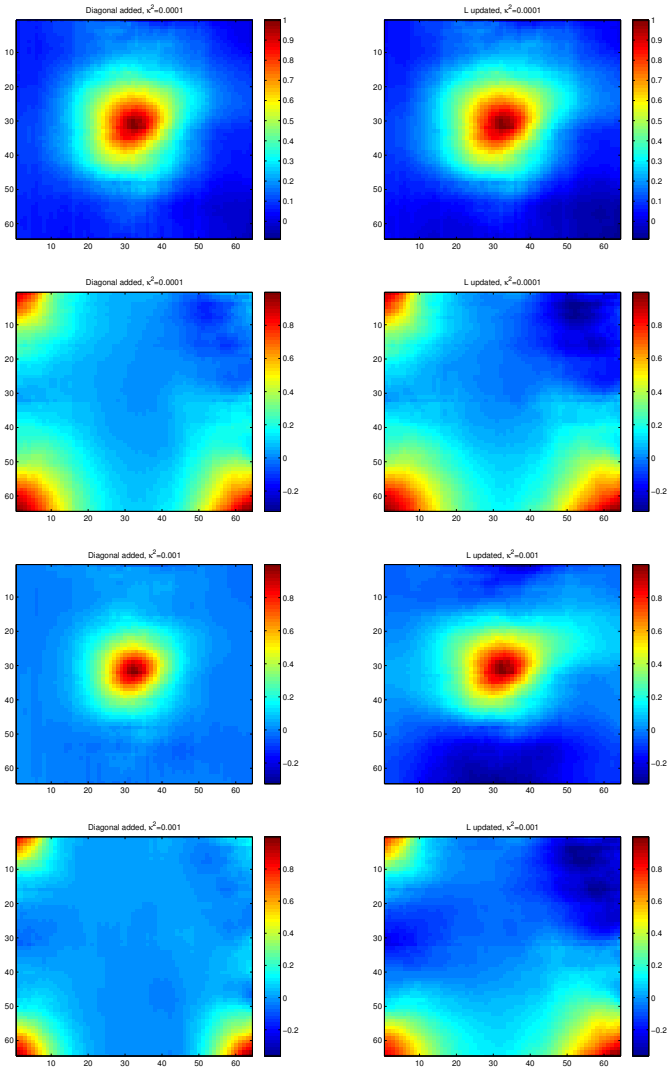


Figure 3.14: *Effect of updating strategy vs. adding a constant diagonal term,* $\kappa^2 = 10^{-2}$ *and* $\kappa^2 = 10^{-1}$

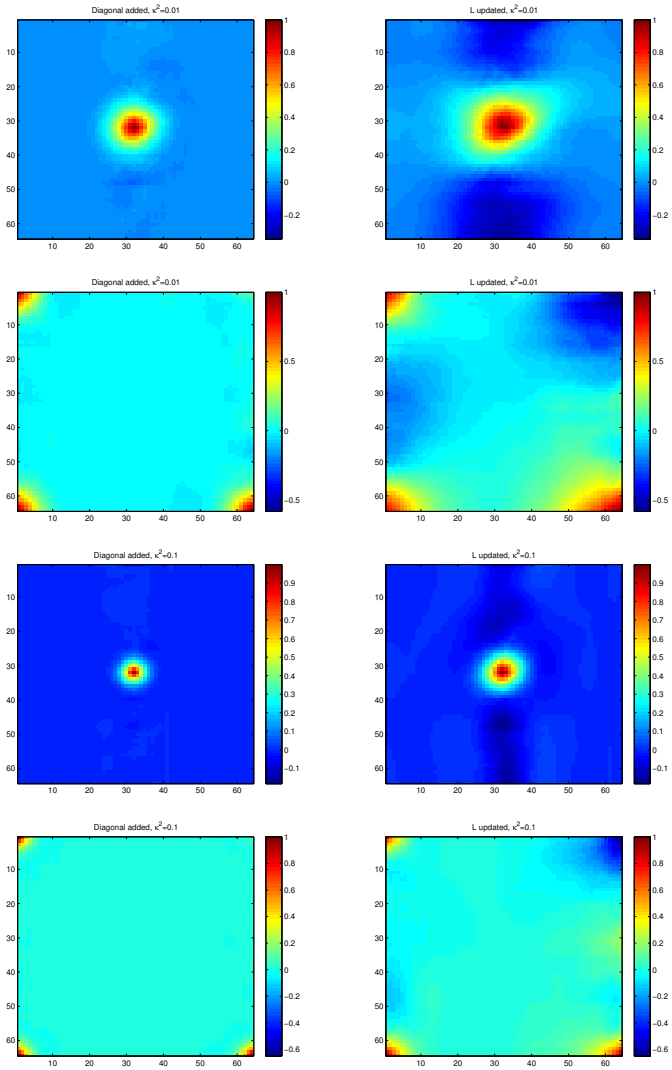
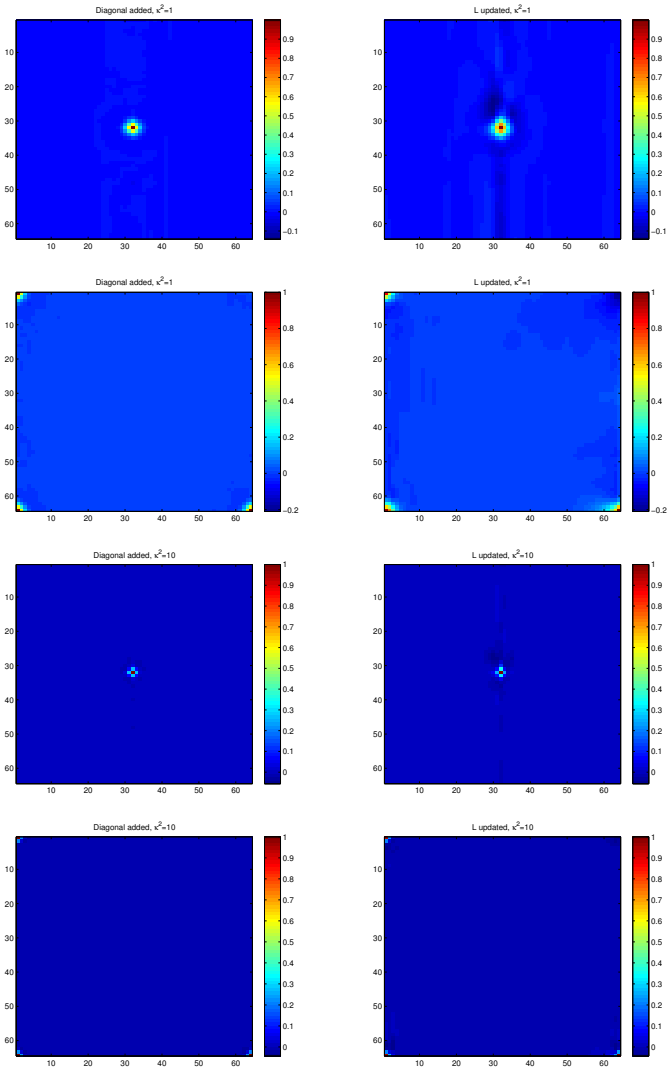


Figure 3.15: *Effect of updating strategy vs. adding a constant diagonal term,* $\kappa^2 = 10^0$ *and* $\kappa^2 = 10^1$



sion matrix construction. The only thing that is left is the determinant $\log\det(\mathbf{L}\mathbf{D}\mathbf{L}^T + (1/\lambda^2)\mathbf{A}^T\mathbf{A})$, for which other techniques need to be considered. If the the observation matrix is the identity, it is easy to use approach with rational approximations to that matrix logarithm with probing vectors, as we have a good preconditioner available, and using just updated diagonals of $\mathbf{L}$ may be good enough. When $\mathbf{A}^T\mathbf{A}$ does not modify the sparsity structure of $\mathbf{L}\mathbf{D}\mathbf{L}^T$, the incomplete Cholesky factorisation with no fill-in should work quite well (as long as its positive definite), given the structure of the problem. If $\mathbf{A}^T\mathbf{A}$ is diagonally dominant, one can use the optimal diagonal preconditioner as shift parameters in the updating scheme for $\mathbf{L}$ to construct a good preconditioner, and thereafter use probing vectors with shifted systems to approximate the log-determinant. We emphasize that no subtractive cancellation will occur, since one log-determinant is exact. Hence, we need not worry about other things than computing the log-determinant.

Another strategy that is potentially even better, is the use of an incomplete Cholesky factorisation in each computational part, based on the initial sparsity pattern obtained by an incomplete factorisation of $\mathbf{W}\mathbf{Q}\mathbf{W}^T + \kappa_0^2\mathbf{I}$. This should give models that resemble the original even better than using the aforementioned preconditioning strategy for obtaining approximate models. Since the sparsity pattern is fixed and having few non-zero entries, computing an updated Cholesky factor is cheap. In order for this to work, we need to store the parts of the matrix $\mathbf{W}\mathbf{Q}\mathbf{W}^T$, corresponding to the sparsity structure of the incomplete factorisation, and the parts corresponding to the sparsity structure of $\mathbf{A}^T\mathbf{A}$, where $\mathbf{A}$ is the observation process. When using this approach, we do not need to consider different techniques for computing the two needed log-determinants, as the Cholesky factorisation with pre-specified sparsity pattern of the shifted model is essentially as easy to compute as the incomplete factorisation for $\mathbf{W}\mathbf{Q}\mathbf{W}^T$.

3.4 Examples

In this section we apply the approximate log-determinant methods to parameter estimation on three examples. The examples are chosen to emphasise both the nice properties and challenges that occur in practical implementations. In the notation here, we assume that $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, (\sigma^2/\lambda^2)\mathbf{Q}_{\kappa^2}^{-1})$, and $\mathbf{y} = \mathbf{A}_{\boldsymbol{\theta}}\mathbf{x} + \sigma\,\mathcal{N}(\mathbf{0}, \mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}}^{-1})$, with essentially $\mathbf{A}_{\boldsymbol{\theta}} = \mathbf{A}$ and $\mathbf{Q}_{\boldsymbol{\epsilon},\boldsymbol{\eta}} = \mathbf{I}$ in subsequent sections. This corresponds to a SPDE model with i.i.d. observations on top of it. When approximating the determinants for parameter estimation in our examples, we solely used the techniques in Section 3.1 with random sign-flipping in the probing vectors.

We compare the estimates using the approach explored in the previous sections with those obtained by a block composite likelihood approach, see Eidsvik et al. (2011). The main idea behind composite likelihoods is to replace the computationally demanding likelihood expression with several block-type expressions. Each term requires less memory and computational time. Thus, rather than working with the full likelihood function $\log p(\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta}, \sigma^2, \lambda^2)$, which in the Gaussian setting is given by (3.3), the block composite likelihood approach adds up Gaussian composite terms built from block interactions.

In essence, partition the domain $\mathcal{D}$ into pairwise disjoint subdomains; $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset, i \neq j$ and $\bigcup_{i=1}^{M} \mathcal{D}_i = \mathcal{D}$. Thereafter, assume that the only interaction terms are between neighbouring blocks. Let $\mathbf{y}_k$ and $\mathbf{y}_l$ be the data in domains $\mathcal{D}_k$ and $\mathcal{D}_l$. Then, the block composite likelihood is available by

$$\begin{aligned} 2\log \quad p_{CL} \quad & (\mathbf{y}|\boldsymbol{\eta}, \boldsymbol{\theta}, \sigma^2, \lambda^2) \\ = \quad & \sum_{k=1}^{M-1} \sum_{l>k} 2\log p(\mathbf{y}_{kl}|\boldsymbol{\eta}, \boldsymbol{\theta}, \sigma^2, \lambda^2) \\ = \quad & \sum_{k=1}^{M-1} \sum_{l>k} \Big(\log\det \mathbf{Q}_{y,kl} \\ & -(\mathbf{y}_{kl} - \boldsymbol{\mu}_{y,kl})^T \mathbf{Q}_{y,kl} (\mathbf{y}_{kl} - \boldsymbol{\mu}_{y,kl}) \Big) \end{aligned} \tag{3.41}$$

where $\boldsymbol{\mu}_{y,kl}$ is the mean of block variables $(\mathbf{y}_k^T, \mathbf{y}_l^T)^T$ and $\mathbf{Q}_{y,kl}^{-1} = \text{Cov}(\mathbf{y}_k, \mathbf{y}_l)$ is the covariance matrix for this block pair.

The maximum composite likelihood estimators are the parameter values $(\boldsymbol{\eta}, \boldsymbol{\theta}, \sigma^2, \lambda^2)$ that optimize expression (3.41). Theoretical considerations and computational approaches for this block composite likelihood model can be found in Eidsvik et al. (2011).

3.4.1 3-D Matérn field with direct and indirect observations

In spatial statistics, it is fairly common to assume that an underlying spatial field comes from the Matérn family or to use a Gaussian prior coming from the same family. The underlying field or prior field is then described by (3.4). We mention that from a physical point of view, the same article makes good arguments for using Neumann boundary instead of imposing artificial boundary conditions corresponding to completely unchanging marginal variances at each site. If we observe $x(\mathbf{s})$ directly, we have direct observations, and we only need to compute one log-determinant for the likelihood evaluation and we avoid the previously discussed effects of subtractive cancellation. If we have an observation process on top of $x(\mathbf{s})$, we are in the setting of (3.3), and we have the problem described in Section 3.1.5. In Figure 3.16 we see a slice of the direct observations and a slice of the corresponding indirect observations, as well as a reconstruction from the indirect observations. In the following examples, we assume that $\alpha = 2$.

In our example, we assume that we gradually observe more sites of the total field, from 15^3 sites to 120^3 sites.

Direct observations

For the rare case where direct (non-noisy) observations are available, the log-likelihood for the Gaussian represents the objective function, presuming no prior information on the parameters to estimate is available. In this

Figure 3.16: *Direct (left) and indirect (right) observations of Matérn field, and a reconstructed field (bottom).*

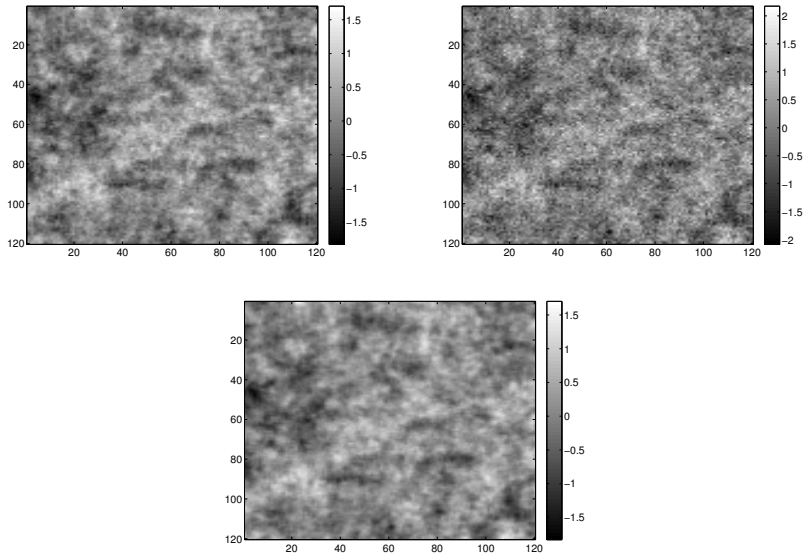


Table 3.3: *Estimation of precision parameters in a Matérn field with respect to distance colouring and observed part of field. This is for the situation with direct observations.*

	4-distance		8-distance		16-distance	
	τ^2	κ^2	τ^2	κ^2	τ^2	κ^2
15^3	0.98362	0.23740	1.00158	0.17137	1.00675	0.15188
30^3	0.97116	0.18467	0.99003	0.11396	0.99783	0.08355
60^3	0.97135	0.18442	0.99147	0.10676	1.00067	0.06988
120^3	0.96716	0.18112	0.98759	0.10131	0.99741	0.06152

setting, two parameters need to be estimated, κ^2, representing the range, and a scaling parameter, τ^2, representing a measure of the amplitude of the realisation. In Table 3.3, we see the effect of using different distance colourings of the precision matrix and observing smaller and larger parts of the field. The true parameters were $\tau^2 = 1, \kappa^2 = 0.05$.

Indirect observations

Suppose that the discretized field $\mathbf{x}$ generated by (3.4) has the observation process $\mathbf{y} = \mathbf{x} + \sigma\boldsymbol{\epsilon}$ attached to it, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we optimise (3.3) for the parameters $\boldsymbol{\eta} = (\kappa^2, \tau^2, \sigma^2)$ in the corresponding precision matrices. In addition to generating a table of the estimated parameters for different distance colourings, we compare them with parameters obtained from the block composite likelihood method. In order to obtain comparable accuracy between for the two log-determinant evaluations in (3.3), we needed to use a larger distance colouring for the perturbed matrix, reflected by n_1/n_2 in Table 3.4. In addition, we use the re-parametrisation $\lambda^2 = \tau^2\,\sigma^2$, since this is beneficent in the optimisation routine. The true parameters are $\sigma^2 = 0.1, \lambda^2 = 0.1$ and $\kappa^2 = 0.05$, and we see that we recover these well by observing more of the field and using more probing vectors.

In order to compare our results with those resulting from the block composite likelihood procedure, some care must be taken: the parameter we

Table 3.4: *Estimation of precision parameters in a Matérn field with respect to distance colouring and observed part of field. This is for the situation with indirect observations. The rightmost column indicates the typical number of iterations needed in the Krylov method to compute one* $\log\det(\mathbf{Q})\mathbf{v}$.

	4/5-distance			8/9-distance			16/17-distance		
	λ^2	κ^2	σ^2	λ^2	κ^2	σ^2	λ^2	κ^2	σ^2
15^3	0.0822	0.2714	0.0966	0.1179	0.1423	0.1040	0.1286	0.10587	0.1055
30^3	0.0667	0.2431	0.0925	0.0941	0.1193	0.1001	0.1042	0.07595	0.1020
60^3	0.0633	0.2559	0.0906	0.0906	0.1189	0.0984	0.0989	0.06943	0.1007
120^3	0.0601	0.2603	0.0894	0.0861	0.1191	0.0972	0.0968	0.06520	0.0994

Table 3.5: *Iteration count for Table 3.4*

Dimensions	Typical iteration count
15^3	67/48
30^3	85/48
60^3	85/48
120^3	85/48

estimate in this model are not completely equivalent to those coming from using covariance functions. In our case, we have the SPDE from the previous section, given by (3.4) for $\alpha = 2$ and the corresponding exponential covariance function in three dimensions,

$$C(r) = \gamma^2 e^{-r/\phi}. \tag{3.42}$$

In particular, the marginal variance parameter, γ^2 for the field is estimated in the composite likelihood approach, while in the SPDE model, using the natural Neumann boundary conditions, the entries of $\mathbf{Q}_{ii}^{-1}$ differ, depending on how far i is from the boundary. Now, $\text{tr}\,(\mathbf{Q}^{-1})/n$ gives a natural estimate for the marginal variance parameter for the overall field and should be comparable to that coming from the composite likelihood approach. Similarly, the range parameter ϕ has its relative in the parameter κ^2, but here there is also no direct correspondence. A natural surrogate in this case is the correlation length, ℓ, which can be computed from the probing vectors. The parameter σ^2 is directly comparable between the two models.

A comparison of estimates achieved by approximating the log-determinant and the composite likelihood estimation is shown in Table 3.6. The estimates are very similar. It appears as if the composite likelihood returns slightly larger range values ℓ and signal to noise γ^2, while the measurement noise level σ^2 is a little smaller. For the log-determinant approach there seems to be a monotone trend in all the parameters when observing more of the field. This is a desirable property that does not seem to hold for the composite likelihood approach. In Table 3.6, there is an artefact for the correlation distance in dimensions 15^3 – this is a consequence of the discretization being so coarse that it is impossible to properly adjudicate it.

Optimisation for the full model using a 16/17-distance colouring took about 50 hours using 24 Intel Xeon cores running at 2.67 GHz. These cores were active with other processes at the same time. Comparing timing between examples is quite hard in this case, since using a computing server usually is busy with many activities at the same time. Additionally, it is possible to tweak performance quite a lot using, for instance, more efficient matrix

Table 3.6: *A comparison of block composite likelihood and the determinant approximation. The γ relates to signal to noise ratio, the ℓ is the correlation range parameter and σ is the measurement noise standard deviation.*

	Comp. lik.			log-det approx.		
	γ^2	ℓ	σ^2	γ^2	ℓ	σ^2
15^3	0.455^2	9.86	0.302^2	0.485^2	27.3	0.325^2
30^3	0.464^2	11.3	0.309^2	0.467^2	10.7	0.319^2
60^3	0.453^2	11.2	0.309^2	0.437^2	10.5	0.317^2
120^3	0.450^2	10.7	0.306^2	0.425^2	10.3	0.315^2

vector products (which is possible with a 3-D Matérn matrix on a grid) and using the extensions discussed in the previous section. The point of these examples is not to compare computational speed, but rather to show that it is possible to obtain maximum likelihood estimates that were previously not possible to do because of the matrix dimensions and properties.

3.4.2 Ozone column estimation

In this example, we analyse total column ozone (TCO) data acquired from an orbiting satelite. This is a popular dataset that has been analysed in Cressie and Johannesson (2008) using a fixed rank Kriging approach and in Eidsvik et al. (2011) using the block composite likelihood method. The dataset has also been modeled using a nested SPDE approach in Bolin and Lindgren (2011). What is special about this dataset is that it is 1) on the sphere and 2) since the data is acquired along the transects of the satellite and therefore a rather special sampling pattern is obtained.

We use the SPDE approach as in the previous sections, only this time on a sphere. We use a "uniform" triangulation on the sphere coming from a triangle fan starting from the (northern) polar vertex for discretizing the SPDE. This gives us a different observation process than in the previous section: the observed data is given by $\mathbf{y} = \mathbf{A}\mathbf{x} + \sigma\boldsymbol{\epsilon}$, where the matrix $\mathbf{A}$ interpolates from the uniform triangulation on the sphere to the observation

pattern given by the satellite. Our discretization consists of 324 002 points on the sphere, and we have 173 405 observations. Since the observations are not snapshots of the globe at a given time, we also get temporal effects in the data. We do not model this temporal effect. An illustration of the observations is given in Figure 3.17.

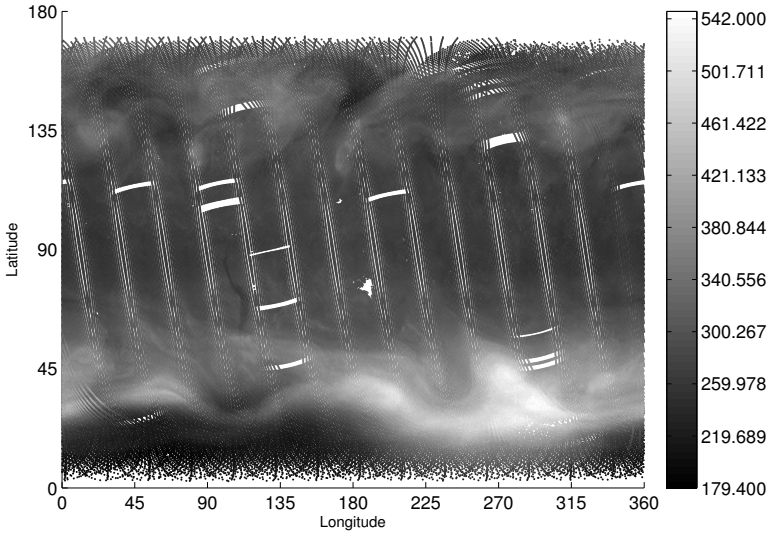
The estimated parameters are $\lambda^2 \approx 0.0117^2$, $\kappa^2 \approx 1.61^2$ and $\sigma^2 \approx 5.015^2$. Here we used a 32-distance colouring for choosing the probing vectors. Using the same tricks as in the last section, this converts to, $\gamma^2 \approx 55.3^2$, $\ell \approx 10\,567$ km. In comparison, the block composite likelihood model, with 15×15 blocks in latitude and longitude, gives the $\gamma^2 \approx 73.6^2$, $\ell \approx 7028$ km, and $\sigma^2 \approx 4.7^2$, which is reasonably similar. We mention that the blocking in the block composite likelihood may not be sufficient to capture all large scale variations. In addition, the covariance function is the exponential, leaving the SPDE model and the block composite likelihood model slightly dissimilar.

In this example, it is actually possible to estimate the parameters using direct factorisation methods for calculating the determinant using a computer with sufficient memory. We did this and the estimates were equal to the ones coming from the approximation technique (γ^2, κ^2 to the first 2 digits, and σ^2 to the first 4 digits). Of course, using a smaller number of probing vectors could potentially influence this negatively, but this issue is of minor importance.

3.5 Discussion

In this chapter, we have presented a new method for performing statistical inference on Gaussian models that are too large for conventional techniques to work. Focussing on the problem of computing likelihoods for large Gaussian Markov random vectors, we have shown that by combining a number of approximation techniques, we can evaluate the likelihood of truly large models in a reasonable amount of time with reasonable accuracy. In particular, we have shown that a combination of function approximation, graph theory,

Figure 3.17: *Total column ozone observations (dots) acquired along satellite orbits. There are* 173 405 *measurements in total. Measurements in Dobson units*



wavelet methods, modern numerical linear algebra, and problem specific tricks are necessary when a problem is so large that Cholesky factorisations are no longer feasible. The explosion of complexity of the proposed computational methods – indicative of the difficulty of the problem – comes at the advantage that *we can actually solve these models*, which is not possible using standard techniques. Furthermore, when combined with the work of Simpson et al. (2008); Simpson (2008); Aune et al. (2012a), this work completes a suite of methods for performing statistical calculations with large Gaussian random vectors using Krylov subspace methods.

3.5.1 Extensions and future work

An article that inspired this work in many ways is the work on using probing vectors for finding the diagonal of the matrix inverse (Tang and Saad (2010)). In this approach, the entire diagonal is wanted - not just its sum - and hence a variant expression is needed, namely

$$\operatorname{diag} f(\mathbf{Q}) = \left(\sum_{j=1}^{n} \mathbf{v}_j^T \odot f(\mathbf{Q})\, \mathbf{v}_j \right) \oslash \left(\sum_{j=1}^{n} \mathbf{v}_j \odot \mathbf{v}_j \right), \tag{3.43}$$

where $\oslash$ and $\odot$, respectively, are element-wise division and multiplication of vectors. Using the same probing vectors as those needed for the determinant, and $f(t) = t^{-1}$ will then yield an estimate for the diagonal of the inverse of the precision matrix, i.e. the marginal variances. Note that it is much easier to compute the diagonal of the inverse using probing vectors, since in this case we are dealing with the matrix inverse. Preconditioning can therefore be applied directly, and since we need to compute $\mathbf{Q}^{-1}\mathbf{v}$ for quite some vectors, traditionally expensive preconditioners can be worth looking into. Specifically, combination of AINV (see. Huckle and Grote (1997)) and wavelet compression may be well suited for extracting this diagonal. In this situation, we get a dual benefit from the wavelet compression: it may both improve upon the AINV preconditioner and decrease the number of probing vectors needed.

The marginal variances together with the log-determinant (required for optimisation) are components needed in dealing with inference by the integrated nested Laplace approximation (INLA) (Rue et al. (2009)), and the approach given in this chapter is a way to extend the INLA approximation to larger models than can not be handled with the current direct methods.

Another potential application of (3.43) is the computation of communication in graphical models, such as social networks and networks of oscillators (Estrada et al. (2011)). Using the matrix exponential or relatives as the map, the diagonal of this is a measure for self-communicability or sub-graph centrality, which is used in analysis of complex networks. Naturally, a matrix-vector type method is needed for the action $\exp(\alpha\,\mathbf{Q})\,\mathbf{v}$, and an innovative approach for this can be found in Al-Mohy and Higham (2011). This approach is well suited for computing the matrix exponential times several probing vectors in parallel.

Using rational approximations for the square-root or inverse square-root with random vectors ($\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) with Krylov subspaces is another venue which has been pursued in Aune et al. (2012a) and Simpson et al. (2008). In these articles, the authors demonstrate that in circumstances where the Cholesky factorisation is impossible to compute due to memory constraints, using rational approximations with Krylov methods is a good substitute, and also show that it is competitive in other circumstances.

In some cases, we may be interested in other entries of $f(\mathbf{Q})$ than its diagonal ones. For $f(t) = t^{-1}$, we obtain correlations between specific nodes and for $f = \exp$ we can obtain an estimate of the communicability between two nodes in an undirected graph. Looking at (3.43), we note that if we change $\mathbf{v}_j^T \odot f(\mathbf{Q})\,\mathbf{v}_j$ to $\mathbf{w}_j \odot f(\mathbf{Q})\,\mathbf{v}_j$, it may be possible to extract other entries of the matrix $f(\mathbf{Q})$. The question that remains is how to choose $\{\mathbf{w}_i\}_{j=1}^{kn}$ corresponding to the set $\{\mathbf{v}_j\}_{j=1}^{n}$. A heuristic that may help in forming the set of $\mathbf{w}_i$s is that if $\mathbf{v}_j$ is given, the corresponding $\mathbf{w}_i$s should be those corresponding to the neighbours of the non-zero entries of $\mathbf{v}_j$. We do not pursue this idea in here, but it is an interesting topic for future research.

3.5.2 Software

The software package KRYLSTAT by Aune, E. contains an implementation of the log-determinant approximation outlined in Section 3.1 with random flipping in probing vectors. For ease of use, MATLAB (MATLAB (2010)) wrappers for the relevant functions are included. It also contains implementations of one of the sampling procedures found in Aune et al. (2012a) and a refined version of the marginal variance computations found in Tang and Saad (2010). The package can be found on `http://www.math.ntnu.no/~erlenda/KRYLSTAT/`.

Chapter 4

Non-linear and non-stationary AVA inversion in 3-D

The inversion of seismic amplitude versus angle (AVA) data is relevant for petroleum exploration and production. One goal of the inversion is to extract the elastic parameters of the subsurface from the processed seismic AVA data, while incorporating a priori understanding of the geological conditions. Another important goal is to obtain the uncertainties of the inversion results, which allow decision making under uncertainty.

There are three modeling elements that have to be specified for doing seismic AVA inversion: i) We need an AVA reflectivity model consisting of physical relations for the receiver responses, given the subsurface properties, see Aki and Richards (1980). ii) The statistical error model for these seismic AVA responses must be defined. Together, i) and ii) comprise what is called the likelihood model for the seismic AVA data. iii) A prior model for the elastic parameters must be specified, enforcing solutions that are geologically representative. This prior also corresponds to a regularisation term in the numerical literature. The prior and likelihood model define a Bayesian framework, where we have an associated, consistent, posterior distribution for the elastic parameters, given the seismic AVA data, see e.g. Ulrych et al. (2001) and Malinverno and Briggs (2004). Of course, it is critical to use

realistic modeling assumptions in i)-iii) in order to get reliable results in the seismic AVA inversion.

The key issue in step i) is the geophysical model for the seismic AVA reflectivity. We will focus only on the PP reflectivity, and we will compare the linear approximation of Buland and Omre (2003) and the quadratic approximation developed in Stovas and Ursin (2003). It is well known that the linearized version does not work so well for large angles, or for large contrasts in the elastic parameters at interfaces. One contribution in this chapter is to study the impact of linearized reflection models on the inversion results. Another important modeling aspect in step ii) and iii) is the structure imposed by the prior model and the error terms in the likelihood. In this chapter we will use Gaussian distributions for the prior on elastic parameters and for the error terms in the likelihood model. We will allow non-stationarity in these Gaussian processes, i.e. use varying mean or covariances for different locations, and study how stationarity assumptions (Buland et al., 2003) influence the inversion results.

When we use a linear reflectivity model, we get a Gaussian posterior distribution for the elastic parameters. For the quadratic reflectivity model, the posterior is not available in closed form. The objective function may then contain multiple optima. In a similar way, we would get intractable posterior distributions if we imposed a non-Gaussian a priori model, or non-Gaussian error terms in the likelihood. This has for instance been used to obtain blocky inversion results, see e.g. Farquharson and Oldenburg (1998), Youzwishen and Sacchi (2006) and Theune et al. (2010). We show that flexible models can be constructed by staying within the Gaussian class, but incorporating non-stationarity.

Seismic data come in very high dimensions, often with observation counts of $\mathcal{O}(10^5) - \mathcal{O}(10^9)$, depending on how they are processed and acquired. The effective dimension is not that large, because there is much smoothing of the data via the wavelet convolution, but nevertheless there are big computational challenges in handling the dimension. The statistical model and the numerical techniques, often intertwined, are constrained both in

computational cost and in memory. In order to construct relevant solutions in 3D, we therefore focus on sparsity both of the statistical model and the numerical techniques. The Gaussian processes are parametrized by sparse precision matrices (inverse covariance matrices), which lowers the memory requirements.

Our focus in this chapter is to identify the regions of noise-space where it may be advantageous to use the quadratic reflectivity model, and we study the impact of incorporating a specific type of non-stationarity in the model. The main contribution can be seen as an extension of Rabben et al. (2008), who showed the effect of non-linearity in a 2D model at an interface. We extend this approach to 3D applications, and allow non-stationarity in the Gaussian models. One can also regard the methodology in this chapter as an extension of the 3D linearised inversion of Buland et al. (2003), where we now allow non-linearity and/or non-stationarity. One of the main challenges in this is the numerics required to obtain proper inversion. By extension, this is also a natural focus of this text.

In Section 2 we discuss the geophysical model for the seismic AVA reflectivity. Section 3 specifies our statistical modeling assumptions. The computational aspects, which become so important in massive 3D seismic datasets, are presented in Section 4. We analyze and interpret numerical experiments in Section 5. In particular, we investigate both the ℓ_2-error and ℓ_∞-error in the inversion result: The ℓ_2-error captures the mean performance over the entire field, but is unsuitable for assessing performance locally in a high-contrast layer. The ℓ_∞-error is more appropriate for investigating possible performance gains when using a quadratic forward model, as it is in the high-contrast layer(s) we expect to gain the most. However, if this gain comes at a large expense in the ℓ_2 world, it may not be completely desirable.

4.1 AVA model

General formulae for the seismic AVA reflections at an interface date back to Zoeppritz, and this topic is thoroughly presented in Aki and Richards (1980). The relationships hold for different subsurface media, and for P- and S-wave incidence or reflections. Several approximate representations of these formulae have been presented in the literature. The seismic AVA model we adopt in this chapter is primarily based on the one given in Rabben et al. (2008). This model uses the quadratic approximation derived in Stovas and Ursin (2003) for the reflection coefficients in layered transversely isotropic viscoelastic media. Here, the seismic PP reflection coefficients, at an interface, and for incidence angle θ, are given by

$$\begin{aligned} r_{PP} =& \frac{1}{2\cos^2(\theta)} m_1 - 4\sin^2(\theta)\gamma^2 m_2 - \frac{1}{2}\tan^2(\theta)(1 - 4\gamma^2\cos^2(\theta)) m_3 + \\ & \frac{\tan(\theta)\gamma}{1-\gamma^2\sin^2(\theta)} \Big\{ 4\gamma^2(1-\sin^2(\theta)(1+\gamma^2)) m_2^2 - \\ & 4\gamma^2(1-\sin^2(\theta)(3/2+\gamma^2)) m_2 m_3 + (\gamma^2(1-(2+\gamma^2)\sin^2(\theta) - 1/4) m_3^2 \Big\}, \end{aligned} \tag{4.1}$$

where the relative difference of elastic parameters are,

$$m_1 = \frac{\triangle I_P}{\overline{I}_P}, \quad m_2 = \frac{\triangle I_S}{\overline{I}_S}, \quad m_3 = \frac{\triangle \rho}{\overline{\rho}}. \tag{4.2}$$

Here the $\triangle$ denotes differences in time. Hence, m_1, m_2, m_3 denotes the relative difference of P-impedance, S-impedance and density respectively. Moreover, γ denotes the background v_s/v_p-ratio, which is allowed to depend on depth. Of course, the elastic parameters m_1, m_2 and m_3 depend on the position in the subsurface x (inline), y (crossline) and t (traveltime). It is possible to parametrise r_{PP} in terms of other quantities as well, such as P-velocity, S-velocity and density, as is done in Buland and Omre (2003), but it is shown in Tarantola (1986) that it is more efficient to work with impedances.

In the common situation with processed 3D seismic AVA data, the reflections r_{PP} at interfaces are represented by a convolution model, see Buland and Omre (2003). Mathematically, we represent the observations by,

$$d(x,y,t,\theta) = \int_{\mathbb{R}} w(\tau,\theta)(r_{PP}(x,y,t-\tau,\theta) + \epsilon_1(x,y,t-\tau,\theta))d\tau + \epsilon_2(x,y,t), \tag{4.3}$$

where w is a wavelet working in the vertical direction, and ϵ_1, ϵ_2 are wavelet convolved and independent error terms, respectively. We will work with the model (4.3) under different assumptions on ϵ_1, ϵ_2 and r_{PP}. Explicitly;

- if we drop line 2 and 3 of (4.1), we are left with the corresponding linear model, and we compare this simplified model with the quadratic expression.
- We investigate effects of the stationarity assumption on the error processes ϵ_1, ϵ_2.

In practice we collect the entire dataset, for all angles and at all crossline, inline and depth coordinates, in a long vector $\mathbf{d}$. The same holds for the elastic parameters $\mathbf{m}$. In matrix-vector notation, our discretized model then reads

$$\mathbf{d} = \mathbf{W}(q(\mathbf{m}) + \boldsymbol{\epsilon}_1) + \boldsymbol{\epsilon}_2, \tag{4.4}$$

where q denotes the functional operation defined by (4.1), collected into a long vector over all interfaces for various traveltimes and for all seismic inline-crossline traces. Moreover, $\mathbf{W}$ denotes the discretized version of w in (4.3), $\mathbf{m}$ is discretized elastic parameters and $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2$ are vectors of noise components. If we use the linear reflection model in (4.1) – i.e. the first line of the equation – we set $q(\mathbf{m}) \sim \mathbf{Am}$, signifying the linearity by using a fixed matrix $\mathbf{A}$ that does not depend on $\mathbf{m}$.

It is known that for large angles, and large relative differences in the elastic parameters, the linear model is insufficient for producing accurate response data (Aki and Richards, 1980). The quadratic approximation

used here alleviates this to some extent (Stovas and Ursin, 2003). It is not, however, obvious that the quadratic model robustly reconstructs the elastic parameters in a seismic inversion, and we will address this topic.

In addition to this, we assume that we observe the elastic properties $\mathbf{m}$ directly from one or more well logs. This can be written by $\mathbf{d}_3 = \mathbf{G}\mathbf{m} + \boldsymbol{\epsilon}_3$, where $\mathbf{G}$ simply picks out some coefficients of $\mathbf{m}$, i.e. $G(i,j) = 1$ when well observation i is a measurement of the elastic parameter indexed j. We will assume that $\boldsymbol{\epsilon}_3$ is small, i.e. the well log information, at the location where it is collected, has much higher fidelity than the seismic data.

4.2 Statistical model

Now, we assign distributions to the stochastic elements of the model, and study the impact on the posterior model under different reflection models. Explicitly, we will assume Gaussian distributions, i.e.

$$\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{Q}_1^{-1}), \quad \boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{Q}_2^{-1}), \quad \boldsymbol{\epsilon}_3 \sim \mathcal{N}(\mathbf{0}, \sigma_3^2\mathbf{Q}_3^{-1}), \quad \mathbf{m} \sim \mathcal{N}\left(\boldsymbol{\mu}_m, \lambda^2\mathbf{Q}_m^{-1}\right). \tag{4.5}$$

Here the overall noise level σ^2 depends on σ_1^2 and σ_2^2, but it is not needed in any computational parts in this text, and is therefore generally suppressed in our notation.

The precision matrices, i.e. the inverse covariance matrices, denoted $\mathbf{Q}$, will depend on some statistical model parameters, but this is suppressed in the notation. Notably, the precision or covariances matrices include parameters that specify the measurement noise level – possibly depth dependent – and the a priori uncertainty in the elastic properties. In addition, the dependencies between the various elastic properties (P-, S-impedance and density) are represented in a parametric form. Finally, there is spatial dependence involved, stating that the parameters at one spatial location are closely related to the neighbouring parameters.

The modeling assumptions entail that the likelihood model, defining the conditional distribution of the observations given the parameters, is $\mathbf{d}$ –

$\mathbf{W}q(\mathbf{m}) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{lik}^{-1})$. Using (4.4), and assuming independent $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$, we get a likelihood precision matrix given by

$$\mathbf{Q}_{lik} = (\sigma_1^2 \mathbf{W}\mathbf{Q}_1^{-1}\mathbf{W}^T + \sigma_2^2 \mathbf{Q}_2^{-1})^{-1}. \tag{4.6}$$

The resulting likelihood is denoted by $p(\mathbf{d}|\mathbf{m})$, while the prior model, $p(\mathbf{m})$, is Gaussian defined in (4.5). The posterior model is then given by Bayes rule:

$$p(\mathbf{m}|\mathbf{d}) = \frac{p(\mathbf{d}|\mathbf{m})p(\mathbf{m})}{p(\mathbf{d})} \propto p(\mathbf{d}|\mathbf{m})p(\mathbf{m}). \tag{4.7}$$

This posterior is the key output from a Bayesian inversion of seismic AVA data. If a linear approximation $\mathbf{A}\mathbf{m}$ is used in place of $q(\mathbf{m})$. In this case, the posterior is Gaussian, and the mean or mode value is available in closed form. It might be computationally demanding to solve for the posterior mean, and not to mention the posterior variance, but in theory its form is known. If the quadratic model $q(\mathbf{m})$ is used, the posterior is not available in closed form. One can still approximate the mode of the posterior by iterative methods. It is also possible to fit a Gaussian approximation by matching the curvature (second moment) at the mode. We discuss both approaches below. We will incorporate information obtained from the well log in the prior distribution for $\mathbf{m}$ and use this for both the linear and non-linear case. The derivation in the Appendix 4.6 also gives guidance on how to incorporate this well log information into the prior. In this respect, the well log part is integrated through $p(\mathbf{m}) = p(\mathbf{m}|\mathbf{d}_3)$. Before studying the posterior distributions, we will specify the full modeling assumptions for the covariances.

4.2.1 Prior and likelihood modeling

The standard technique for prior and likelihood modeling in 3-D is the one given in Buland et al. (2003). This technique puts some limitations on both the likelihood and prior structure, but provides an extremely fast and stable inversion algorithm due to its reliance on the fast Fourier transform

(FFT). More explicitly, the assumption made is that $\mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}\mathbf{A} + \mathbf{Q}_m$ is (block-)circulant. To facilitate this, the matrix $\mathbf{A}$ needs to have a fixed v_S/v_P-ratio and $\mathbf{Q}_m$ and $\mathbf{Q}_{lik}$ need to be circulant and hence cannot depend on the position in the field.

Until this point we have treated the elastic parameters and data quite generically, without going into details about dimensions. When defining the covariances or precisions properly, we require some more notation. Let the 3-D grid of seismic data be of size n_x, n_y and n_t, which is most commonly related to the number of inlines, crosslines and the indexed traveltimes. The resolution of the grid may differ, and depends both on acquisition and processing. At each grid cell there are three elastic parameters (P-, S-impedance and density), and n_θ observations for various angles. Our model specifications will rely on cell-wise components, trace-wise components, and spatial dependence components.

For the likelihood terms, we assume that $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are diagonal matrices. We set $\mathbf{Q}_2 = \sigma_2^{-2}\mathbf{I}$ indicating independent measurement noise at the receiver end, and assume that the variance σ_2^2 is quite low compared with the wavelet convolved noise term. For $\mathbf{Q}_1$, representing the precision of the wavelet convolved noise, factors in the geophysical properties of the subsurface are propagated, and we therefore assume that this noise level increases with depth. The assumption leads to non-stationarities in the likelihood (and the posterior). The precision matrix $\mathbf{Q}_1$ is built to decay with depth:

$$\mathbf{Q}_1 = \mathbf{P}\left(\mathbf{Q}_\theta \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \mathbf{Q}_w)\right)\mathbf{P}^T, \tag{4.8}$$

where $\mathbf{Q}_\theta$ represents the (inverse of) the correlation between the reflection angles (assumed diagonal here), $\mathbf{Q}_w$ induces the increasing variance with depth and $\mathbf{P}$ is a permutation matrix giving the wanted ordering of the inline crossline and traveltime cells. The Kronecker product is represented by the symbol $\otimes$. We stress that it is not necessary to use the permutation explicitly, but rather access indices as needed during computation. In general, it is difficult to construct $\mathbf{Q}_{lik}$ explicitly, and the most convenient representation of (4.6) depends on the structure of $\mathbf{Q}_1$ and $\mathbf{Q}_2$. Setting

$\mathbf{Q}_\theta = \mathbf{I}$, it is possible to write a computationally efficient expression for $\mathbf{Q}_{lik}$. First note that $\mathbf{W} = \mathbf{I}_{n_\theta \cdot n_x \cdot n_y} \otimes \mathbf{W}_0$, where $\mathbf{W}_0$ is contains wavelets that are replicated for each trace. Moreover,

$$\mathbf{Q}_1^{-1} = (\mathbf{Q}_\theta \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \mathbf{Q}_w))^{-1} = \mathbf{Q}_\theta^{-1} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \mathbf{Q}_w^{-1}) \tag{4.9}$$

Now, with $\mathbf{Q}_\theta = \mathbf{I}$,

$$\begin{aligned}
\sigma_1^2 \mathbf{W}\mathbf{Q}_1^{-1}\mathbf{W}^T + \sigma_2^2\mathbf{I} &= \sigma_1^2(\mathbf{I}_{n_\theta \cdot n_x \cdot n_y} \otimes \mathbf{W}_0)\mathbf{Q}_\theta^{-1} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \mathbf{Q}_w^{-1})\mathbf{W}^T + \sigma_2^2\mathbf{I} \\
&= \sigma_1^2(\mathbf{Q}_\theta^{-1} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \mathbf{W}_0\mathbf{Q}_w^{-1}))\mathbf{W}^T + \sigma_2^2\mathbf{I} \\
&= (\mathbf{Q}_\theta^{-1} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes \sigma_1^2\mathbf{W}_0\mathbf{Q}_w^{-1}\mathbf{W}_0^T)) + \sigma_2^2\mathbf{I} \\
&= \mathbf{I}_{n_\theta} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes (\sigma_1^2\mathbf{W}_0\mathbf{Q}_w^{-1}\mathbf{W}_0^T + \sigma_2^2\mathbf{I}_{nt})).
\end{aligned} \tag{4.10}$$

This gives

$$\mathbf{Q}_{lik} = (\sigma_1^2\mathbf{W}\mathbf{Q}_1^{-1}\mathbf{W}^T + \sigma_2^2\mathbf{I})^{-1} = \mathbf{I}_{n_\theta} \otimes (\mathbf{I}_{n_x \cdot n_y} \otimes (\sigma_1^2\mathbf{W}_0\mathbf{Q}_w^{-1}\mathbf{W}_0^T + \sigma_2^2\mathbf{I}_{nt})^{-1}). \tag{4.11}$$

Since $n_t < 1000$, it is easy to compute this inverse, which will be blocked and very sparse.

For specifying the prior precision $\mathbf{Q}_m$, we use a variant of the framework developed in Lindgren et al. (2011). More explicitly, we will assume that the prior is defined by the following separable system of stochastic partial differential equations:

$$(\mathbf{Q}_0 \otimes \mathbf{M})\mathbf{m}(s) = \tau(s) \odot \mathcal{W}(s), \tag{4.12}$$

where $\mathbf{M} = (\kappa - \nabla \cdot \mathbf{H}\nabla)^{\alpha/2}$, wherein κ denotes the range parameter. Furthermore, $\mathbf{H}$ is positive definite matrix, determining the anisotropy of the prior field, $\tau(s)$ is a scaling term, and $\odot$ denotes element-wise multiplication.

For the case where $\mathbf{H}$ is diagonal and κ does not vary spatially, the model reduces to $(\kappa - \triangle)^{\alpha/2} m_i = \mathcal{W}$. Lindgren et al. (2011) show through a

spectral argument how this model is equivalent to the Matérn covariance function (Matérn (1960)) for stationary spatial Gaussian fields

$$r(\mathbf{s},\mathbf{t}) = \frac{\varsigma^2}{\Gamma(\nu)2^{\nu-1}}(\kappa\|\mathbf{s}-\mathbf{t}\|_2)^{\nu}K_{\nu}(\kappa\|\mathbf{s}-\mathbf{t}\|_2). \tag{4.13}$$

Here ς^2 is defines the marginal variance of the field, ν is a smoothness parameter and κ is a range parameter. K_ν is a modified Bessel function of the second kind. The parameters in this model relates to the stationary SPDE model through

$$\alpha = \nu + d/2, \tag{4.14}$$

where d is the dimension of the field. Hence, α and ν govern the differentiability of the field.

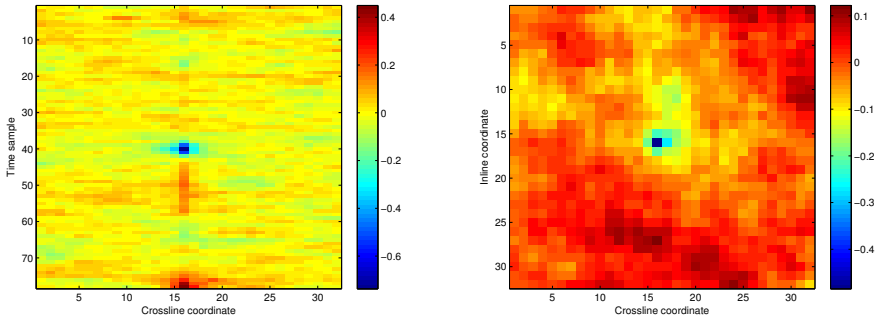
We will use Neumann boundary conditions for the differential equation, i.e. that normal derivatives at the boundary are zero. This is a reasonable assumption for our model: when getting closer to the boundary, less information from the surrounding field is available for lending strength to the value of the parameter at that point in space.

Discretizing equation (4.12) gives a zero-mean realization $\mathbf{m}$ which has sparse precision matrix $\mathbf{Q}_m$, and we add the mean μ_m to get the correct expected value in the prior. The resulting structure of $\mathbf{Q}_m$ defines a Markov random field for the Gaussian process, where non-zero elements in $\mathbf{Q}_m$ indicate neighbours on the defined neighbourhood for the 3-D lattice. A realisation from the prior is given in Figure 4.1, where we see that the information from the well incorporated clearly.

4.2.2 Mode of the posterior

The main goal of a Bayesian inversion is the mode of the posterior distribution in (4.7). When we use a linear approximation, both prior and likelihood

Figure 4.1: *Realisation from prior distribution. Time view (left) and lateral coordinates (right)*



exponents are quadratic forms in $\mathbf{m}$. Thus, the posterior distribution can be written explicitly:

$$p(\mathbf{m}|\mathbf{d}) = (2\pi)^{n/2} \det(\mathbf{Q}_{m|d})^{1/2} e^{-\frac{1}{2}(\mathbf{m}-\boldsymbol{\mu}_{m|d})^T \mathbf{Q}_{m|d}(\mathbf{m}-\boldsymbol{\mu}_{m|d})} \tag{4.15}$$

with posterior precision matrix

$$\mathbf{Q}_{m|d} = \lambda^2 \mathbf{Q}_m + \mathbf{A}^T \mathbf{W}^T \mathbf{Q}_{lik} \mathbf{W} \mathbf{A} \tag{4.16}$$

assuming that the prior for $\mathbf{m}$ is distributed as $\mathcal{N}\left(\mu_m, \lambda^{-2}\mathbf{Q}_m^{-1}\right)$, and $\mathbf{Q}_m$ is defined as in the previous section. Moreover, the posterior mean is

$$\boldsymbol{\mu}_{m|d} = \mathbf{Q}_{m|d}^{-1} \mathbf{d}_{\text{mod}}, \tag{4.17}$$

where $\mathbf{d}_{\text{mod}} = \left(\lambda^2 \mathbf{Q}_m \boldsymbol{\mu}_m + \mathbf{A}^T \mathbf{W}^T \mathbf{Q}_{lik} \mathbf{d}\right)$. Even though this expression is in closed form, the posterior mean in (4.17) requires the solution to a very high dimensional linear system of equations. Since the precision matrices are quite sparse, we benefit from using Krylov subspace methods here, such as conjugate gradients (CG), see below.

For the linear case the mean equals the mode because of symmetry. For the non-linear case, neither mean nor mode of the distribution can be

determined explicitly. The posterior mode is defined by

$$\hat{\mathbf{m}} = \operatorname{argmin}\left[(\mathbf{d} - \mathbf{W}(q(\mathbf{m}))^T \mathbf{Q}_{lik}(\mathbf{d} - \mathbf{W}(q(\mathbf{m})) + \lambda^2(\mathbf{m} - \mu_m)^T \mathbf{Q}_m(\mathbf{m} - \mu_m)\right]. \tag{4.18}$$

We solve for the posterior mode using iterative techniques. At each stage we solve a high dimensional linear system similar to that in (4.17), where the matrix $\mathbf{A}$ and $\mathbf{d}$ are modified according to a linearisation at the current value of the iterative scheme. Computational details are provided below.

4.2.3 Marginal variances of the posterior

In order to assess the variability of the elastic parameters, it is natural to look at the marginal variances at each site. Unfortunately, since we are in the precision domain, these are not explicitly available - contrary to the situation where modeling is through covariance functions. We therefore need a way to extract these from the posterior precision matrix. In the linear case, this is merely $\mathbf{Q}_{m|d}$ as in (4.16). In the non-linear case, the matrix $\mathbf{A}$ is replaced by $Dq(\mathbf{m})\big|_{\hat{\mathbf{m}}}$ - i.e. the derivative of q at the mode, and we then have a proper surrogate for the posterior precision.

In order to compute the marginal variances, we adopt the strategy described in Tang and Saad (2010) with the modification of Aune et al. (2012c). Let $\oslash$ be element-wise division. Then, the diagonal elements of the precision matrix are approximated by

$$\operatorname{diag} \mathbf{Q}^{-1} \approx \left(\sum_{j=1}^{N} \mathbf{v}_j \odot \mathbf{u}_j\right) \oslash \left(\sum_{j=1}^{N} \mathbf{v}_j \odot \mathbf{v}_j\right), \tag{4.19}$$

where $\mathbf{v}_j$ are so-called probing vectors and $\mathbf{u}_j = \mathbf{Q}_{m|d}^{-1}\mathbf{v}_j$. These $\mathbf{v}_j$s are chosen by doing a k-distance colouring of the graph induced by $\mathbf{Q}_{m|d}$, and setting, for colour i, $\mathbf{v}_j^i = f \iff \mathbf{Q}_{m|d}^k(i,j) \neq 0$, where $f = \pm 1$ with probability $1/2$ for each value. The number of probing vectors, N, depends heavily on the basis choice and spatial correlation length induced by $\mathbf{Q}_{m|d}$

but it is usually moderate, say $N = 100$, which is much smaller than the number of elements in $\mathbf{v}_j$. The equation $\mathbf{Q}_{m|d}\mathbf{u}_j = \mathbf{v}_j$ must again be solved by a Krylov method in order to keep the memory requirements low enough. For more details, see Aune et al. (2012c).

4.2.4 Wavelet parametrization and estimation

So far we have treated all statistical model parameters as fixed. This also holds for the wavelet operator. In practice one must estimate the wavelet from data. Commonly well logs are combined with seismic AVA data at the well location to perform this estimation task.

We introduce an alternative wavelet parametrisation in order to reduce the parameter space when doing optimisation. The class is parsimonious, yet quite flexible. First, assume that ψ is a symmetric, smooth wavelet; i.e. that $\lim_{t\to\infty}(1+|t|^{\alpha})\psi(t) = 0$ for $\alpha > 0$ and that $\int_{\mathbb{R}} \psi(t)dt = 0$. Now, let g be smooth, bounded with $\sup_t h(t) = B$ and antisymmetric. Set $h = \frac{g}{2b} + 1/2$, and let $w = \psi \cdot h$, then w is a skewed wavelet.

An example of a parametrisation of this kind is given as follows: Let ψ_n be the nth Hermite function and let $g = \text{erf}$, then we get the following parametrisation

$$w(t|s,a,p,v,n) = (-1)^n a\, \psi_n(t/v - p)\{\text{erf}[s(t/v - p)] + 1\}/2 \tag{4.20}$$

where s governs the skewness, a governs scale, v governs dilation and p position of the wavelet.

Suppose we have both seismic AVA and well data, and we assume the relative noise level, σ_3^2, in the well log is minuscule. Then, we define the objective function by the residual

$$\Phi_\theta(s,a,p,v,n) = \sum_{i=1}^{N} |\mathbf{w}_\theta \star q_\theta(\mathbf{m}_{\text{well}_i}) - \mathbf{d}^\theta_{\text{seism}}| \tag{4.21}$$

where $\mathbf{d}^{\theta}_{seism}$ is the seismic AVA data along the well at angle θ, q_θ is the forward reflection coefficient model for angle θ, $\mathbf{w}_\theta$ is the discretized wavelet for angle θ and $\star$ denotes discrete convolution in time. The wavelet, $\mathbf{w}_\theta$, depends implicitly on the parameters in (4.20).

The objective function, Φ_θ, is non-linear with multiple local maxima, so we cannot expect that starting Newton iterations from an arbitrary starting position will yield a global maximum. To remedy this, we make a sobol sequence (Bratley and Fox, 1988) to have a low-discrepancy sequence of starting positions. The discrepancy of a set $P = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{R}^5$ is given by

$$\sup_{B \in \mathcal{J}} \left| \frac{A(B;P)}{n} - \mu(B) \right|, \tag{4.22}$$

where μ is 5-dimensional Lebesgue measure, $A(B;P)$ is the number of points in P that falls into B, and $\mathcal{J}$ is the set of all 5-dimensional boxes. Hence, a sequence has low discrepancy if the number of points falling in an arbitrary box is close to proportional to to the measure of that box.

Having a low discrepancy sequence should leave at least one point in each region of convexity of the parameter space. After that, we perform (Quasi) Newton iterations on each starting position and at the end, choose the one which minimises Φ_θ. In our synthetic case below, the wavelets are recovered with high precision using this automatised fitting procedure.

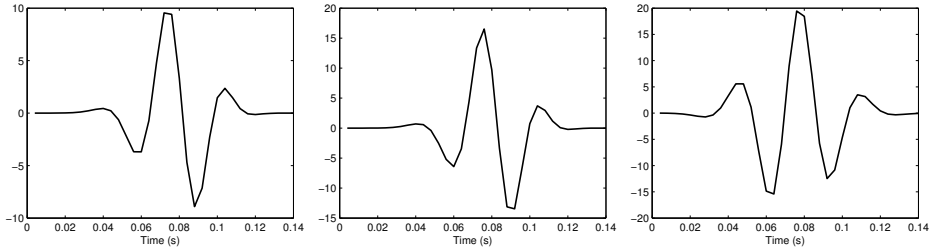
We mention that in general, it may be advantageous to leave one well out for prediction purposes and cross-validation to see the performance of the prediction using the other wells for estimating the wavelet.

4.3 Computational methods and challenges

When we use the linear likelihood model, the inversion for the elastic parameters entails a solving the linear system

$$\mathbf{Q}_{m|d}\mathbf{m} = \mathbf{d}_{\text{mod}} \tag{4.23}$$

Figure 4.2: *Wavelet estimates – near (left), middle (middle) and far (right)*



with $\mathbf{d}_{\text{mod}} = Dq(\mathbf{m})^T\mathbf{W}^T\mathbf{d} + \lambda^2\mu_m$. It is also a linear least squares problem – maybe an ill-conditioned one – easily treated by the methods in Björck (1996).

In order to invert for the elastic parameters, the hyperparameters in the prior model, κ^2 and σ_3^2 are set, and in the likelihood σ_1^2 and σ_2^2 are swept over. Estimation of these parameters may be done in any way that is convenient for latent Gaussian model – for the quadratic model, no modification is needed. Typically, they can be estimated by maximum likelihood procedures, or, more generally, by sampling. For Gaussian models, these matters are treated in e.g. Rasmussen and Wiliams (2006); Rue and Held (2005); Cressie and Wikle (2011). Typically, procedures reminiscent of the EM algorithm (Dempster et al., 1977) can be used for estimation. In our treatment, these estimates are not the matters of interest – rather, we would like to see how different regimes of the parameters may affect a possible superiority of using the quadratic model over the linear one. Therefore, the parameters σ_1^2, σ_2^2 are swept over, while the range parameter, κ^2, is estimated from the well, and σ_3^2 is simply kept at a very low level. The parameters σ_1^2, σ_2^2 jointly determine the total noise level of the model.

For estimating the wavelet, we use the approach described in Section 4.2.4, with Hermite functions as building blocks. In Figure 4.2, we see what these estimates look like in a specific case.

Before going into details, it is convenient to specify the variable ordering of

our discretization scheme; it is naturally easy to go from one ordering to another through specific permutation matrices. However, to fix our notation for Kronecker-type operations, we specify it here. In discretizing, we sample regularly in the x, y, t, leaving us with $x_i, i = 1, \ldots, n_x$, $y_j, j = 1, \ldots, n_y$, $t_k, k = 1, \ldots, n_t$, where n_x, n_y, n_t are $\mathcal{O}(10^2)$. Discretising the angles, θ_i, comes naturally from the different incident angles of the data collected. Typically n_θ is less than 10. We adopt the following ordering:

$$\mathbf{m} = \big(m_1(x_1, y_1, t_1), m_2(x_1, y_1, t_1), m_3(x_1, y_1, t_1), m_1(x_1, y_1, t_2), \cdots, \\ m_1(x_1, y_2, t_1), \cdots, m_3(x_{n_x}, y_{n_y}, t_{n_t})\big)^T,$$

and similarly,

$$\mathbf{d} = \big(d_{\theta_1}(x_1, y_1, t_1), \cdots, d_{\theta_{n_\theta}}(x_1, y_1, t_1), d_{\theta_1}(x_1, y_1, t_2), \cdots, d_{\theta_{n_\theta}}(x_{n_x}, y_{n_y}, t_{n_t})\big)^T.$$

4.3.1 Linear inversion

Central to the outlined seismic AVA inversion are efficient iterative solvers for linear systems. That is, how to solve $\mathbf{Q}_{m|d}\mathbf{m} = \mathbf{d}_{\text{mod}}$. The method of choice is highly dependent on the structure of the posterior precision. In our context this matrix is sparse and with varying structure properties.

It is possible to define a stationary approximation to the model discussed in the previous sections. If we let the prior precision be stationary, which in our setting means that we do not incorporate the precision modification at the well location, let the v_S/v_P-ratio be constant for the entire field, let the wavelet be symmetric, and do not let the error model depend on depth, then we have a completely stationary model. In this setting, if we wrap our matrices at the edges, leading to a periodic boundary, we obtain a circulant approximation to $\mathbf{Q}_{m|d}$ in (4.17). For such circulant matrices, all computations can be done using the FFT (Gray, 2006). In this particular case, the inversion is of $\mathcal{O}(n \log n)$, and this is the main advantage for using a stationary model.

In the non-stationary linear case, the natural domain is iterative solvers for the corresponding linear system. These typically use the matrix vector product $\mathbf{Q}_{m|d}\mathbf{v}$ or some modification of that multiple times for different $\mathbf{v}$s in each iteration, and there are several options to choose from, going from the usual Gauss-Seidel iterations, through multigrid methods to Krylov methods. An overview of all these can be found in Saad (2003), with further references. We focus on Krylov methods – a suite of methods that are very general and require only the matrix vector product $\mathbf{Q}_{m|d}\mathbf{v}$ in each iteration for pursuing the solution.

The essence of Krylov methods is the following: build a Krylov subspace, defined by

$$\mathcal{K}_m(\mathbf{Q}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{Q}\mathbf{r}_0, \mathbf{Q}^2\mathbf{r}_0, \ldots, \mathbf{Q}^{m-1}\mathbf{r}_0\}, \tag{4.24}$$

with $\mathbf{r}_0 = \mathbf{d} - \mathbf{Q}\mathbf{x}_0$, where $\mathbf{x}_0$ is an initial guess of the solution. Now project the solution of $\mathbf{Q}\mathbf{m} = \mathbf{d}$ onto this subspace. The most widespread and known Krylov method is the conjugate gradient (CG) method, first published in Hestenes and Stiefel (1952). This method, can be used directly for solving (4.17) to find the posterior mean using the normal equations. There are, however, Krylov methods that may be better suited for least squares problems of this sort – in particular, the LSQR-method (Paige and Saunders, 1982), which has better convergence properties than using the normal equations directly, mainly coming from a condition number argument.

4.3.2 Non-linear inversion

For the non-linear case, the situation is a bit different. In this case, we can define the following objective function

$$\Phi(\mathbf{m}, \boldsymbol{\eta}) = f(\boldsymbol{\eta}) + (\mathbf{d} - \mathbf{W}q(\mathbf{m}))^T\mathbf{Q}_{lik}(\mathbf{d} - \mathbf{W}q(\mathbf{m})) + \lambda^2(\mathbf{m} - \boldsymbol{\mu})^T\mathbf{Q}_m(\mathbf{m} - \boldsymbol{\mu}) \tag{4.25}$$

where in general $f(\boldsymbol{\eta})$ involves the determinantal quantities of the model matrices and the prior distribution for the hyperparameters, $\boldsymbol{\eta}$. A vital

issue with this model is that the objective function Φ may be multimodal. We will abuse notation $\Phi(\mathbf{m}) = \Phi(\mathbf{m}, \boldsymbol{\eta})$, whenever $\boldsymbol{\eta}$ is treated as fixed.

In order to find $\arg\max_{\mathbf{m}} \Phi(\mathbf{m})$, we need to compute its gradient. We have the following partial derivatives, needed to compute the Jacobian of $q(\mathbf{m})$

$$\frac{\partial r_{PP}}{\partial m_1} = \frac{1}{2\cos^2(\theta)} \tag{4.26}$$

$$\frac{\partial r_{PP}}{\partial m_2} = -4\sin^2(\theta)\gamma^2 + \frac{\tan(\theta)\gamma}{1-\gamma^2\sin^2(\theta)}\big\{8\gamma^2(1-\sin^2(\theta)(1+\gamma^2))m_2 - 4\gamma^2(1-\sin^2(\theta)(3/2+\gamma^2))m_3\big\} \tag{4.27}$$

$$\frac{\partial r_{PP}}{\partial m_3} = \frac{1}{2}\tan^2(\theta)(1-4\gamma^2\cos^2(\theta)) + \frac{\tan(\theta)\gamma}{1-\gamma^2\sin^2(\theta)} + \big\{-4\gamma^2(1-\sin^2(\theta)(3/2+\gamma^2))m_2 + 2(\gamma^2(1-(2+\gamma^2)\sin^2(\theta)-1/4)m_3\big\} \tag{4.28}$$

It is important to note that the Jacobian, $Dq(\mathbf{m})^T = \mathbf{A}^T$ in the linear case and hence only needs to be populated once. For the non-linear case, it changes for each iteration and it becomes too expensive to populate the matrix each time, and hence we only compute its action on a vector.

The gradient of the objective function is

$$\nabla\Phi(\mathbf{m}) = Dq(\mathbf{m})^T\mathbf{W}^T\mathbf{Q}_{lik}(\mathbf{W}q(\mathbf{m}) - \mathbf{d}) + \lambda^2\mathbf{Q}_m(\mathbf{m} - \boldsymbol{\mu}_m). \tag{4.29}$$

Now, to employ a Newton-Krylov type method to solve this optimisation problem, we would also need the Hessian matrix, $\mathbf{H}(\mathbf{m}) = D^2\Phi(\mathbf{m})$. However, it is possible to overcome this challenge by using an inexact Hessian matrix vector product. We use the finite difference approximation as in Kelley (1999),

$$D^2\Phi(\mathbf{m} : \mathbf{p}) = \frac{\nabla\Phi(\mathbf{m} + h\mathbf{p}/\|\mathbf{p}\|) - \nabla\Phi(\mathbf{m})}{h/\|\mathbf{p}\|}, \tag{4.30}$$

where h is the step-length for the finite difference approximation, and $\mathbf{p}$ is the direction in which this finite difference is computed. It is also possible to

use a complex-step derivative approximation, if greater accuracy is needed. This is given by

$$D^2\Phi(\mathbf{m}:\mathbf{p}) = \frac{\mathrm{Im}(\nabla\Phi(\mathbf{m} + i\,h\,\mathbf{p}/\|\mathbf{p}\|)}{h/\|\mathbf{p}\|}. \tag{4.31}$$

where Im takes out the imaginary part in the expression, and i denotes the imaginary unit. In this expression, h can be set extremely low, say $h = 10^{-300}$ in order to achieve very high accuracy. This can be done since no subtractive cancellation occurs. The expression is, however, more computationally demanding than the usual finite differences. The first of these is faster and the preferred inexact method when the condition number of the resulting system is not too large. This depends on the total noise level of the process.

For the non-linear case, Krylov methods enter in a natural way. In the classical Newton algorithm (see e.g. Kelley (1999); Nocedal and Wright (1999)), the linear system

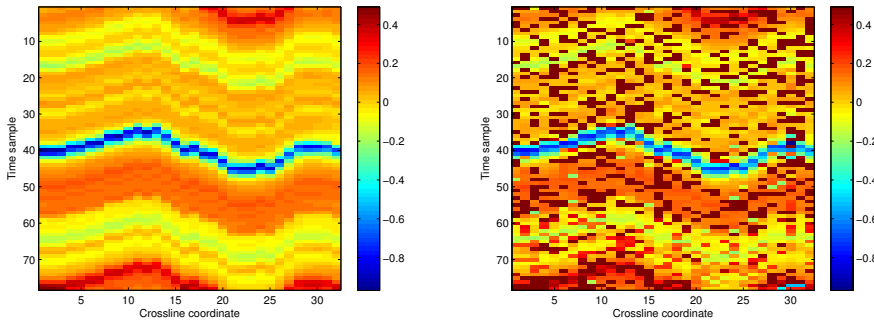
$$D^2\Phi(\mathbf{m}_k)\mathbf{p}_k = -\nabla\Phi(\mathbf{m}_k) \tag{4.32}$$

must be solved for each outer iteration. Here k is denotes the kth outer iteration and $\mathbf{p}_k$ is the kth search direction. In our case, the exact Hessian is not available, and we use the approximate Hessian matrix vector product, (4.30), in the Krylov method. Additionally, it is sufficient to let $\mathbf{r}_k = \|D^2\Phi(\mathbf{m}_k)\mathbf{p}_k + \nabla\Phi(\mathbf{m}_k)\| < \eta_k\|\nabla\Phi(\mathbf{m}_k)\|$ to get a convergent algorithm. Methods based on this criterion are called inexact Newton algorithms. See Kelley (1999); Nocedal and Wright (1999) for an overview of their properties. The Krylov method of choice in the inner iterations could be both the CG algorithm, based on the "local" normal equations, or a more sophisticated LSQR algorithm. In our computations, we have used the CG method.

The objective function, $\Phi(\mathbf{m})$, is inherently multimodal for small λ^2. To see this, simply initiate a Newton algorithm with a random starting vector and see how the norm of the gradient quickly goes to zero and does not yield the vector $\mathbf{m}^*$ that generated $\mathbf{d}$. In Figure 4.3, we see an example of this.

It occurs merely because as a multivariate quartic polynomial, $\Phi(\mathbf{m})$ may have several optima in some of its variables.

Figure 4.3: *True m_2 (left), estimated m_2 from random starting position terminated at $\|\nabla\Phi(\mathbf{m})\|_2 < 10^{-3}$ (right)*



Our hopes lie in the following: that our starting vector, generated from the linear approximation is in the domain of attraction of the physical solution to the optimisation problem. Additionally, our prior for $\mathbf{m}$ convexifies the the optimisation problem and helps in identifying the correct solution. We note that neither of these strategies guarantee that we find the correct optimum.

Comparing the 3-D model presented here to the 2-D problem treated in Rabben and Ursin (2009), we observe that the introduction of the wavelet dramatically increases the condition number of the system over that in Rabben and Ursin (2009). Without a very good preconditioner, full inversion with estimation of hyper-parameters both in the forward model and precision matrices is infeasible.

4.3.3 Preconditioners

One of the most important aspect of getting a non-linear inversion problem of this type to work on a large scale is the availability of adaptive preconditioners for the Krylov-step in the optimisation algorithm. Potential preconditioners for a system related to the one defined by (4.25) will need to adapt to changing $\mathbf{m}$, need to be fast to compute, and need to approximate the inverse of the sum of two matrices, $\mathbf{M} \approx (\mathbf{F}+\mathbf{Q}_m)^{-1}\Big|_{\mathbf{m}_{cur}}$, in each iteration, where $\mathbf{M}$ is the preconditioner. $\mathbf{F}$ in this case is either $\mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}\mathbf{A}$ or $Dq(\mathbf{m})^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}Dq(\mathbf{m})$, where differentiation is taken with respect to the current value of $\mathbf{m}$. Note that the usual BFGS-type Newton scheme (see e.g. Nocedal and Wright (1999)) is too demanding in this application since the computational demands for approximate Hessians are infeasible.

Candidates that are feasible for our systems are ones based on operator splitting and ones based on the discrete cosine transform (DCT). For the splitting approach, the preconditioner has the following form

$$\mathbf{M} = \Big((p(\mathbf{F})+\gamma\mathbf{I})^{1/2}(p(\mathbf{Q}_m)+\gamma\mathbf{I})(p(\mathbf{F})+\gamma\mathbf{I})^{1/2})\Big)^{-1} \tag{4.33}$$

where p defines some preconditioning operation on the respective matrices and γ is a tuning parameter. By construction $\mathbf{F}$ has a structure that makes it easy to approximate $\mathbf{F}^{-1/2}\mathbf{b}$ quickly and explicitly, and $\mathbf{Q}_m$ is very close to being diagonalised by the DCT, since the discrete Laplacian with Neumann boundaries is diagonalised by this transform. In practice, it is hard to get this split preconditioner to work on our system.

Another possibility is to use a DCT-type preconditioner on both matrices simultaneously, but this is highly dependent on how $\mathbf{F}$ reacts to such preconditioners. In this case, we approximate the full operator $\mathbf{F}+\mathbf{Q}_m$ by $c(\mathbf{F}+\mathbf{Q}_m) = c(\mathbf{F})+c(\mathbf{Q}_m)$, where c denotes the optimal DCT preconditioner. The optimal DCT preconditioner is defined by $c(\mathbf{F}) = \arg\min_{\mathbf{B}\in\mathcal{B}} \|\mathbf{F}-\mathbf{B}\|_F$, where $\mathcal{B}$ is the n-dimensional space of matrices diagonalizable by the DCT, and $\|\cdot\|_F$ is the Frobenius norm. A method to construct this preconditioner

is described in Chan et al. (1999). The method described therein extends to 3-D in the obvious way. We are then left the eigenvalue matrices for the operators $c(\mathbf{F})$ and $c(\mathbf{Q}_m)$, denoted by $\mathbf{\Gamma}^c_F, \mathbf{\Gamma}^c_{\mathbf{Q}_m}$. The preconditioner is then $\mathbf{M} = \mathcal{C}(\mathbf{\Gamma}^c_F + \mathbf{\Gamma}^c_{\mathbf{Q}_m})^{-1}\mathcal{C}^T$, where $\mathcal{C}$ denotes the DCT matrix. In practice this is computed in sequence by the fast 3-D DCT. This preconditioner is also easy to split, if that is required for the Krylov method in question. Our experience is that this preconditioner works well for the problem we investigate, and it is our default choice.

In order to apply the preconditioners on the non-linear system, updating of the preconditioners must be feasible. In practice, it may be sufficient to update the operator once in a while in the outer iterations of the Krylov-Newton method.

4.4 Numerical results

In this section we compute and compare posterior estimates for $\mathbf{m}$ for different models. For testing potential performance, we use the following specifications for the physical model

- Low relative differences in elastic parameters and small angles
- Low relative differences in elastic parameters and large angles
- One jump with high relative differences in elastic parameters and small angles
- One jump with high relative differences in elastic parameters and large angles

Here, the small angles are given by $\theta_{\text{small}} = (5, 10, 25)$ and large angles are given by $\theta_{\text{large}} = (10, 25, 35)$. The maximum absolute value of relative differences where we have low relative differences is $m^{\text{low}}_{\text{max}} =$, and for high relative differences, $m^{\text{high}}_{\text{max}} =$. We invert for elastic parameters using three different strategies

- Stationary linear model
- Full linear model – that is, non-stationary prior and likelihood
- Quadratic model

The data is generated from the quadratic model with non-stationarity for all test cases, and we present the results as follows: We start by examining the noiseless inversion case to demonstrate its potential, but keep in mind the multi-modality of the objective function it induces (Figure 4.3). We do the same for the non-stationary model. Thereafter, we give inversion results for the quadratic model and its differences from the linear model using noisy data to illustrate what we actually get – we do the same for the linear non-stationary model and its stationary approximation. To conclude, we give parameter sweeps of inversions using different noise levels to see potential gains from using a quadratic or non-stationary model over their respective counterparts. We do not do a sweep for the case where we have low relative differences and small angles in the quadratic vs. linear nor stationary vs. non-stationary case.

4.4.1 Inversion examples – quadratic vs. linear and stationary vs. non-stationary

The noiseless model gives a standard (non-linear) least-squares problem. The reasoning for considering the noiseless model is the following: in order to assess what potential benefits for using a more complex model in the noisy case, it is beneficent to see what happens in the noiseless case. The data in this case is generated without the noise components. The performance criteria are relative differences in the ℓ_2-norm and the ℓ_∞-norm. We depict this in Figure 4.4, where the true parameters are on the left, and the difference between those coming from a quadratic and a linear inversion on the right, where time-east is on the upper panel and east-west on the lower. In the noiseless case, the non-linear inversion procedure recovers the parameters exactly – as expected.

Figure 4.4: *Noiseless inversion, large contrasts, large angles. True parameters (left), difference quadratic - linear (right)*

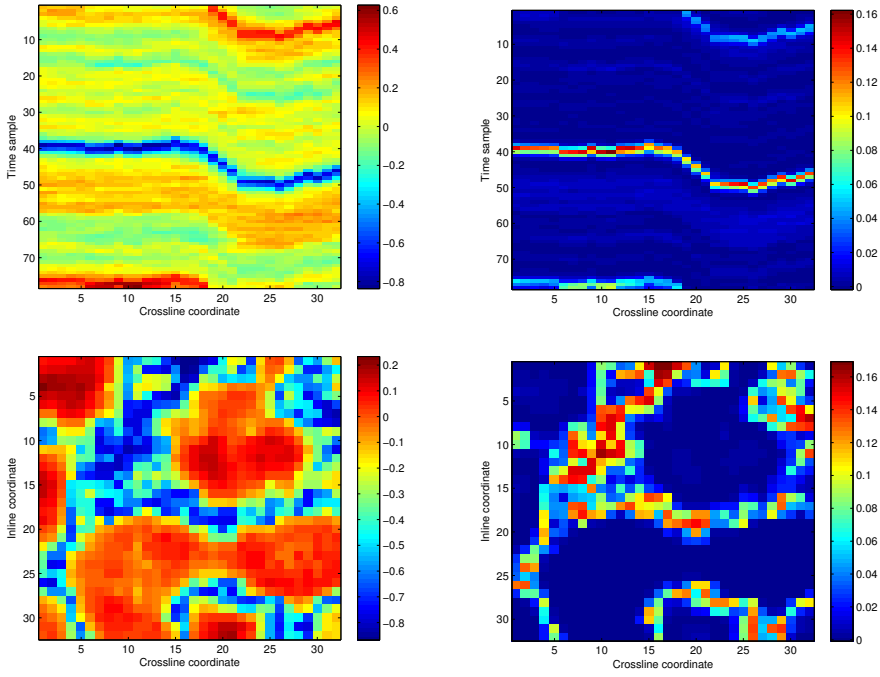
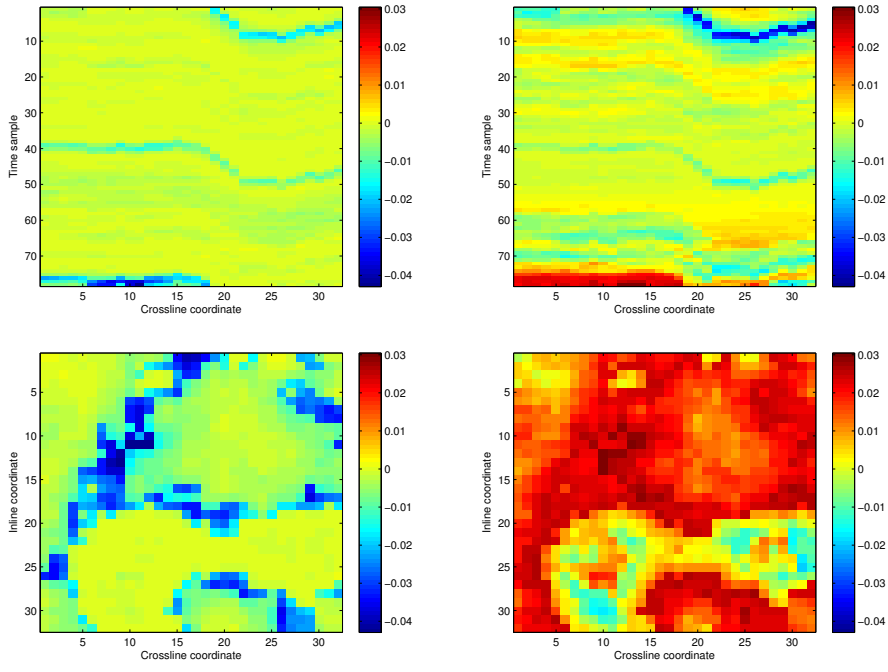


Figure 4.5: *Noiseless inversion, low contrasts, small angles. Differences non-stationary (left), differences stationary (right)*



We also include a noiseless inversion for the non-stationary versus stationary case for small angles and low contrasts. Figure 4.5 depicts deviations from the truth, and it is clearly seen that the stationary approximation is worse than the non-stationary one, albeit not very much worse.

We give several visualisations in the noisy case, and we summarise the models in Table 4.1. In all the figures, the left panels give inversion results for the more sophisticated model, i.e. quadratic when the quadratic model is compared to the full linear one, and the full linear one when this is compared to the stationary one. In the right panel, the difference in inversion results is shown, i.e. the inversion results of the more sophisticated model minus

Table 4.1: *Table of different models and their respective figures*

Contrasts	Angles	Quadratic	Stationary	Figures	Sweep figures
Low	Small	No	Yes	4.5,4.10	4.18,4.19
Low	Small	No	No	4.5,4.10,4.6	4.18,4.19
Low	Small	Yes	No	4.6	No
Low	Large	No	No	4.7	4.12,4.13
Low	Large	Yes	No	4.7	4.12,4.13
High	Small	No	No	4.8	4.14,4.15
High	Small	Yes	No	4.8	4.14,4.15
High	Large	No	No	4.4,4.9	4.16,4.17
High	Large	Yes	No	4.4,4.9	4.16,4.17

the results for the less sophisticated one.

In Figure 4.9 we see results for one layer with high contrasts and relatively large angles. The same trends as from the noiseless inversion is apparent here – in the high contrast layer, the differences are high, while in the other areas, the inversion results are virtually identical. Obviously, the effects from the prior comes into play, giving smoother images than for the noiseless case. In the subsequent figures, we see that the large differences in inversion results come where there are large contrasts. In Figure 4.6, where we have small angles and low contrasts, the inversion results are very close to one another.

Comparing the differences between inversion results in the stationary vs. non-stationary case (Figure 4.10), we see that there are differences in the inversion results, but they are not localised only at interfaces.

Marginal variances

For completeness, we include marginal variance estimates for the quadratic model at the mode. The estimates are given in Figure 4.11. Here we clearly see that the marginal variance estimates are affected by the estimated

Figure 4.6: *Noisy inversion, low contrasts, small angles. Quadratic inversion (left), differences (right)*

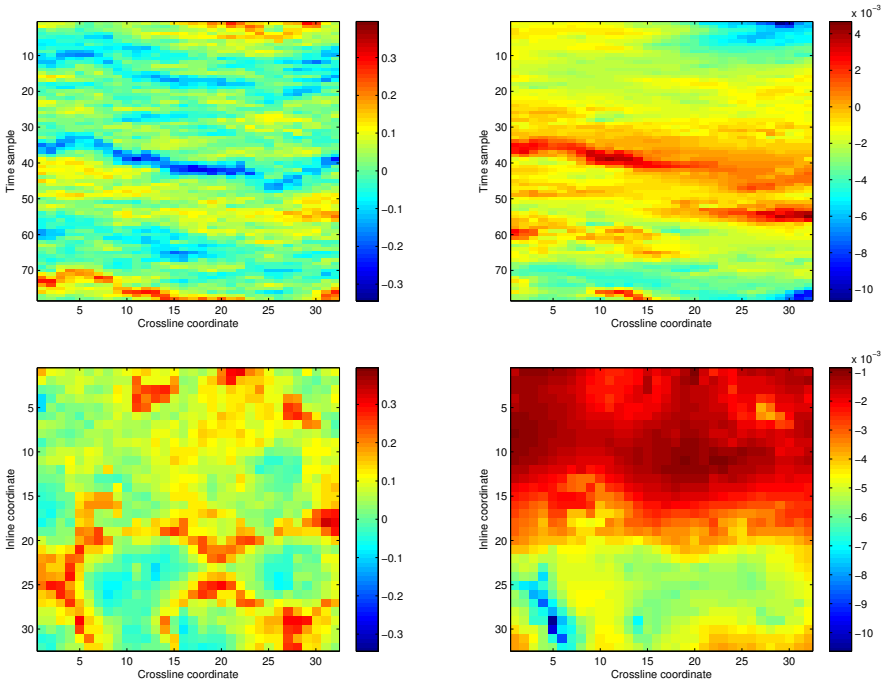


Figure 4.7: *Noisy inversion, small contrasts, large angles. Quadratic inversion (left), differences (right)*

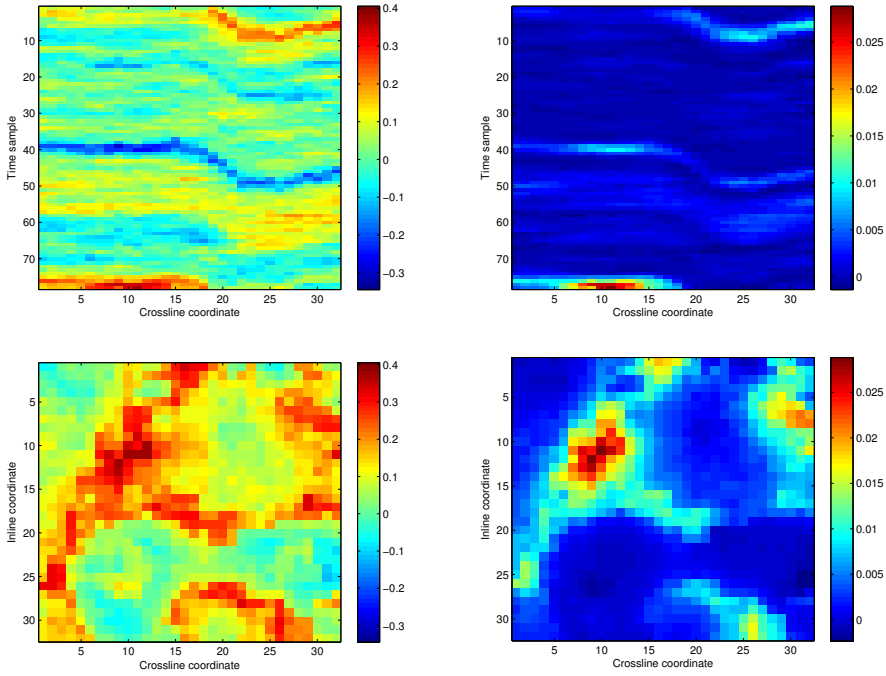


Figure 4.8: *Noisy inversion, large contrasts, small angles. Quadratic inversion (left), differences (right)*

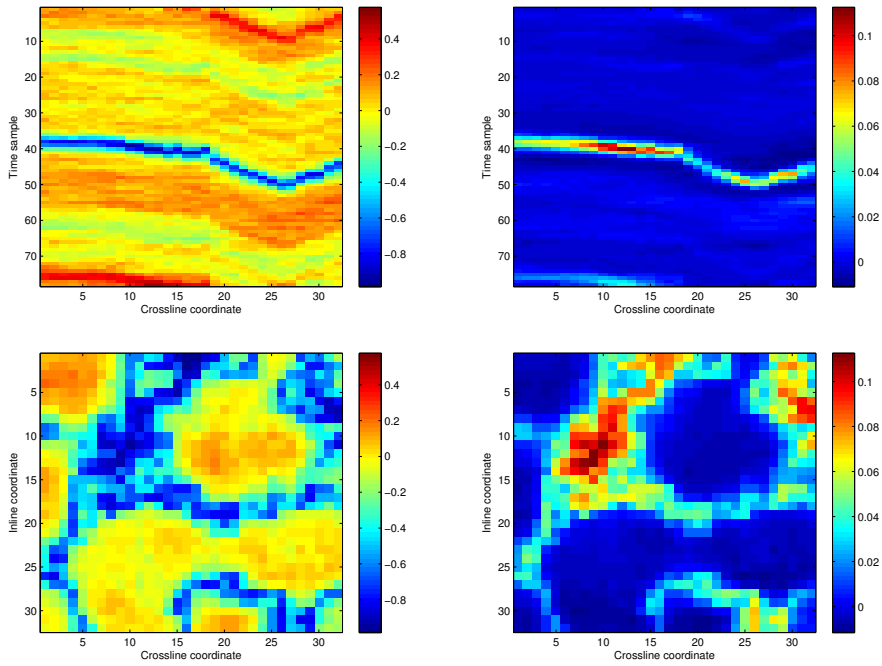


Figure 4.9: *Noisy inversion, large contrasts, large angles. Quadratic inversion (left), differences (right)*

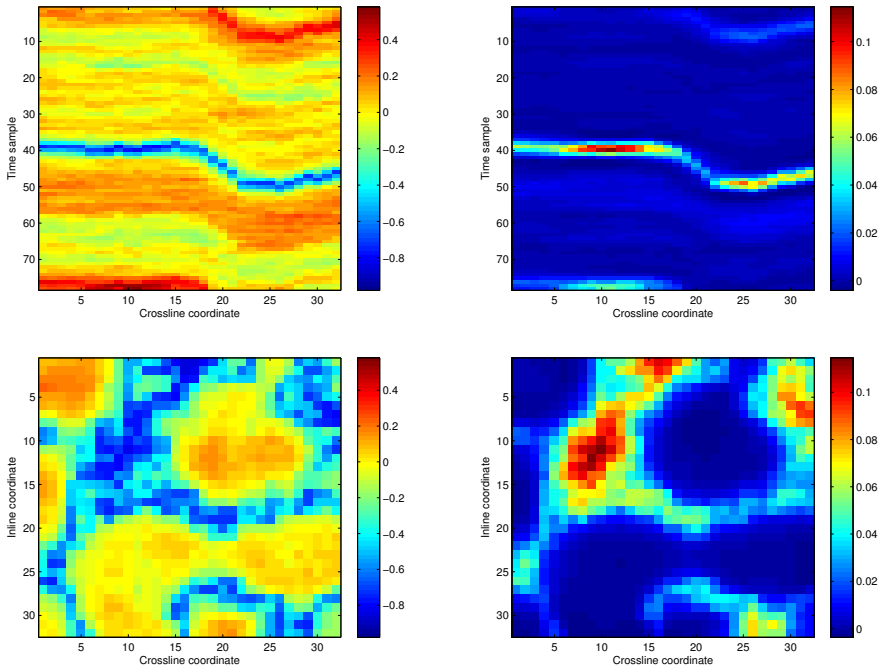


Figure 4.10: *Noisy inversion, low contrasts, small angles. Non-stationary inversion (left), differences (right)*

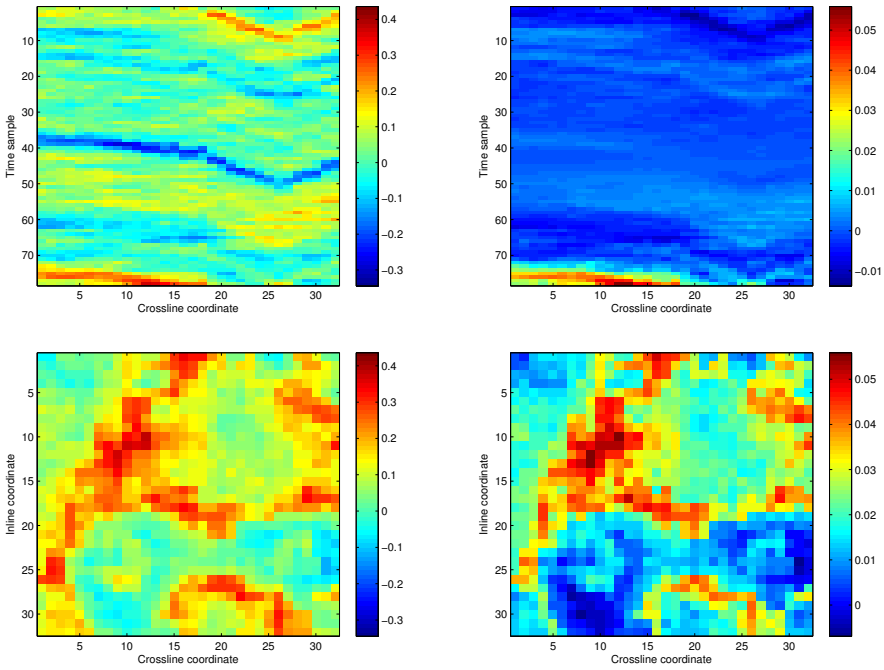
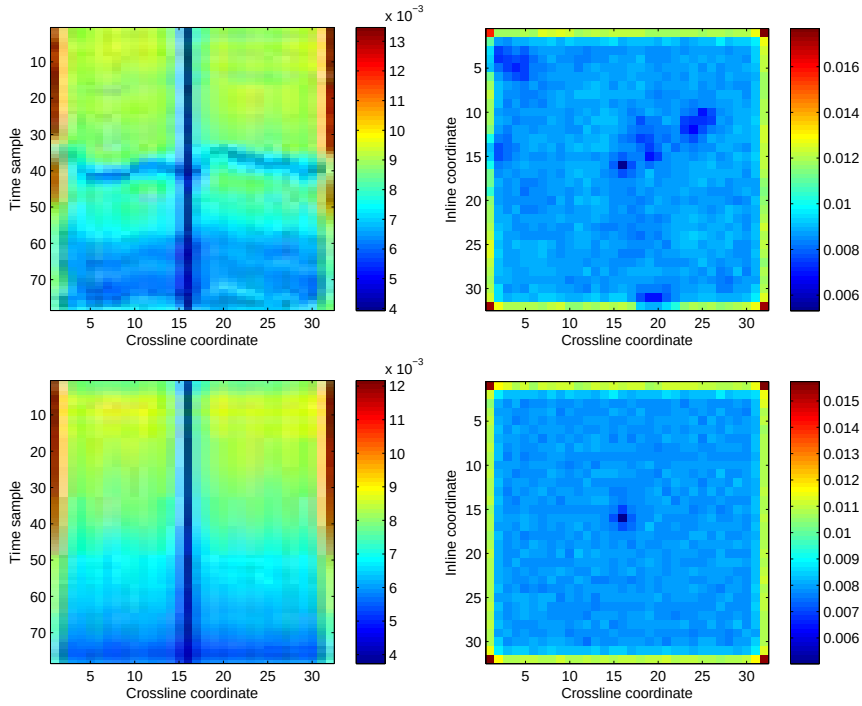


Figure 4.11: *Estimate for marginal variances, quadratic inversion.*



parameters in the quadratic model, and that at the well, the variance is much less than everywhere else. Comparing this to the marginal variances from the linear model, we see that there is clearly a difference at the high contrast interface, and notably, the variances adapt to the data in a specific way.

4.4.2 Parameter sweep

In order to identify where the non-linear model yields superior estimates to that of the linear one, we employ a parameter sweep over the two components,

σ_1^2, σ_2^2. With increasing noise levels, we expect the prior information to dominate. For very low noise levels, however, the non-linear model should yield superior estimates. This is done for the four scenarios discussed in the previous section. Some particular quantities of interest are

$$f_2(\sigma_1^2, \sigma_2^2) = \frac{\|\mathbf{m} - \hat{\mathbf{m}}^{\text{quad}}\|_2}{\|\mathbf{m} - \hat{\mathbf{m}}^{\text{lin}}\|_2}(\sigma_1^2, \sigma_2^2)$$
$$f_\infty(\sigma_1^2, \sigma_2^2) = \frac{\|\mathbf{m} - \hat{\mathbf{m}}^{\text{quad}}\|_\infty}{\|\mathbf{m} - \hat{\mathbf{m}}^{\text{lin}}\|_\infty}(\sigma_1^2, \sigma_2^2). \tag{4.34}$$

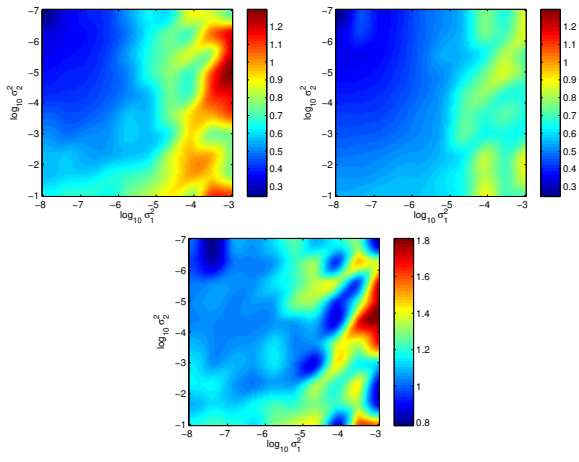
These two quantities are the relative mismatch in ℓ_2- and ℓ_∞-norm. Here, $\mathbf{m}$ contains the elastic parameters used for data generation, where the data is generated using the quadratic model. Moreover, $\hat{\mathbf{m}}^{\text{lin}}$ and $\hat{\mathbf{m}}^{\text{quad}}$ are estimated parameters using the linear modeling assumption and quadratic modeling assumption respectively.

In our sweep, we estimate parameters from scenarios where the noisy data essentially is indistinguishable from noiseless data to scenarios where the noiseless data is very different to that with noise. The two noise components are defined with levels $(\sigma_1^2, \sigma_2^2) \in [5 \cdot 10^{-8}, 10^{-3}] \times [5 \cdot 10^{-7}, 5 \cdot 10^{-1}]$ – quite a big range of different noise levels.

The figures for low contrasts and large angles are Figure 4.12 and 4.13. In on the top of the first figure, we see that for ℓ_2-loss, the quadratic inversion gives a smoother pictures, with lower errors at high noise levels which is counter-intuitive to what we would expect. We cannot give any good explanation for this behaviour. For ℓ_∞-loss, the situation is more along what we would expect, with better predictions in the low-noise region.

In Figure 4.14 and 4.15, we also have some erratic behaviour in th ℓ_2 case, where we have some islands where the quadratic inversion does much better for moderate noise levels. In the ℓ_∞ case, the expected behaviour in the low noise region is seen, with quadratic inversion being better than the linear one. The quadratic inversion for ℓ_∞-loss also produces quite a smooth error landscape, while the ℓ_2 is less smooth.

Figure 4.12: *Parameter sweep, low contrasts, large angles, ℓ_2-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_2/\|\mathbf{m}_{true}\|_2$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_2/\|\mathbf{m}_{true}\|_2$. *Bottom:* $f_2(\sigma_1^2, \sigma_2^2)$



In Figure 4.16 and 4.17 the sweeps are shown,for large contrasts and angles. This is the scenario in which we expect to gain the most from using the quadratic approximations in (4.1). In the upper left corner of the two norm comparisons, the noise level is so low that the quadratic approximation is much better than the linear one, but this quickly changes as the noise levels increase. However, for the large contrasts, there is a rather large region on the left part of Figure 4.16,4.17 that gives better predictions for the quadratic approximation than the linear one. This is the critical area that determines whether it may be worth doing the extra work required. We believe this is situation dependent. In most cases, however, the predictions have similar accuracy, and we gain nothing by using the more intricate approximation. Of all the three sweeps, this one seems to be the one that correspond to our expectations the most.

In Figure 4.18,4.19, the relative improvement on using the non-stationary model over the stationary one is illustrated. Recall, that the non-stationarity is introduced in the prior precision at the well locations and in the likelihood

Figure 4.13: *Parameter sweep, low contrasts, large angles, ℓ_∞-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Bottom:* $f_\infty(\sigma_1^2, \sigma_2^2)$

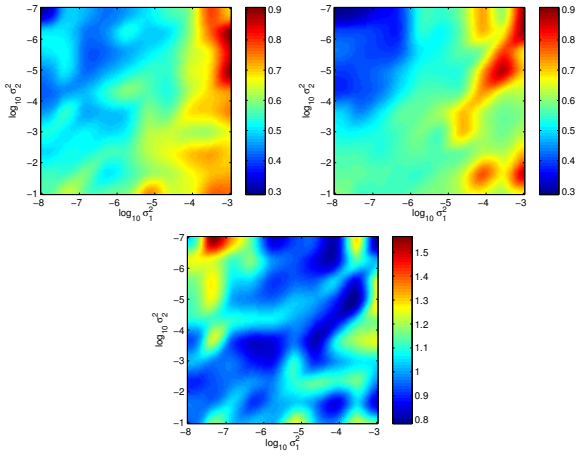


Figure 4.14: *Parameter sweep, high contrasts, small angles, ℓ_2-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Bottom:* $f_2(\sigma_1^2, \sigma_2^2)$

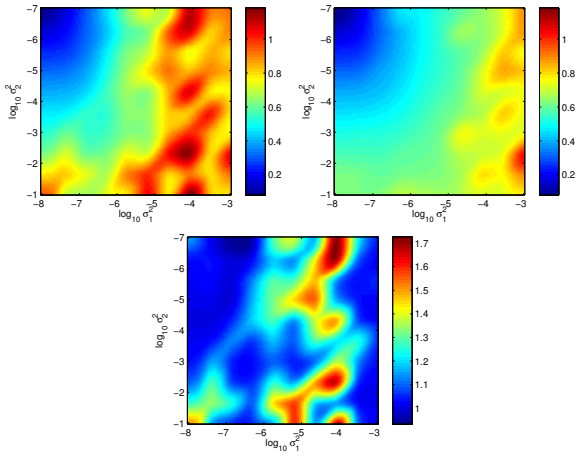


Figure 4.15: *Parameter sweep, high contrasts, small angles, ℓ_∞-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Bottom:* $f_\infty(\sigma_1^2, \sigma_2^2)$

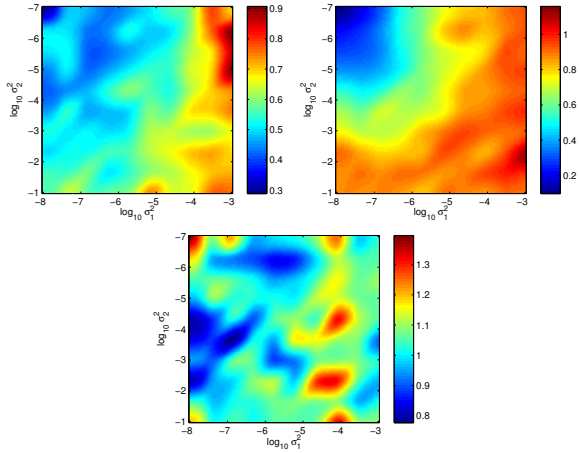


Figure 4.16: *Parameter sweep, high contrasts, large angles, ℓ_2-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Bottom:* $f_2(\sigma_1^2, \sigma_2^2)$

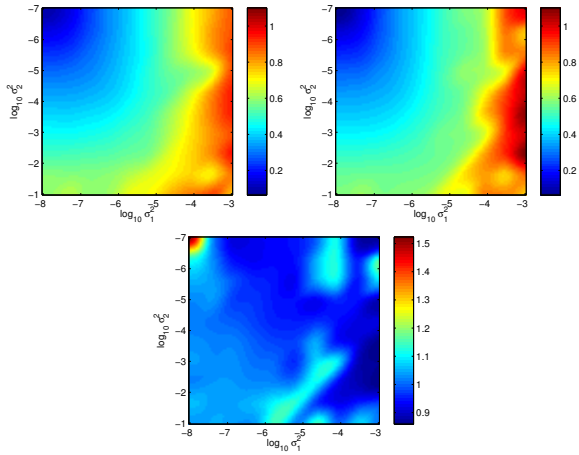
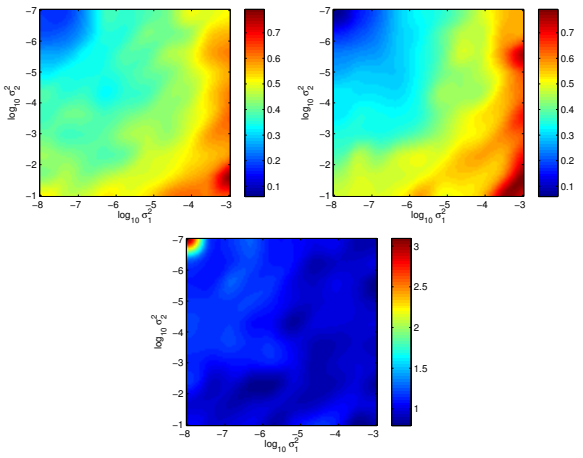


Figure 4.17: *Parameter sweep, high contrasts, large angles, ℓ_∞-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Right:* $\|\hat{\mathbf{m}}^{quad} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Bottom:* $f_\infty(\sigma_1^2, \sigma_2^2)$



through increasing variances with traveltime. Here, we see that essentially, the non-stationary one is better – especially for the ℓ_∞-norm, but as the noise levels increase, they perform similarly. The non-stationarity included in this model is not very severe, so this observation is not surprising, but for a more tailored one, e.g. the one that is described in Aune and Simpson (2012), the results may be more diverse.

4.5 Discussion and conclusion

The linear and quadratic models have been compared throughout this text for seismic AVA inversion in 3-D. Our formulation allows the inclusion of non-stationary Gaussian processes in the prior model or for the likelihood noise terms. This gives added flexibility over the current state of the art. It is, however, difficult to give an unanimous conclusion on using these models. Each have its own merit, and which to use depends very much on what is the goal of the inversion. Stationary inversion is extremely fast, and can easily

Figure 4.18: *Parameter sweep, non-stationary (left), stationary (right), ratio (bottom), ℓ_2-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Right:* $\|\hat{\mathbf{m}}^{stat} - \mathbf{m}_{true}\|_2 / \|\mathbf{m}_{true}\|_2$. *Bottom:* $f_2(\sigma_1^2, \sigma_2^2)$

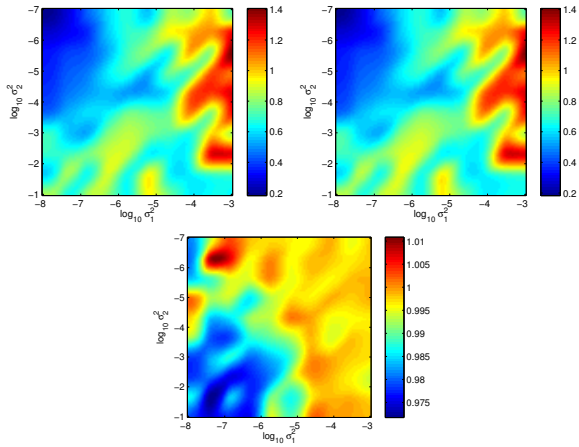
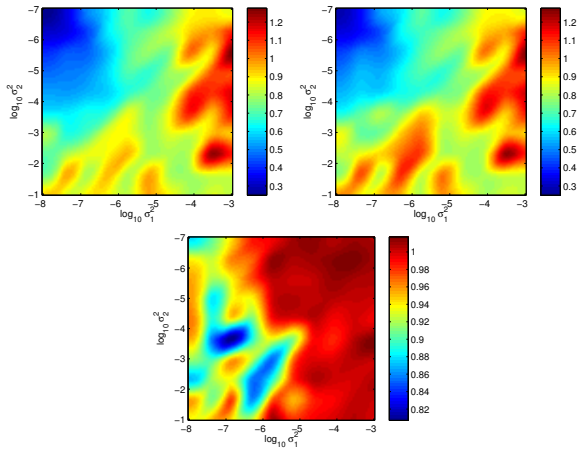


Figure 4.19: *Parameter sweep, non-stationary (left), stationary (right), ratio (bottom), ℓ_∞-loss. Left:* $\|\hat{\mathbf{m}}^{lin} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Right:* $\|\hat{\mathbf{m}}^{stat} - \mathbf{m}_{true}\|_\infty / \|\mathbf{m}_{true}\|_\infty$. *Bottom:* $f_\infty(\sigma_1^2, \sigma_2^2)$



be used on huge datasets, and while the non-stationary model discussed here can be almost as fast with the use of appropriate preconditioners. There are many more details, both in the physical and implementation domain, that need to be addressed in order to achieve equivalently fast inversions. For a rough estimate of the parameters, the good old stationary model may indeed be sufficient.

Inversion using the quadratic forward model is much slower than when using the linear non-stationary one, and needs as input an inversion from the linear non-stationary one in order to get consistent estimates, i.e. converging to the correct optimum. On the other hand, it may give better estimates of high-contrast interfaces, and even overall better estimates in general, depending on the noise levels. Optimally, we would like the speed of the linear non-stationary model and perhaps add local information from the quadratic model in specific areas of the subsurface. A potential candidate for this is to essentially use the non-stationary linear model everywhere, and extract high-contrast interfaces from this inversion and refine the inversion at these high contrast interfaces using the non-linear quadratic model. This is highly motivated by Figure 4.6-4.9, where we see that differences in inversion results using the respective models are local only for high contrast interfaces. The boundary conditions in this case should facilitate compatibility with the linear model, so that the inversion is consistent. We believe that future research on non-linear AVA inversion should focus on this aspect.

In this modeling framework, there are new important aspects to consider. How can we incorporate the linear inversion results into the non-linear one? Should we use it as initial conditions for a Newton method or use it as a pseudo-prior. If we use this inversion result as some pseudo-prior, what sort of boundary and mixing conditions should we use, and what precision matrix?

A point that is worth addressing is the summary of the performance of one inversion model over another. In this chapter, we proposed using a parameter sweep and thereafter to look at some relative error estimate over another. As we at best expect local improvements of one model over

another, it is worth asking if there are other performance criteria that are better suited for this comparison. Of course, the ℓ_∞-norm performance gives in a sense a maximal local comparison criterion, but one could imagine taking out high-contrast interfaces and looking at some average there. This strategy is, of course, hard to automate and requires a lot of manual input.

An advantage of the parameter sweep is that when estimating hyper-parameters, one can immediately identify whether it is any point doing any further inversion – if the estimated σ_1^2, σ_2^2 lie in an area in which non-linear inversion gives no benefit over the linear one, the natural choice is, naturally, to only do a linear one. Unfortunately, this may be field dependent, and appropriate sweeps should be made for each case study.

Overall, we believe that this work highlights both difficulties and benefits from using the different models we have addressed.

4.6 Addendum: Derivation of linear precision structure and conditional expectation

For the Gauss-linear model, we have the following joint covariance structure for $\mathbf{m}$, $\mathbf{WAm} + \mathbf{W}\boldsymbol{\epsilon}_1\mathbf{W}^T + \boldsymbol{\epsilon}_2$ and $\mathbf{d}_3 = \mathbf{Gm} + \boldsymbol{\epsilon}_3$

$$\mathrm{Cov}\begin{pmatrix} \mathbf{m} \\ \mathbf{d} \\ \mathbf{d_w} \end{pmatrix} = \begin{pmatrix} \mathbf{Q}_m^{-1} & \mathbf{Q}_m^{-1}\mathbf{A}\mathbf{W}^T & \mathbf{Q}_m^{-1}\mathbf{G}^T \\ \mathbf{WAQ}_m^{-1} & \mathbf{WAQ}_m^{-1}\mathbf{A}^T\mathbf{W}^T + \mathbf{Q}_{lik}^{-1} & \mathbf{WAQ}_m^{-1}\mathbf{G}^T \\ \mathbf{GQ}_m^{-1} & \mathbf{G}^T\mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{GQ}_m^{-1}\mathbf{G}^T + \mathbf{Q}_3^{-1} \end{pmatrix} \tag{4.35}$$

Now, block inversion of this matrix for $\mathrm{Cov}(\mathbf{m},\mathbf{d},\mathbf{d}_3)^{-1} = \begin{pmatrix} \mathbf{Q}_{m|d,d_3} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{D} \end{pmatrix}$ gives

$$\begin{aligned}
\mathbf{Q}_{m|d,d_3} &= \mathbf{Q}_m + \mathbf{Q}_m \begin{pmatrix} \mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{Q}_m^{-1}\mathbf{G}^T \end{pmatrix} \\
&\quad \left(\mathrm{Cov}(\mathbf{d},\mathbf{d}_3) - \begin{pmatrix} \mathbf{W}\mathbf{A}\mathbf{Q}_m^{-1} \\ \mathbf{G}\mathbf{Q}_m^{-1} \end{pmatrix} \mathbf{Q}_m \begin{pmatrix} \mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{Q}_m^{-1}\mathbf{G}^T \end{pmatrix}\right)^{-1} \\
&\quad \begin{pmatrix} \mathbf{W}\mathbf{A}\mathbf{Q}_m^{-1} \\ \mathbf{G}\mathbf{Q}_m^{-1} \end{pmatrix} \mathbf{Q}_m \\
&= \mathbf{Q}_m + \begin{pmatrix} \mathbf{A}^T\mathbf{W}^T & \mathbf{G}^T \end{pmatrix} \\
&\quad \left(\mathrm{Cov}(\mathbf{d},\mathbf{d}_3) - \begin{pmatrix} \mathbf{W}\mathbf{A}\mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{W}\mathbf{A}\mathbf{Q}_m^{-1}\mathbf{G}^T \\ \mathbf{G}^T\mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{G}\mathbf{Q}_m^{-1}\mathbf{G}^T \end{pmatrix}\right)^{-1} \begin{pmatrix} \mathbf{W}\mathbf{A} \\ \mathbf{G} \end{pmatrix} \\
&= \mathbf{Q}_m + \begin{pmatrix} \mathbf{A}^T\mathbf{W}^T & \mathbf{G}^T \end{pmatrix} \begin{pmatrix} \mathbf{Q}_{lik}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_3^{-1} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{W}\mathbf{A} \\ \mathbf{G} \end{pmatrix} \\
&= \mathbf{Q}_m + \mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik}\mathbf{W}\mathbf{A} + \mathbf{G}^T\mathbf{Q}_3\mathbf{G} \qquad (4.36)
\end{aligned}$$

The off-diagonal block yields

$$\begin{aligned}
\mathbf{B}^T &= -\mathbf{Q}_m \begin{pmatrix} \mathbf{Q}_m^{-1}\mathbf{A}^T\mathbf{W}^T & \mathbf{Q}_m^{-1}\mathbf{G}^T \end{pmatrix} \begin{pmatrix} \mathbf{Q}_{lik}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_3^{-1} \end{pmatrix}^{-1} \\
&= -\begin{pmatrix} \mathbf{A}^T\mathbf{W}^T\mathbf{Q}_{lik} & \mathbf{G}^T\mathbf{Q}_3 \end{pmatrix} \qquad (4.37)
\end{aligned}$$

and lastly,

$$\mathbf{D} = \begin{pmatrix} \mathbf{Q}_{lik} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_3 \end{pmatrix} \qquad (4.38)$$

The conditional expectation is hence given by

$$\boldsymbol{\mu}_{m|d,d_3} = \mathbf{Q}_{m|d,d_3}^{-1}(\mathbf{Q}_m \quad -\mathbf{B}^T) \begin{pmatrix} \mathbf{m} \\ \mathbf{d} \\ \mathbf{d}_3 \end{pmatrix} \qquad (4.39)$$

Chapter 5

The use of systems of stochastic PDEs as priors for multivariate models with discrete structures

A challenge in multivariate problems with discrete structures is the inclusion of prior information that may differ in each separate structure. A particular example of this is seismic amplitude versus angle (AVA) inversion to elastic parameters, where the discrete structures are geologic layers. Recently, the use of systems of linear stochastic partial differential equations (SPDEs) have become a popular tool for specifying priors in latent Gaussian models. This approach allows for flexible incorporation of non-stationarity and anisotropy in the prior model. Another advantage is that the prior field is Markovian and therefore the precision matrix is very sparse, introducing huge computational and memory benefits. We present a novel approach for parametrising correlations that differ in the different discrete structures, and additionally a geodesic blending approach for quantifying fuzziness of interfaces between the structures.

5.1 Introduction

In spatial statistics, the need for specifying different behaviour in different regions in space is crucial for making a good prior model. The literature is abundant with methodologies for this. In the multivariate setting, this generalises to having different correlations between the fields in different regions and different cross-differentiability properties.

A particular model problem where this is important is the seismic AVA inversion problem, which is well studied in the geophysical literature. There are several incarnations of this problem with varying degrees of complexity. In this chapter, our primary example is the inversion problem studied in Buland and Omre (2003); Buland et al. (2003); Rabben et al. (2008), using the wave-field propagation approximations in Aki and Richards (1980), which results in linear systems of equations to solve. Variants and extensions of these equations are found in Stovas and Ursin (2003), including non-linear approximations that may yield better inversion results in some situations. We exemplify our contributions using this example explicitly throughout the chapter.

The model we adopt in this text is the same as in Buland and Omre (2003), which is essentially

$$d(\mathbf{s}) = w \star r_{PP}(\mathbf{m})(\mathbf{s}) + \epsilon, \tag{5.1}$$

where $\star$ denotes convolution in time, w is an approximation to the source wavelet – i.e. the shape of the wave travelling through the subsurface, and $r_{PP}(\mathbf{m})$ denotes a reflectivity operator. The reflectivity operator takes relative differences in elastic parameters to reflection coefficients for the wave. We adopt the following elastic parameters,

$$m_1 = \frac{\triangle v_P}{\overline{v}_P}, \quad m_2 = \frac{\triangle v_S}{\overline{v}_S}, \quad m_3 = \frac{\triangle \rho}{\overline{\rho}}. \tag{5.2}$$

I.e. m_1, m_2, m_3 denotes the relative difference of P-velocity (pressure wave velocity), S-velocity (shear wave velocity) and density respectively, and the

reflectivity operator is defined by

$$r_{PP}(\theta) = \frac{m_1}{2}(1 + \tan^2\theta) - 4m_2\gamma^2 \sin^2\theta + \frac{m_3}{2}(1 - 4\gamma^2 \sin^2\theta), \qquad (5.3)$$

there θ is the reflection angle and γ^2 denotes a background (v_S/v_P)-ratio. Rewriting this in matrix notation yields

$$\mathbf{d} = \mathbf{WAm} + \boldsymbol{\epsilon}, \qquad (5.4)$$

where $\mathbf{d}$ are observations, $\mathbf{W}$ is the discretized wavelet operator, $\mathbf{A}$, the discretized reflectivity operator, $\mathbf{m}$ the elastic parameters and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ an error term which is often assumed to be normally distributed.

In this text, we will explore a novel method for designing a good prior for $\mathbf{m}$ using linear systems of stochastic partial differential equations. We emphasize, however, that while the approach developed here is designed with seismic AVA inversion in mind, it is very flexible and can be adopted in any setting where we have multivariate fields with separate regions where we would like to incorporate prior information.

All the figures that appear in this text have comparative scales, so that the colour schemes have the same min-max values in each individual figure. Hence, the figures makes sense, without cluttering them with additional colour bars.

5.2 Prior specification

The choice of prior in the inversion problem is of great importance when it comes to the performance of the inversion. It is vital to choose a "good" prior to emphasise the properties of $\mathbf{m}$ that we know it has. For us, $\mathbf{m}$ will denote the parameters of interest, and it depends on position. We construct the prior by combining heuristics and expert knowledge of the spatial model. For a Gaussian prior model, the standard way of specifying the prior model is through the covariance function, which is often assumed to be stationary

(see, e.g. Buland and Omre (2003)). A stationary covariance function is defined by a correlation function that defines how much a point is correlated with its neighbours and a marginal variance parameter, ϱ^2 through

$$\varrho^2 c(\|\mathbf{x}-\mathbf{y}\|_{\mathbf{A}}) = \mathrm{Cov}(\mathbf{x},\mathbf{y}), \tag{5.5}$$

where $\mathbf{H}$ is a positive definite matrix that defines the non-changing anisotropy of the field. In the Gaussian case, this defines a strictly stationary process if the mean is constant. There is a list of widely used covariance functions in Cressie (1993). We will throughout this text assume that the prior is from the Gaussian family. This family is defined by having density

$$p(\mathbf{x}|\mathbf{Q},\boldsymbol{\mu}_x) = (2\pi)^{-n/2}\det(\mathbf{Q})^{1/2}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_x)^T\mathbf{Q}(\mathbf{x}-\boldsymbol{\mu}_x)\right), \tag{5.6}$$

where $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix – the inverse of the covariance matrix $\boldsymbol{\Sigma}$ – and $\boldsymbol{\mu}_x$ is the expectation, $\mathbb{E}(\mathbf{x}|\boldsymbol{\mu}_x)$.

Moreover, the fields m_1, m_2, m_3 are assumed correlated with correlations specified by well data and/or other local knowledge. In the discretized domain, this allows for the following decomposition of the total covariance matrix

$$\boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}_{\text{space}} \otimes \boldsymbol{\Sigma}_0 \tag{5.7}$$

where $\boldsymbol{\Sigma}_{\text{space}}$ denotes the spatial covariance matrix, typically defined through a covariance function, and $\boldsymbol{\Sigma}_0$ the correlations between the elastic parameters. Since seismic observations typically are on a regular grid, either in 2-D or 3-D, it is possible to let $\boldsymbol{\Sigma}_{\text{space}}$ be circulant by extending the grid by as many points as is needed to get the correlation below a threshold – typically 0.1 or 0.05. This allows us to use fast Fourier transforms for computing quantities of interest related to the covariance matrix. This, together with the Kronecker structure of $\boldsymbol{\Sigma}_m$ allows for fast computations. See Buland et al. (2003); Rue and Held (2005); Gray (2006) for details. This approach also has very low memory requirements; since $\boldsymbol{\Sigma}_{\text{space}}$ is circulant it may be stored using only one vector. Hence storage is $\mathcal{O}(n)$ and computations (of any kind) are at most $\mathcal{O}(n \log n)$, where n is the number of nodes in the extended lattice.

5.2.1 SPDE formulation

While this decomposition is sensible, it is also very inflexible and requires stationarity for low storage requirements. Another way of pursuing good prior models with fast computations and low memory requirements is through the use of elliptic (pseudo) differential operators (Ruzhansky and Turunen (2009), part 2 is an accessible source). The theory of pseudo differential operators is closely related to Weyl transforms and short-time Fourier transforms or Gabor transforms (Feichtinger et al. (2008)) and usual spectral considerations in seismology apply. In this approach, it is the sparsity of the resulting precision matrices that makes storage and computation manageable. Recently, Lindgren et al. (2011), studied how to apply such operators in a statistical setting. They studied a Riesz-Bessel operator, $(-\triangle+\kappa)^{\alpha/2}$ and its relation to computation and Matérn covariance models (Matérn, 1960; Whittle, 1963). The main lessons are firstly, if

$$M_{\kappa,\alpha}x(s) := (\kappa^2 - \triangle)^{\alpha/2}x(s) = \mathcal{W}(s), \tag{5.8}$$

where $\mathcal{W}$ is spatial Gaussian white noise, then $x(s)$ has Matérn type covariance function, i.e.,

$$\rho(r) = \frac{\varrho^2}{\Gamma(\alpha - d/2)2^{\alpha-d/2-1}} (\kappa r)^{\alpha-d/2} K_{\alpha-d/2}(\kappa r), \tag{5.9}$$

$$\varrho^2 = \frac{\Gamma(\alpha - d/2)}{\Gamma(\alpha)(4\pi)^{d/2}\kappa^{2(\alpha-d/2)}}, \tag{5.10}$$

where K_s is the modified Bessel function of the first kind. Secondly, fast computations through finite element methods or other discretizations of the differential operator in (5.8) are available through the induced Markov properties of the discretisation matrix, $\mathbf{Q}_{\text{space}}$. That essentially means that $\mathbf{Q}_m = \mathbf{Q}_{\text{space}} \otimes \mathbf{Q}_0$ is (very) sparse and with a structure amenable to Cholesky factorisation. An alternative requirement is that we can construct the matrix vector product $\mathbf{Q}_m\mathbf{v}$ and $\det(\mathbf{Q}_m)$ relatively quickly through some iterative or direct procedure, see Simpson (2008); Aune et al. (2012a,c)

When addressing the "stationarity" of the field defined by (5.8), it is only stationary in the sense of (5.5) if it is defined on the whole of $\mathbb{R}^k$, where $k = 2, 3$ in our case – alternatively when the corresponding operator is defined on a manifold without boundary. In our case the domain on which (5.5) is defined is merely a subset, namely a square or box in $\mathbb{R}^2$ or $\mathbb{R}^3$. Hence boundary effects resulting from boundary conditions may destroy its direct interpretation in terms of this equation. It is, of course, possible to specify boundary conditions in such a way that you retain the property in (5.5), but usually there are more natural physical boundary conditions that in our opinion improves upon the specification through SPDEs compared to the model defined by covariance matrices through stationary covariance functions also in the stationary case.

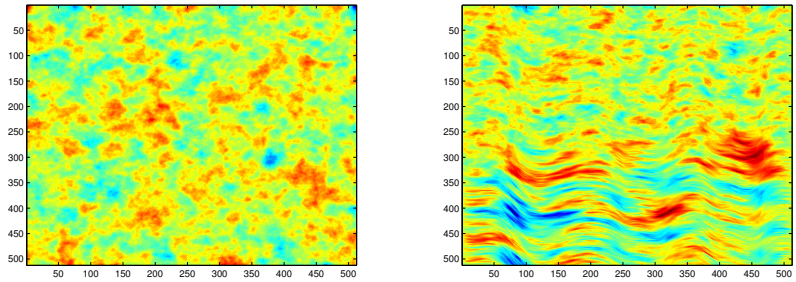
There are two properties that are desirable to have in the prior model in AVA inversion. The first is being able to have different correlation length at different points in space. If a geologist have sound reasons to believe that a layer is very inhomogeneous, it may warrant putting a lower correlation length here than in a layer that is thought to be very homogeneous with very similar properties. Facilitating this is trivial - one merely lets $\kappa^2 = \kappa^2(s)$ vary with space. The other property that is very desirable to have is anisotropy. Letting the correlation length vary with direction is very natural given that the layers are typically not flat but are deformed in a specific way. The SPDE resulting is the following variant of (5.8):

$$M_{\kappa,\alpha,s}x(s) = (\kappa^2(s) - \nabla \cdot \mathbf{H}(s)\nabla)^{\alpha/2}x(s) = \mathcal{W}(s), \tag{5.11}$$

where $\mathbf{H}$ is a 3×3 symmetric positive definite matrix defining the anisotropy angle and principal correlation length in the three directions defined by the eigenvectors of the matrix. Realisations of the stationary model and the non-stationary model is given in Figure 5.1. Here we have illustrated the "layer" flexibility mentioned above, where the top layer is isotropic, and the bottom layer is anisotropic with deformation defined by the layer.

To see how this relates to the usual approach, consider $\mathbf{Q}_0 = \mathbf{\Sigma}_0^{-1}$ and say that m_1, m_2, m_3 have equal Matérn covariance models (this includes the

Figure 5.1: *Realisations from stationary model given by* (5.8) *(left) and non-stationary model given by* (5.11) *(right). The non-stationary model has a curved interface, and the field below the interface has anisotropy directed along the curve, while above the interface it is almost isotropic.*



widely used exponential and Gaussian models), then the prior given as in (5.7) is given by the following system of stochastic differential equations:

$$(M_{\kappa,\alpha,s} \otimes \mathbf{Q}_0)\mathbf{m} = \boldsymbol{\mathcal{W}} \tag{5.12}$$

where $\boldsymbol{\mathcal{W}}$ is vector Gaussian spatial white noise. The experience in AVA-inversion is that at least one component of $\mathbf{m}$ is worse resolved than the others, with m_1 being resolved the best (see Rabben et al. (2008) or any other article treating this problem). The obvious next question then is whether or not (5.12) specifies the best way of lending strength to the least resolved parameters. If not, can we find better operators on the diagonal in (5.12), and/or replace the off-diagonals with other operators that have better properties in the inversion problem? The answer to this question is not obvious, but we investigate some alternatives and see how they perform in our inversion problem; the criterion for a better prior in the synthetic case being that $\mathbb{E}((\mathbf{m_{true}} - \mathbf{m_{est}^{new}})^2) < \mathbb{E}((\mathbf{m_{true}} - \mathbf{m_{est}^{base}})^2)$, where $\mathbf{m_{est}^{base}}$ is given by the prior model (5.7).

It is possible to replace the operator $M_{\kappa,\alpha}$ in (5.12) by more general pseudo-differential operators. Representations of such operators in terms of its

symbol are given by

$$(K_\sigma f)(x) = \int_{\mathbb{R}^d} \sigma(x,\xi)\hat{f}(\xi)e^{2\pi i x\cdot\xi}d\xi, \tag{5.13}$$

where $\hat{f}$ is the Fourier transform of f, and σ is the symbol of the operator. The symbol can be interpreted as defining the local spectrum of the operator. A deep theorem given in Rozanov (1977) states that a stationary random field is Markov (in the continuous sense) if and only if σ^{-1} is a symmetric positive polynomial. Hence Markov fields are represented by differential operators. Now, if the field in question is not Markov, it is possible to approximate σ by a rational approximation, $\sigma(x,\xi)^{-1} \approx \sigma^{-1}_{\text{rat}}(x,\xi) = \sum_{j=0}^{k} a_j(x)(2\pi i\xi)^j$. To find the a_js one can, for instance, use optimisation techniques. This is one way to do it, but we suspect that the time-frequency localisation of such an approach may be suboptimal, and discretization of the non-Markov operator may be better suited for time-frequency compressing approaches inducing approximate Markovity. We do not pursue these type of ideas here, but mention them as they are good candidates for future research.

5.3 Systems of SPDEs – generalising "$\mathbf{Q}_0$"

It is easy to write the form the generalised approach must have. First, for $i,j = 1,\ldots,3$, let

$$K_{ij} = q_{ij}(\mathbf{s})(\kappa_{ij}(\mathbf{s}) - \nabla\cdot\mathbf{H}_{ij}(\mathbf{s})\nabla)^{\alpha_{ij}/2} \tag{5.14}$$

and define the following system of SPDEs

$$\mathbf{Km}(\mathbf{s}) = \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{12} & K_{22} & K_{23} \\ K_{13} & K_{23} & K_{33} \end{pmatrix} \mathbf{m}(\mathbf{s}) = \boldsymbol{\mathcal{W}}(\mathbf{s}) \tag{5.15}$$

For $q_{ij}(\mathbf{s}) = Q^0_{ij}$ and $K_{ij} = M_{\kappa,\alpha}$ we recover the structure in the previous section with stationarity. For convenience, we call $q_{ij}(\mathbf{s})$ the blending

coefficients. In Hu et al. (2012), they study the properties of this model in the stationary case, and give the link to the multivariate Matérn fields in Gneiting et al. (2010). Any choice of K_{ij} defines a valid Gaussian Markov random field, both in the continuous sense and when discretized. In our treatment, we restrict ourselves to the case where $\alpha_{ij} = \alpha, \mathbf{H}_{ij}(\mathbf{s}) = \mathbf{H}(\mathbf{s})$ and $\kappa_{ij}(\mathbf{s}) = \kappa(\mathbf{s})$.

5.3.1 Parametrising the blending coefficients

In general, it is both hard to interpret a local precision matrix, $\mathbf{Q}_0(\mathbf{s}) = \{q_{ij}(\mathbf{s})\}_{ij}$ defining how the individual parts of the multivariate fields is related to each other at position $\mathbf{s}$, and to ensure that this matrix is positive definite. It is much more natural to work with the inverse, namely the correlation matrix defining the local correlation of the fields, $\mathbf{\Sigma}_0(\mathbf{s}) = \mathbf{Q}_0^{-1}(\mathbf{s})$. The $q_{ij}(\mathbf{s})$ is then simply given by the corresponding matrix elements. In the AVA inversion problem, information about correlation in different layers may come from geologists or geophysicists for who may know of phase changes when going from one layer to another in the different layers, or other, more complex phenomena. It may also come from well-logs that may contain information about such matters.

Suppose that $\mathbf{\Sigma}_0(\mathbf{s}) = \mathbf{\Sigma}_{0,1}$ for $\mathbf{s} \in S_1 \subset \mathbb{R}^d$ and $\mathbf{\Sigma}_{0,2}$ for $\mathbf{s} \in S_2 \subset \mathbb{R}^d$. Then we have a model that has specific correlations in one spatial region of the multivariate fields, and different correlations in another spatial region. There is obviously a transition between these two states. If the transition is discontinuous, this may be seen as a discontinuity of the correlations in the realisation of the multivariate random field, which may make sense in some situations.

In order to visualise what this means, we give realisations of the four major prior models we have discussed. In Figure 5.2, no prior information about the geometry of the subsurface can be included. In Figure 5.3, geometric information has been incorporated, but no change in the correlation between the parameters in space can be included. In Figure 5.4, an example

Figure 5.2: *Stationary model given by* (5.7)*. The field looks the same wherever we are.*

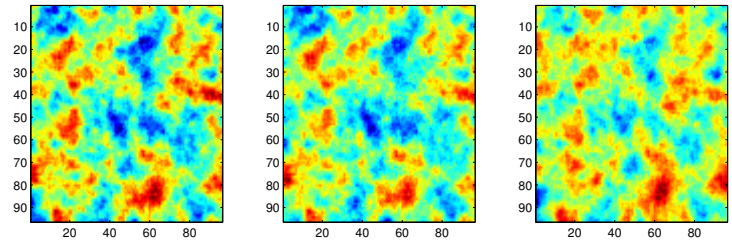
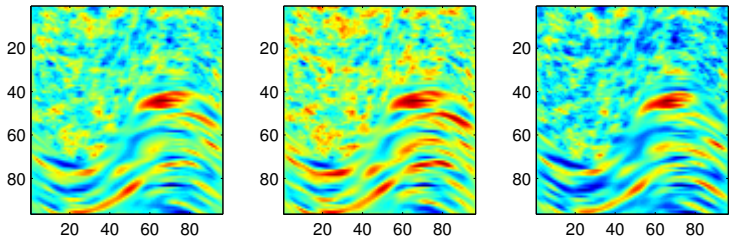


Figure 5.3: *Nonstationary model with fixed* $\mathbf{Q}_0$*. Here the bottom layer has anisotropy along the curve of the interface, and the correlation between the fields is fixed through space.*

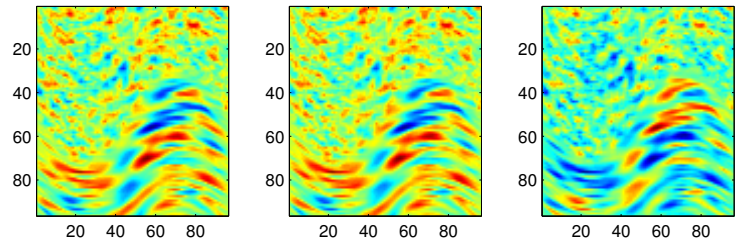


realisation from the full model is given. Pay attention to the rightmost field – here the correlation to the other two fields changes from being positive in the top layer to being negative in the bottom layer.

5.3.2 Geodesic blending

There are obviously many ways of making a smooth transition between $\mathbf{\Sigma}_{0,1}$ and $\mathbf{\Sigma}_{0,2}$, but one key consideration is that $\mathbf{\Sigma}_0(\mathbf{s})$ must remain positive definite for all $\mathbf{s}$ in some transition domain S_T. One thing is certain - it is not necessarily enough to let the off-diagonals element in $\mathbf{\Sigma}_{0,1}$ change

Figure 5.4: *Full nonstationary model with varying $q_{ij}(\mathbf{s})$ according to* (5.20)*. The bottom layer has anisotropy along the curved interface, and the correlation between the fields changes between interfaces. In particular, the rightmost field is positively correlated to the others above the interface and negatively correlated below the interface.*



linearly in $\mathbb{R}^3$ to the corresponding off-diagonal elements in $\mathbf{\Sigma}_{0,2}$.

A very natural way of making such a transition between $\mathbf{\Sigma}_{0,1}$ and $\mathbf{\Sigma}_{0,2}$ is by considering geodesics on the manifold of symmetric positive definite matrices, denoted $\mathbb{P}_d$. The natural metric on this space has a reasonable statistical interpretation, closely related to information entropy and Kullback-Leibler divergence, and an accessible account for the theory is given in Bhatia (2007). Different treatments are given in (Ohara et al., 1996; Hiai and Petz, 2009). For completeness, we give a small account of the definition and properties we need related to this manifold. This exposition is based on Hiai and Petz (2009); Bhatia (2007).

The Boltzmann entropy of the Gaussian distribution (5.6), defining an information potential, is given by

$$B(p(\mathbf{x}|\mathbf{Q}, \boldsymbol{\mu}_x)) = B(\mathbf{Q}) = \frac{1}{2} \log\det \mathbf{\Sigma} + C, \tag{5.16}$$

where C is an arbitrary constant and $\mathbf{\Sigma} = \mathbf{Q}^{-1}$ is any positive definite matrix. The Riemannian metric based on this information potential is the

Hessian

$$\mathrm{g}_{\mathbf{Q}}(\mathbf{H},\mathbf{M}) = \frac{\partial^2}{\partial s \partial t}\bigg|_{s=0,t=0} B(\mathbf{Q}+s\mathbf{H}+t\mathbf{M}) = \operatorname{tr}\mathbf{QHQK}, \tag{5.17}$$

and where $\mathbf{H},\mathbf{S} \in \mathbb{S}_d$, the tangent space of symmetric matrices, $\mathbb{S}_d = \{\mathbf{V} \in \mathbb{R}^{d\times d} | \mathbf{V} = \mathbf{V}^T\}$. This defines the line element

$$ds = \left(\operatorname{tr}\left[(\mathbf{Q}^{-1/2}d\mathbf{Q}\mathbf{Q}^{-1/2})^2\right]\right)^{1/2}. \tag{5.18}$$

Hence, if we have a curve in $\mathbb{P}_d$, i.e. $\gamma : [a,b] \to \mathbb{P}_d$, its length can be calculated as

$$L(\gamma) = \int_a^b \left(\operatorname{tr}\left[(\gamma(t)^{1/2}\gamma'(t)\gamma(t)^{1/2})^2\right]\right)^{1/2} dt \tag{5.19}$$

A nice property that follows from this is that lengths of curves are invariant under congruence transformations. That is, if $g(t) = \mathbf{X}^T\gamma(t)\mathbf{X}$, $L(\gamma) = L(g)$. The geodesic, the curve with minimal length, between two matrices, $\mathbf{A}$ and $\mathbf{B}$ can from this be deduced to be

$$\mathrm{g}_{\mathbf{A},\mathbf{B}}(t) = \mathbf{A}\#_t\mathbf{B} = \mathbf{A}^{1/2}\left(\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2}\right)^t\mathbf{A}^{1/2}, \quad t\in[0,1]. \tag{5.20}$$

Obviously, $\mathrm{g}_{\mathbf{A},\mathbf{B}}(0) = \mathbf{A}$ and $\mathrm{g}_{\mathbf{A},\mathbf{B}}(1) = \mathbf{B}$. It is this curve we use when we go from $\mathbf{A} = \mathbf{Q}_{0,1} = \mathbf{\Sigma}_{0,1}^{-1}$ to $\mathbf{B} = \mathbf{Q}_{0,2} = \mathbf{\Sigma}_{0,2}$ in different discrete structures in our prior model, and this ensures that we are within the realm of positive definite matrices in a natural way. Noting that $(\mathbf{A}\#_t\mathbf{B})^{-1} = \mathbf{A}^{-1}\#_t\mathbf{B}^{-1}$, we see that it is unproblematic to work with precision matrices rather than covariance matrices. Integrating $\mathrm{g}_{\mathbf{A},\mathbf{B}}(t)$ yields the distance between the two matrices,

$$d_{\mathbb{P}_d}(\mathbf{A},\mathbf{B}) = \int_0^1 \mathrm{g}_{\mathbf{A},\mathbf{B}}(t)dt = \left(\operatorname{tr}\left[(\log\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2})^2\right]\right)^{1/2}. \tag{5.21}$$

A potential drawback of using this strategy is that if $\mathbf{Q}_{0,1},\mathbf{Q}_{0,2}$ are correlation matrices, and what you want is a continuum of correlation matrices,

$\mathrm{g}_{\mathbf{Q}_{0,1},\mathbf{Q}_{0,2}}(t)$ are not correlation matrices for $t \in (0,1)$. It is possible to correct for this by using geodesics on the sub-manifold of correlation matrices in $\mathbb{P}_d$. In practice, however, $\mathrm{g}_{\mathbf{Q}_{0,1},\mathbf{Q}_{0,2}}(t)$ seem to be very close to being correlation matrices in most cases. We do not have any counterexamples.

Fuzzy interfaces

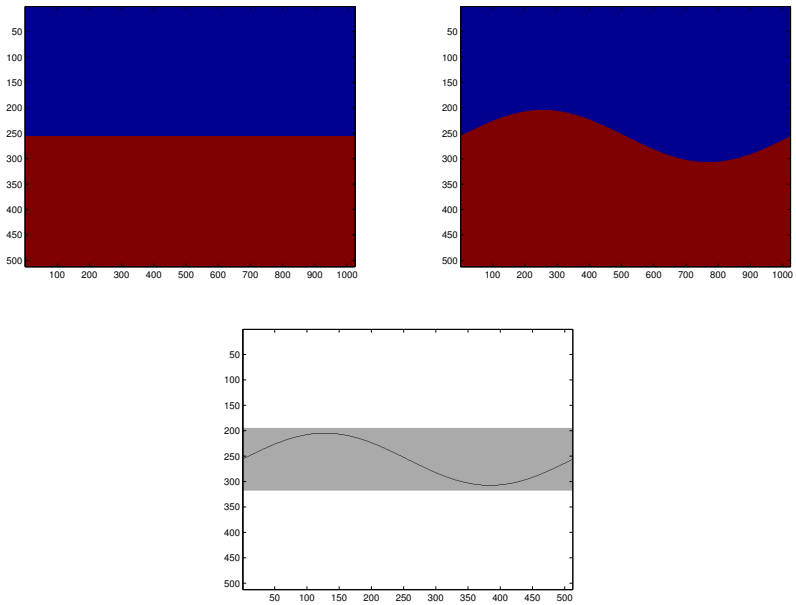
In some situations, we may actually have a hard interface in our multivariate field, but even in this situation, experts may place the interface incorrectly, which may lead to imprecise interpretation of the field. The geodesic blending strategy discussed in the previous section gives us a way to handle this situation in a specific way: the blending range may serve as quantifying the uncertainty or fuzziness of this interface. This range may then be estimated based on realisations of the field, possibly requiring a strong prior for identifiability.

To illustrate this, suppose that an expert says that the interface is as in the upper left part of Figure 5.5, while the real interface is given on the right. The bottom illustration in Figure 5.5 shows what the geodesic range should be in this case (grey area) – it should cover the true interface properly, showing that there actually is a fair amount of uncertainty in the placement of the interface. In Section 5.4, we investigate whether this range may be estimated purely from data or if a strong prior on the range is needed. It is, of course, possible to combine this idea with procedures for actually estimating the interface, but, as always, this increases the complexity of the model that is to be estimated. Additionally, the blend range may easily confound with potential parameters needed to estimate the actual location of the interface.

5.3.3 Modified parametrisations

There are many ways to modify the parametrisation described above to reduce parametrisation demand or incorporate different flexibility. A possible

Figure 5.5: *Illustration of blend range. Guessed interface (left), true interface (right), blend range illustrated in grey (bottom)*



way to reduce the parametrisation demand further is to do the modeling in the Cholesky domain. This is a simplification, but it is one we believe should increase interpretability and possibly estimation properties. To motivate this approach, consider the following: Suppose that the Cholesky factorisation of $\mathbf{Q}_0$ is given by $\mathbf{Q}_0 = \mathbf{L}_0\mathbf{L}_0^T$, and that $\mathbf{Q}_{\text{space}} = \mathbf{B}_1^s(\mathbf{Q}_2^s)^*$, for some matrices $\mathbf{B}_1^s, \mathbf{B}_2^s$. Generating the matrices $\{\mathbf{B}_i^s\}_{1,2}$ can for instance be done by using $\alpha_s = \alpha/2$ in (5.8) and discretizing this operator, but there exist many other factorisations that may behave in better way for the problem at hand. By a Kronecker product identity, $\mathbf{Q}_{\text{space}} \otimes \mathbf{Q}_0 = (\mathbf{B}_1^s \otimes \mathbf{L}_0)((\mathbf{B}_2^s)^* \otimes \mathbf{L}_0^T)$. The intuition stemming from this identity carries over to the more general case in a natural way: Let $l_{ij}(\mathbf{s})$ be entry i, j of the Cholesky factor of the matrix $\{q_{ij}(\mathbf{s})\}_{ij}$ locally, and define locally operators that will correspond to some square root of its original form in (5.14). It is possible to define the operators in such a way that we get back (5.15), but this is of minor concern in practice as long as we get the interpretability we want. This is reminiscent to the triangular approach mentioned in Hu et al. (2012).

5.4 Parameter estimation and conditional expectation

In order to show that our proposed model is useful with confidence in the realm of seismic AVA inversion, we must show that estimation of hyper-parameters in the prior model is feasible and that the conditional expectation, $\mathbb{E}(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta})$ (here $\boldsymbol{\theta}$ denotes model parameters), is better than in the simpler model. A natural way to see if the hyper-parameters are identifiable is to simulate from the prior fields and do maximum likelihood estimates on these. If this works well, one may go one level higher and assume noisy observations of the form

$$\mathbf{y} = \mathbf{W}\mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}, \tag{5.22}$$

where $\mathbf{W}$ denotes a convolution matrix defined by the wavelet, and $\mathbf{A}$ denotes the reflectivity matrix, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It is also here possible to do maximum likelihood estimates. For more information on estimating

this type of model, consult your favourite treatise that discusses latent (Gaussian) models for, e.g. Rasmussen and Wiliams (2006); Rue and Held (2005); Cressie and Wikle (2011). In treating this estimation problem, we use the simpler anisotropic model where the correlation changes from positive to negative at an interface defined by a straight line. It is, of course, possible to estimate the geometry as well, but this is beyond the scope of this text.

When estimating the q_{ij}, supposing it changes between layers, we must impose constraints to enforce the interpretability we want – namely that of its local inverse being the correlation matrix of the multivariate field at that point. Now, the matrix consisting of ones on its diagonal with three free parameters off its diagonal uniquely specifies these constraints through its eigenvalues: they must all be greater than zero. Hence we have three constraints, depending only on the off-diagonal elements of the local correlation matrix. The same type of restriction would apply if we were to use general local 3×3 covariances instead. In that scenario, however, the three constraints would depend on six parameters instead of three. In this section, we will denote the different models as follows

1. Model 1 is the simple stationary $\mathbf{Q}_0 \otimes \mathbf{Q}_{\text{space}}$ as in (5.7)
2. Model 2 is stationary in space using the extended $q_{ij}(\mathbf{s})$ parametrisation as in Section 5.3.1, equation (5.15), using interpretability constraints
3. Model 3 is nonstationary in space and using the extended $q_{ij}(\mathbf{s})$ parametrisation as in Section 5.3.1, equation (5.15). Additionally, we use a blending of two correlation matrices at the interface, so that the correlation change is not discontinuous.

5.4.1 Identifiability

We show that the parameters in $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1$ are identifiable through simulation. To do this, we simulate from many multivariate fields and estimate the parameters by maximum likelihood. If the estimated maximum likelihood

density – which is estimated from several realisations – is unimodal, the parameters are identifiable. Suppose that $\mathbf{\Sigma}_0, \mathbf{\Sigma}_1$ are given by

$$\mathbf{\Sigma}_0 = \begin{pmatrix} 1 & 0.7 & 0.2 \\ & 1 & 0.4 \\ & & 1 \end{pmatrix}, \quad \mathbf{\Sigma}_1 = \begin{pmatrix} 1 & 0.7 & -0.9 \\ & 1 & -0.85 \\ & & 1 \end{pmatrix}, \tag{5.23}$$

using $\kappa^2 = 0.1$, and $\tau^2\mathbf{Q}$, with $\tau^2 = 50$. Using 200 realisations from the field, we get maximum likelihood density estimates for the parameters – these are illustrated in Figure 5.6. For obtaining the parameters, we used a quasi-Newton method with initial correlation parameters being zero. Judging from this figure, since all density estimates are unimodal, all parameters seem to be identifiable.

In the case where we have noisy observations, we use profile likelihood to estimate the noise level, σ^2, and a quasi-Newton method to estimate $\lambda^2 = \tau^2\sigma^2$, κ^2 and the correlation parameters. In this case we used the correlation matrices

$$\mathbf{\Sigma}_0 = \begin{pmatrix} 1 & 0.7 & 0.6 \\ & 1 & 0.95 \\ & & 1 \end{pmatrix}, \quad \mathbf{\Sigma}_1 = \begin{pmatrix} 1 & 0.75 & -0.9 \\ & 1 & -0.85 \\ & & 1 \end{pmatrix}. \tag{5.24}$$

In Figure 5.7 the corresponding estimates for a hidden field is given. Of course, it is much more difficult in this situation, which is reflected through the broad distributional tails in the figure. Overall, however, the estimates seem to recover the true values quite well. One odd observation is the bimodality of $\Sigma_0(2,3)$. We believe it may come from observing rather small fields, from a 64×64-grid, and that it may go away for larger ones. The values over one on the left part of the figure are artefacts coming from using a kernel smoother for estimating the density.

5.4.2 Conditional expectation

The real test on whether it is wise or not to use this advanced parametrisation of the model is essentially the reconstruction problem: based on noisy

Figure 5.6: *Maximum likelihood density estimates for correlation parameters, κ^2 and τ^2 using direct observations of 200 fields*

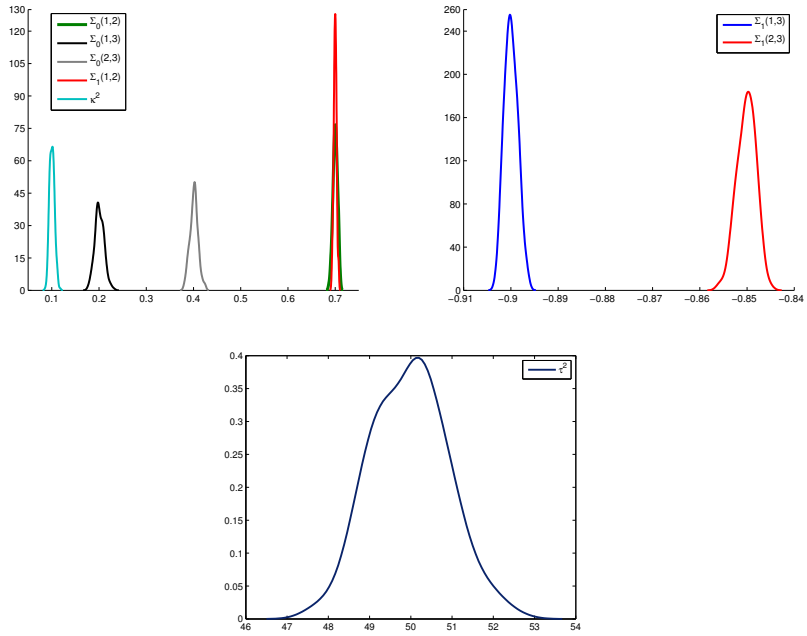
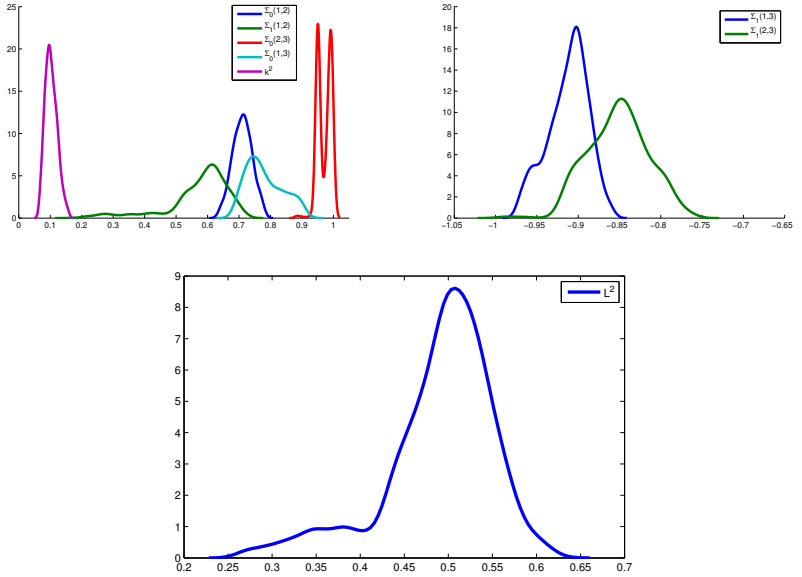


Figure 5.7: *Maximum likelihood density estimates for correlation parameters, κ^2 and τ^2 using indirect observations of 600 fields*



observations, are we able to reconstruct the original fields with higher fidelity?

In the following subsections, we give several reconstructions examples, and we use two observations schemes. The first one is based on identity observations with i.i.d. noise, and the second is based on the AVA model, giving the observation matrix **WA** followed by i.i.d. noise.

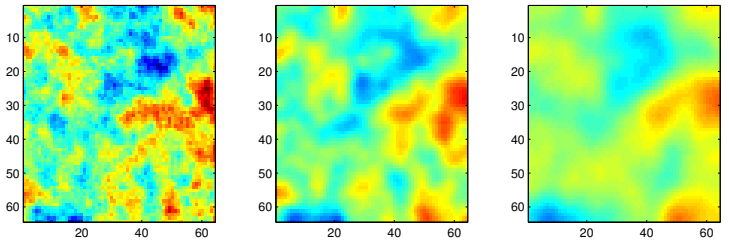
Identity observations

First, we present a reconstruction example where the observation matrix is the identity, followed by iid. noise, and with two signal-to-noise ratios. One with $\lambda^2 = 50$ and one with $\lambda^2 = 0.5$. For these two models, we use the following true correlation matrices

$$\boldsymbol{\Sigma}_0 = \begin{pmatrix} 1 & 0.99 & 0.99 \\ & 1 & 0.99 \\ & & 1 \end{pmatrix}, \quad \boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & \text{-}0.99 & \text{-}0.99 \\ & 1 & 0.99 \\ & & 1 \end{pmatrix}. \tag{5.25}$$

In Figure 5.8, we illustrate reconstruction of the first of the three fields with signal-to-noise ratio 1/50, using a flat interface and identity observations. I.e. the true field is generated by Model 2, followed by i.i.d. noise. A priori we believe one of the worst situations for estimating Model 1, as correlations change very much from structure to structure and the noise level is very high. The likelihood function in the situation with high noise levels appears very flat, requiring high accuracy and many iterations in the optimisation scheme to give consistent estimates. For $\lambda^2 = 50$, $\|\mathbb{E}^{\mathrm{M2}}(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}) - \mathbf{x}\|_2 / \|\mathbf{x}\|_2 = 0.526$ for Model 2 and $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}) - \mathbf{x}\|_2 / \|\mathbf{x}\|_2 = 0.674$ for Model 1. The first field is chosen, as for the correlation matrices defined for this, the first field is the one with changing correlation between interfaces, relative to the others. The main effect we see in this comparison is that the level of the reconstructed field using the Model 1 does not completely reach up to the true levels – we believe this can be attributed to a flattening effect arising from the sum of the two correlations in the different layers being zero.

Figure 5.8: *Kriging for identity observations with signal-to-noise ratio* 1/50. *True parameters (left), kriging using true Model 2 (center), using Model 1 (right)*



In the second comparison we generate fields and observations from Model 2, with $\lambda^2 = 0.5$. Here a different effect is more prominent – we see that the reconstructed field on the right in Figure 5.9, i.e. the reconstruction based on using Model 1, is smoother and does not exhibit as much of the jaggedy effect of the true surface compared to the field in the middle. A consistent property when estimating the Model 1 is that κ^2 seems to be underestimated, leading to a larger range and hence smoother reconstruction. One may think that this smoothing effect of the field on the right in Figure 5.9, but for comparison, we also include reconstructions of the second field, depicted in Figure 5.10. Here the mentioned smoothing effect is not as present as in Figure 5.9. Hence, we believe that this is an effect induced by the changing correlations. In Figure 5.9, $\|\mathbb{E}^{\mathrm{M2}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.179$ for the middle reconstruction, and $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.269$ for the rightmost one, while in Figure 5.10, $\|\mathbb{E}^{\mathrm{M2}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.168$ and $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.195$.

AVA observations

While the results using the identity observations are convincing in the extended models' favour, we also need to investigate the effects where the observation matrix is the seismic AVA model. In this case, the true fields are generated by Model 2, and the observations are linear combinations

Figure 5.9: *Kriging for identity observations with signal-to-noise ratio* 1/0.5*, field 1. True parameters (left), Kriging using Model 2 (center), using Model 1 (right)*

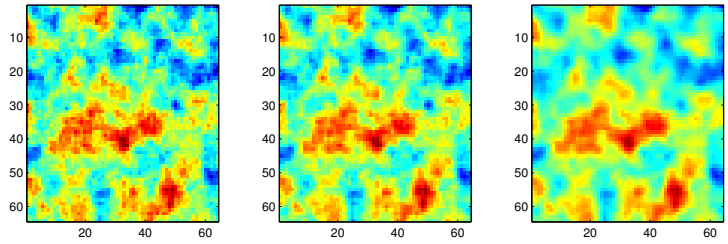


Figure 5.10: *Kriging for identity observations with signal-to-noise ration* 1/0.5*, field 2. True parameters (left), using true model (center), using model defined by* (5.12) *(right)*

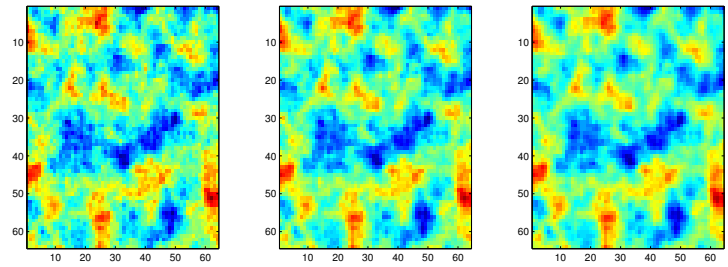
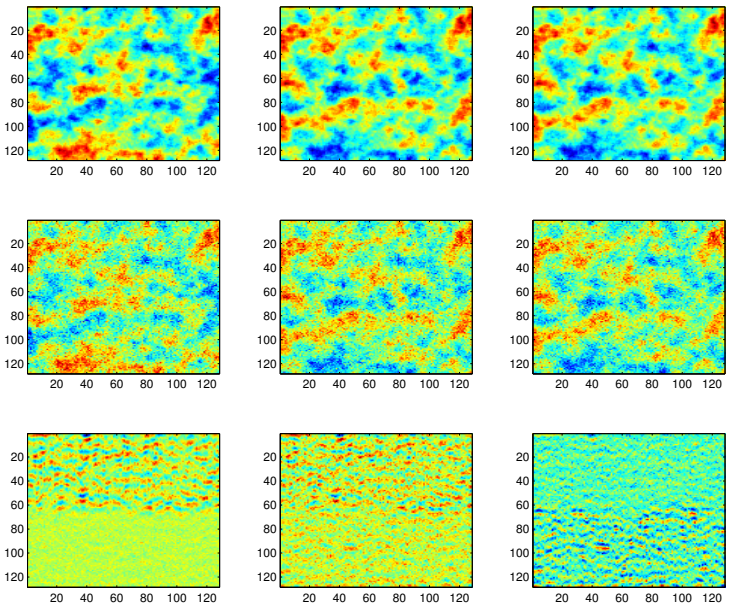


Figure 5.11: *Observations using identity observations (middle) and the seismic AVA model (bottom)*

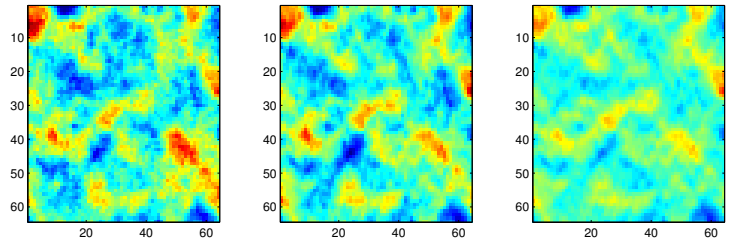


of the various fields at each space location followed by a convolution with a smooth wavelet and i.i.d noise. We use $\lambda^2 = 0.5$ and $\lambda^2 = 20$ in these examples.

In Figure 5.11, we can see the observations that are generated by this process. A key feature in the observations is that there occurs some cancellation, resulting from the fact that they are linear combinations of the underlying fields. This results in varying signal-to-noise ratios depending on the varying correlations.

Reconstructing the original multivariate field using the AVA observation

Figure 5.12: *Reconstructed field using the AVA model with signal-to-noise ratio* 1/0.5. *True parameters (left), kriging using Model 2 (centre), using Model 1 (right).*



scheme is more difficult than for using the identity observation matrix. The aforementioned cancellation effect is a major contributor to this. Additionally, there does not seem to be a straightforward way of interpreting the estimated correlation parameters coming from Model 1. In Figure 5.12, we illustrate the true parameters on the left, with reconstruction using Model 2 in the middle and Model 1 on the right, using a signal-to-noise ratio of 1/0.5. The effects we see are reminiscent of those using identity observations, but the smoothing effect is not present here. In this case $\|\mathbb{E}^{\mathrm{M2}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.461$, while $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\| = 0.728$. Reconstruction using the same model, with a signal-to-noise ratio 1/20 is depicted in Figure 5.13. No smoothing effect relative to Model 2 is observed here, but predictions are worse using Model 1, having $\|\mathbb{E}^{\mathrm{M2}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.762$, while $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\| = 0.863$.

Identity observations and non-stationarity

Until this point, we have only studied the effects coming from changing correlations between interfaces. The model we have described is much richer than that, providing a flexible way of specifying anisotropy that moves along geometry of the subsurface. In this situation, we expect the results to be even more convincing, and we provide one example to cover this situation as well. In this case, we generate the true field by using Model 3, and we have identity

Figure 5.13: *Reconstructed field using the AVA model with signal-to-noise ratio* 1/20. *True parameters (left), kriging using Model 2 (centre), using Model 1 (right).*

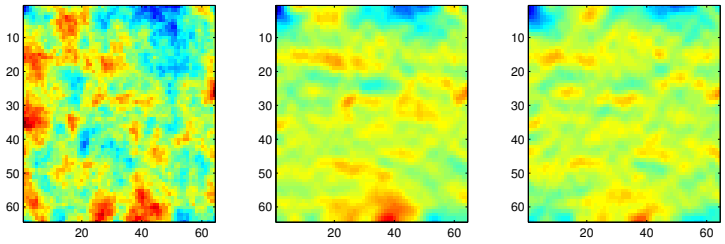
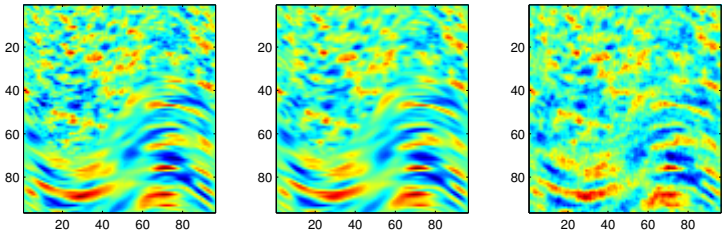


Figure 5.14: *Kriging for identity observations with signal-to-noise ration* 1/0.2, *field 1. True parameters (left), kriging using Model 3 (center), Model 1 (right)*



observations followed by i.i.d. noise. The realisations of the true fields are then similar to the one in Figure 5.4, and we estimate both the simple and complex model for thereafter giving a reconstruction of the latent field. In Figure 5.14, we see the reconstructions using Model 3 (center) and Model 1 (right), and the most prominent effect we see is the smoothness differences in the bottom layer. Reconstruction using Model 1 is rugged and does not capture the anisotropy of the layer at all, contrasting the reconstruction using Model 3. On the top layer, on the other hand, the reconstructions are more comparable. The relative errors are $\|\mathbb{E}^{\mathrm{M3}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.280$ (center) and $\|\mathbb{E}^{\mathrm{M1}}(\mathbf{x}|\mathbf{y},\boldsymbol{\theta}) - \mathbf{x}\|_2/\|\mathbf{x}\|_2 = 0.382$ (right) for the reconstructions – i.e. predictions are about 37% better using the true model.

Reconstruction – final remarks

It is also important to note that if we simulate from the simple model, the parameters here are recovered well by using the parametrisation in Section 5.3.1. This means that the correlations estimated by the simple model are close to the ones estimated by the more complex model. This, of course, adds to the usefulness of the model in situations where we do not know in advance that the correlation changes between interfaces. The uncertainty, however, is greater, leading to more disparate estimates of the correlations than when using the simple parametrisation.

5.4.3 Estimating the blend range for fuzzy interfaces

In this section, we will treat all parameter except the blend range as fixed. The model we will treat is one where the true interface is given as a sine function, and what we guess is a flat interface. This is exactly the model which is depicted in Figure 5.5. In our example, we use Model 2 for constructing the true field, followed by identity observations and i.i.d. noise.

Before actually doing maximum likelihood estimation, we visualise heuristically why it may make sense. In Figure 5.15, we see a sample of the true model at the top, the true sample minus the guessed model in the middle, and the true sample minus the blend model with optimal range at the bottom. The norm of the bottom figure is less than that of the middle one.

Maximum likelihood estimates for the range is given in Figure 5.16, where the left figure is the range estimates when the guessed interface is a line and the true line is a full-period sine with a maximum amplitude of 23 and the right is a half-period sine with maximum amplitude 23. These estimates are good in the sense that the range covers the true model as in Figure 5.5.

A comparison of predictions using the guessed interface with no blend and the one with optimal blend is given in Figure 5.17. Here we see that the

Figure 5.15: *Sample from true model (top), sample from true minus guessed model (middle), sample from true model minus blend model with optimal range (bottom).*

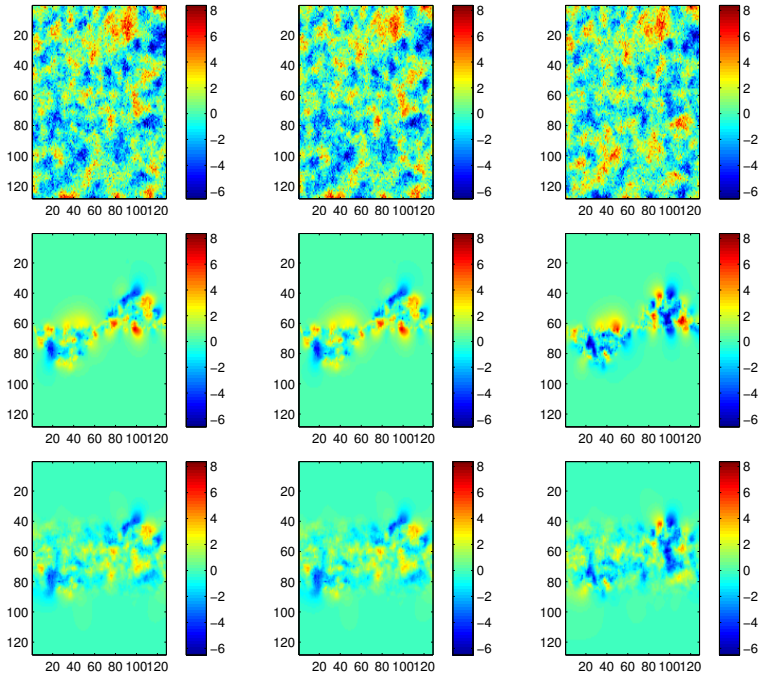
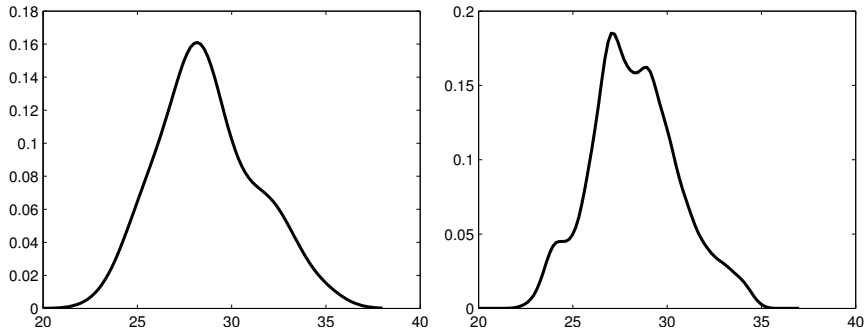


Figure 5.16: *Estimated blend range using maximum likelihood for 200 samples. Sine-interface with full period (left), sine-interface with half period (right).*

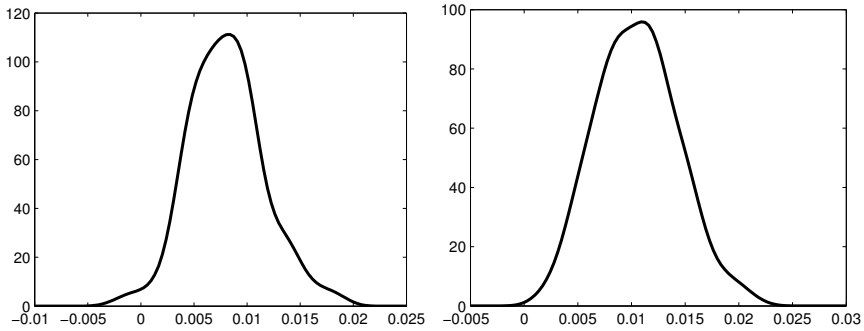


predictions using optimal blend range are only marginally better than using the interface with no blend for both interface structures. For the blend range, however, better prediction is not the goal. The goal here is to get an idea about how uncertain we are about the interface location, and better predictions comes as an additional boon, even if the improvement is marginal.

5.5 Conclusions and future work

In this chapter we have showed three things: First, how it is possible to incorporate information about the geometry of the problem flexibly. Secondly, how to facilitate changing covariances between elastic parameters depending on position. Lastly, we have introduced a novel way of specifying uncertainty related to the position of an interface using the concept of geodesic blending based on local correlation of the multivariate field. The first hinges on using SPDEs in order to specify local properties of the fields, and the second on how systems of SPDEs interrelate depending on position. The geodesic blending approach is based on the smooth manifold structure of the set of positive definite matrices. The ideas presented here are not

Figure 5.17: *Performance gain using the blended interface over the guessed interface with sine-interface with full period (left) and sine-interface with half period (right). The x-axis is defined by* $(\|\mathbf{x}_{noblend}-\mathbf{x}_{true}\|_2-\|\mathbf{x}_{optblend}-\mathbf{x}_{true}\|_2)/\|\mathbf{x}_{true}\|_2$*, and the y-axis through the corresponding density estimate. This is generated from the same 200 samples as in the previous figure*



limited to the relatively simple models described here – rather, they may be used in any spatial inversion problem with a natural geometry where soft constraints based on expert opinion may be used.

5.6 Addendum: Finite difference disretization – the details

This appendix is devoted to the finite difference scheme we used for discretizing the elliptic operator in (5.11). We employ a changed notation in this appendix for convenience, replacing $\mathbf{H}$ with $\mathbf{A}$, and we hope that it is

transparent for readers. For a 2-dimensional field with $\alpha = 1$, we have

$$\begin{aligned}
&\nabla \cdot \begin{pmatrix} a_{11}(x,y) & a_{12}(x,y) \\ a_{21}(x,y) & a_{22}(x,y) \end{pmatrix} \begin{pmatrix} u_x(x,y) \\ u_y(x,y) \end{pmatrix} + \kappa(x,y)u(x,y) \\
=&\nabla \cdot \begin{pmatrix} a_{11}(x,y)u_x(x,y) + a_{1,2}(x,y)u_y(x,y) \\ a_{21}(x,y)u_x(x,y) + a_{22}(x,y)u_y(x,y) \end{pmatrix} + \kappa(x,y)u(x,y) \\
=&\partial_x(a_{11}u_x + a_{12}u_y) + \partial_y(a_{21}u_x + a_{22}u_y) + \kappa u \\
=&a_{11}^x u_x + a_{11}u_{xx} + a_{12}^x u_y + a_{12}u_{yx} + a_{21}^y u_x + a_{21}u_{xy} + a_{22}^y u_y + a_{22}u_{yy} \\
=&\mathrm{diag}(A)\nabla \cdot \nabla u + (a_{12} + a_{21})u_{xy} + a_{11}^x u_x + a_{12}^x u_y + a_{22}^y u_y + a_{21}^y u_x \quad (5.26)
\end{aligned}$$

where $a_{ij}^v, v = x, y$ denotes differentiation wrt. x or y of the i, j element of $\mathbf{A}$, depending implicitly on the position. To discretize (5.26), we employ a finite difference scheme. We define the following finite difference operators

$$\begin{aligned}
\delta_x u &= \frac{1}{h}(u_{i+1}^j - u_i^j) \\
\delta_{\hat{x}} &= \frac{1}{h}(u_i^j - u_{i-1}^j),
\end{aligned}$$

where i, j are positions on the grid, with i denoting the x-direction and j denoting the y-direction. Now, we define the following operators

$$\begin{aligned}
\Lambda_{xx}u = \delta_x\left(\alpha_{11}\delta_{\hat{x}}u\right) &= \delta_x\left(\frac{1}{h}\alpha_{11}\left(u_i^j - u_{i-1}^j\right)\right) \\
&= \frac{1}{h^2}\left(\alpha_{11}^{i+1,j}\left(u_{i+1}^j - u_i^j\right) - \alpha_{11}^{i,j}\left(u_i^j - u_{i-1}^j\right)\right), \qquad (5.27)
\end{aligned}$$

where

$$\begin{aligned}
\alpha_{11}^{i,j} &= \frac{1}{2}\left(a_{11}^{i,j} + a_{11}^{i-1,j}\right) \\
\alpha_{22}^{i,j} &= \frac{1}{2}\left(a_{22}^{i,j} + a_{22}^{i,j-1}\right).
\end{aligned}$$

A equivalent expression holds for $\Lambda_{yy}u$. We define $\alpha_{kk}^{1,1} = a_{11}^{1,1}, k = 1, 2$. For the mixed operators we have

$$\Lambda_{xy}^{+}u = \frac{1}{2}\left(\delta_x\left(a_{12}\delta_y u\right) + \delta_{\hat{x}}\left(a_{12}\delta_{\hat{y}}u\right)\right) \qquad (5.28)$$

and we have

$$\begin{aligned}
\delta_x\left(a_{12}\delta_y u\right) &= \frac{1}{h}\delta_x\left(a_{12}u_i^{j+1} - u_i^j\right) \\
&= \frac{1}{h^2}\left(a_{12}^{i+1,j}(u_{i+1}^{j+1} - u_{i+1}^j) - a_{12}^{i,j}(u_i^{j+1} - u_i^j)\right) \\
\delta_{\hat{x}}\left(a_{12}\delta_{\hat{y}} u\right) &= \frac{1}{h}\delta_{\hat{x}}\left(a_{12}(u_i^j - u_i^{j-1})\right) \\
&= \frac{1}{h^2}\left(a_{12}^{i,j}(u_i^j - u_i^{j-1}) - a_{12}^{i-1,j}(u_{i-1}^j - u_{i-1}^{j-1})\right).
\end{aligned}$$

Hence

$$\begin{aligned}
\Lambda_{xy}^+ u = &\frac{1}{2h^2}\Big(\left(a_{12}^{i+1,j}(u_{i+1}^{j+1} - u_{i+1}^j) - a_{12}^{i,j}(u_i^{j+1} - u_i^j)\right) \\
&+ \left(a_{12}^{i,j}(u_i^j - u_i^{j-1}) - a_{12}^{i-1,j}(u_{i-1}^j - u_{i-1}^{j-1})\right)\Big)
\end{aligned} \tag{5.29}$$

For Λ_{yx}^+ we reverse the order of the difference operators:

$$\begin{aligned}
\Lambda_{yx}^+ u &= \frac{1}{2}(\delta_y(a_{12}\delta_x u) + \delta_{\hat{y}}(a_{12}\delta_{\hat{x}})) \\
&= \frac{1}{2h^2}\Big(\left(a_{12}^{i,j+1}(u_{i+1}^{j+1} - u_i^{j+1}) - a_{12}^{i,j}(u_{i+1}^j - u_i^j)\right) \\
&\quad + \left(a_{12}^{i,j}(u_i^j - u_{i-1}^j) - a_{12}^{i,j-1}(u_i^{j-1} - u_{i-1}^{j-1})\right)\Big)
\end{aligned}$$

And the complete discretisation is

$$(\Lambda_{xx} + \Lambda_{xy}^+ + \Lambda_{yx}^+ + \Lambda_{yy})u = f(u, W) \tag{5.30}$$

In Samarskii et al. (2002), it is proved that this scheme is convergent. If we assume that $\mathbf{A}$ does not vary in space, we can simplify the scheme;

$$\begin{aligned}
\widehat{\Lambda_{xx}}u &= \frac{1}{h^2}(a_{11}(u_{i+1}^j - u_i^j) - a_{11}(u_i^j - u_{i-1}^j)) \\
&= \frac{1}{h^2}a_{11}(u_{i+1}^j - 2u_i^j + u_{i-1}^j) \\
\widehat{\Lambda_{yy}}u &= \frac{1}{h^2}a_{22}(u_i^{j+1} - 2u_i^j + u_i^{j-1}) \\
\widehat{\Lambda_{xy}^+}u &= \frac{1}{2h^2}\Big(\Big(a_{12}(u_{i+1}^{j+1} - u_{i+1}^j) - a_{12}(u_i^{j+1} - u_i^j)\Big) \\
&\quad + \Big(a_{12}(u_i^j - u_i^{j-1}) - a_{12}(u_{i-1}^j - u_{i-1}^{j-1})\Big)\Big) \\
&= \frac{a_{12}}{2h^2}\left(2u_i^j + u_{i+1}^{j+1} + u_{i-1}^{j-1} - u_{i+1}^j - u_i^{j+1} - u_i^{j-1} - u_{i-1}^j\right) \\
\widehat{\Lambda_{yx}^+}u &= \frac{1}{2h^2}\Big(\Big(a_{12}(u_{i+1}^{j+1} - u_i^{j+1}) - a_{12}(u_{i+1}^j - u_i^j)\Big) \\
&\quad + \Big(a_{12}(u_i^j - u_{i-1}^j) - a_{12}(u_i^{j-1} - u_{i-1}^{j-1})\Big)\Big) \\
&= \frac{a_{12}}{2h^2}\left(2u_i^j + u_{i+1}^{j+1} + u_{i-1}^{j-1} - u_{i+1}^j - u_i^{j+1} - u_i^{j-1} - u_{i-1}^j\right) \\
\left(\widehat{\Lambda_{xy}^+} + \widehat{\Lambda_{yx}^+}\right)u &= \frac{a_{12}}{h^2}\left(2u_i^j + u_{i+1}^{j+1} + u_{i-1}^{j-1} - u_{i+1}^j - u_i^{j+1} - u_i^{j-1} - u_{i-1}^j\right)
\end{aligned}$$

This corresponds to the following stencil

$$S = -\frac{1}{h^2}\begin{pmatrix} a_{12} & -a_{22} - a_{12} & 0 \\ -a_{11} - a_{12} & 2(a_{11} + a_{22} + a_{12}) & -a_{11} - a_{12} \\ 0 & -a_{22} - a_{12} & a_{12} \end{pmatrix} \tag{5.31}$$

Bibliography

Adler, R. and Taylor, J. (2007). *Random fields and geometry*, volume 115. Springer Verlag.

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

Aki, K. and Richards, P. G. (1980). *Quantitative Seismology*. WH Freeman and Co.

Al-Mohy, A. and Higham, N. (2011). Computing the Action of the Matrix Exponential, with an Application to Exponential Integrators. Technical report, University of Manchester.

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.

Aune, E., Eidsvik, J., and Pokern, Y. (2012a). Iterative Numerical Methods for Sampling from High Dimensional Gaussian Distributions. *Statistics and Computing*, Accepted.

Aune, E., Eidsvik, J., and Ursin, B. (2012b). Three-dimensional non-stationary and non-linear ava inversion. *Geophysical Journal International. Submitted.*

Aune, E. and Simpson, D. P. (2012). The use of systems of stochastic PDEs as priors for multivariate models with discrete structures. *Scandinavian Journal of Statistics. Submitted.*

Aune, E., Simpson, D. P., and Eidsvik, J. (2012c). Parameter estimation for high dimensional Gaussian distributions. *Statistics and Computing. Revised.*

Bai, Z., Fahey, G., and Golub, G. (1996). Some large-scale matrix computation problems. *Journal of Computational and Applied Mathematics*, 74(1):71–89.

Bai, Z. and Golub, G. (1997). Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices. *Annals of Numerical Mathematics*, 4:29–38.

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.

Bekas, C., Kokiopoulou, E., and Saad, Y. (2007). An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12):1214–1229.

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.

Bellavia, S., De Simone, V., di Serafino, D., and Morini, B. (2011). Efficient preconditioner updates for shifted linear systems. *SIAM Journal on Scientific Computing*, 33(4):1785.

Benzi, M. and Bertaccini, D. (2003). Approximate inverse preconditioning for shifted linear systems. *BIT Numerical Mathematics*, 43(2):231–244.

Benzi, M. and Golub, G. (1999). Bounds for the entries of matrix functions with applications to preconditioning. *BIT Numerical Mathematics*, 39(3):417–438.

Benzi, M., Meyer, C., Tuma, M., et al. (1996). A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149.

Benzi, M. and Razouk, N. (2007). Decay bounds and $\mathcal{O}(n)$ algorithms for approximating functions of sparse matrices. *Electronic Transactions on Numerical Analysis*, 28:16–39.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B*, 36:192–236.

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.

Bhatia, R. (2007). *Positive definite matrices*. Princeton Univ Press.

Björck, Å. (1996). *Numerical methods for least squares problems*. Number 51. Society for Industrial Mathematics.

Bloch, J. and Heybrock, S. (2009). A nested Krylov subspace method for the overlap operator. *Arxiv preprint arXiv:0910.2918*.

Bloch, J. C., Breu, T., , Frommer, A., Heybrock, S., Schfer, K., and Wettig, T. (2009). Krylov subspace methods and the sign function: multishifts and deflation in the non-Hermitian case. *arXiv:0910.2927v1*.

Bogachev, V. (1998). *Gaussian measures*. American Mathematical Society.

Bolin, D. (2012). Spatial Matérn fields driven by non-Gaussian noise. *Arxiv preprint arXiv:1206.0622*.

Bolin, D. and Lindgren, F. (2011). Spatial models generated by nested stochastic partial differential equations, with an application to global ozone mapping. *The Annals of Applied Statistics*, 5(1):523–550.

Bratley, P. and Fox, B. L. (1988). Algorithm 659 Implementing Sobol's Quasirandom Sequence Generator. *ACM Transactions on Mathematical Software*, 14:88–100.

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.

Candès, E., Demanet, L., Donoho, D., and Ying, L. (2006). Fast discrete curvelet transforms. *Multiscale Modeling Simulation*, 5(3):861– 899.

Chan, R., Chan, T., and Wong, C. (1999). Cosine transform based preconditioners for total variation deblurring. *Image Processing, IEEE Transactions on*, 8(10):1472–1478.

Chan, T., Tang, W., and Wan, W. (1997). Wavelet sparse approximate inverse preconditioners. *BIT Numerical Mathematics*, 37(3):644–660.

Chen, Y., Davis, T., Hager, W., and Rajamanickam, S. (2008). Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22.

Choi, S., Paige, C., and Saunders, M. (2010). MINRES-QLP: a Krylov subspace method for indefinite or singular symmetric systems. *Arxiv preprint arXiv:1003.4042*.

Choi, S.-C. (2006). *Iterative methods for singular linear equations and least-squares problems*. PhD thesis, Stanford University.

Cressie, N. (1993). *Statistics for Spatial Data*. Wiley.

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

Cressie, N. and Wikle, C. (2011). *Statistics for spatio-temporal data*, volume 465. Wiley.

Culberson, J. (1992). Iterated greedy graph coloring and the difficulty landscape. Technical report, University of Alberta.

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.

Davis, T. and Hager, W. (1999). Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(3):606–627.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, pages 1–38.

Driscoll, A. (2009). The Schwarz Christoffel Toolbox available at `http://www.math.udel.edu/~driscoll/software/SC`. Electronic.

Eidsvik, J., Shaby, B. A., Reich, B. J., Wheeler, M., and Niemi, J. (2011). Estimation and prediction in spatial models with block composite likelihoods using parallel computing. Technical report, NTNU.

Estrada, E., Hatano, N., and Benzi, M. (2011). The physics of communicability in complex networks. *Arxiv preprint arXiv:1109.2950*.

Faber, K. and Kowalski, B. (1997). Propagation of measurement errors for the validation of predictions obtained by principal component regression and partial least squares. *Journal of chemometrics*, 11(3):181–238.

Farquharson, C. and Oldenburg, D. (1998). Non-linear inversion using general measures of data misfit and model structure. *Geophysical journal international*, 134(1):213–227.

Feichtinger, H., Helffer, B., Lamoureux, M., Lerner, N., Morel, J., Rodino, L., Takens, F., Teissier, B., Toft, J., and Wong, M. (2008). *Pseudo-Differential Operators: Quantization and Signals*. Springer-Verlag Berlin Heidelberg.

Frommer, A. (2003). BiCGStab (l) for families of shifted linear systems. *Computing*, 70(2):87–109.

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

Fuentes, M. (2007). Approximate likelihood for large irregularly spaced spatial data. *Journal of the American Statistical Association*, 102(477):321–331.

Gneiting, T., Kleiber, W., and Schlather, M. (2010). Matérn cross-covariance functions for multivariate random fields. *Journal of the American Statistical Association*, 105(491):1167–1177.

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

Gorjanc, G. (2010). Graphical model representation of pedigree based mixed model. In *Information Technology Interfaces (ITI), 2010 32nd International Conference on*, pages 545–550. IEEE.

Grafakos, L. (2004). *Classical and modern Fourier analysis*. Prentice Hall.

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

Gröchenig, K. (2001). *Foundations of time-frequency analysis*. Birkhauser.

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.

Hasan, A., Kerrigan, E., and Constantinides, G. (2011). Solving a positive definite system of linear equations via the matrix exponential. *Proceedings of the IEEE Conference on Decision and Control and the European Control Conference (ICIAM)*, page 2299.

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.

Hiai, F. and Petz, D. (2009). Riemannian metrics on positive definite matrices related to means. *Linear Algebra and Its Applications*, 430(11-12):3105–3130.

Higham, N. and Tisseur, F. (2000). A Block Algorithm for Matrix 1-Norm Estimation, with an Application to 1-Norm Pseudospectra. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1185–1201.

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

Hochbruck, M., Lubich, C., and Selhofer, H. (1998). Exponential Integrators for Large Systems of Differential Equations. *SIAM Journal on Scientific Computing*, 19(5):1552–1574.

Hu, X., Simpson, D. P., Lindgren, F., and Rue, H. (2012). Multivariate gaussian random fields using stochastic partial differential equations. *N/A*.

Huckle, M. and Grote, M. (1997). Parallel preconditioning with sparse approximate inverses. *Siam J. Sci. Comput*, 18(3):838–853.

Hutchinson, M. (1989). A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Commun. Stat. Simula.*, 18:1059–1076.

Ilic, M., Turner, I. W., and Anh, V. (2008). A Numerical Solution Using an Adaptively Preconditioned Lanczos Method for a Class of Linear Systems Related with the Fractional Poisson Equation. *Journal of Applied Mathematics and Stochastic Analysis*, 2008:Article ID 104525.

Ilic, M., Turner, I. W., and Simpson, D. P. (2010). A restarted Lanczos approximation to functions of a symmetric matrix. *IMA Journal of Numerical Analysis*, 30:1044 – 1061.

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

Karypis, G. and Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359.

Kelley, C. (1999). *Iterative methods for optimization*, volume 18. Society for Industrial Mathematics.

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19:769–789.

Lindgren, F., Lindstrøm, J., and Rue, H. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: The SPDE approach. *Journal of the Royal Statistical Society, Series B*, 73:423–498.

Liu, J. and George, A. (1981). Computer solution of large sparse positive definite systems. Prentice Hall Professional Technical Reference.

Malinverno, A. and Briggs, V. (2004). Expanded uncertainty quantification in inverse problems: Hierarchical Bayes and empirical Bayes. *Geophysics*, 69(4):1005–1016.

Mallat, S. (1998). *A wavelet tour of signal processing*. Academic Press.

Matérn, B. (1960). Spatial variation. *Meddelanden fran Statens Skogsforskningsinstitut (Stockholm)*, 49.

MATLAB (2010). *Version 7.11.0 (R2010b)*. The MathWorks Inc., Natick, Massachusetts.

McPeters, R., Aeronautics, U. S. N., Scientific, S. A., and Branch, T. I. (1996). *Nimbus-7 Total Ozone Mapping Spectrometer (TOMS) data products user's guide*. NASA, Scientific and Technical Information Branch.

Meurant, G. and Strakos, Z. (2006). The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15:471–542.

Mrode, R. and Thompson, R. (2005). *Linear models for the prediction of animal breeding values*. Number 116. Cabi.

Nocedal, J. and Wright, S. (1999). *Numerical optimization*. Springer verlag.

Ohara, A., Suda, N., and Amari, S. (1996). Dualistic differential geometry of positive definite matrices and its applications to related problems. *Linear Algebra and Its Applications*, 247:31–53.

Paige, C. and Saunders, M. (1982). LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71.

Parker, A. and Fox, C. (2011). Sampling Gaussian distributions in Krylov spaces with conjugate gradients. *SIAM Journal on Scientific Computing. Submitted.*

Paul, M., Held, L., and Toschke, A. M. (2008). Multivariate modelling of infectious disease surveillance data. *Statistics in Medicine*, 27:6250 – 6267.

Rabben, T. E. and Ursin, B. (2009). Non-linear least-squares inversion with data-driven Bayesian regularisation. pending.

Rabben, T. E. and Ursin, B. (2011). AVA inversion of the top Utsira Sand reflection at the Sleipner field. *Geophysics*, 76(3):C53–C63.

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.

Riley, J. (1955). Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix. *Mathematical Tables and Other Aids to Computation*, 9(51):96–101.

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*, 59(2):291–317.

Rozanov, J. (1977). Markov random fields and stochastic partial differential equations. *Mathematics of the USSR-Sbornik*, 32:515.

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 Martino, S. (2007). Approximate Bayesian inference for hierarchical Gaussian Markov random field models. *Journal of statistical planning and inference*, 137(10):3177–3192.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the royal statistical society: Series B*, 71(2):319–392.

Rue, H. and Tjelmeland, H. (2002). Fitting Gaussian Markov Random Fields to Gaussian Fields. *Scandinavian Journal of Statistics*, 29:31–49.

Ruzhansky, M. and Turunen, V. (2009). *Pseudo-differential Operators and Symmetries: Background Analysis and Advanced Topics*. Birkhauser.

Saad, Y. (1992). *Numerical methods for large eigenvalue problems*. Manchester Univ Press.

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.

Samarskii, A. A., Matus, P. P., Mazhukin, V. I., and Mozolevski, I. E. (2002). Monotone Difference Schemes for Equations with Mixed Derivatives. *Computers and Mathematics*, 44:501 – 510.

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.

Shewchuk, J. (1994). An introduction to the conjugate gradient method without the agonizing pain `http://www.cs.colorado.edu/~jessup/csci5646/READINGS/painless-conjugate-gradient.pdf`.

Simon, H. (1984). Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear algebra and its applications*, 61:101–131.

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. (2008). Fast sampling from a Gaussian Markov random field using Krylov subspace 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.

Tang, J. and Saad, Y. (2010). A Probing Method for Computing the Diagonal of the Matrix Inverse. Technical report, Minnesota Supercomputing Institute for Advanced Computational Research.

Tarantola, A. (1986). A strategy for nonlinear elastic inversion of seismic reflection data. *Geophysics*, 51(10):1893–1903.

Theune, U., Jensas, I. O., and Eidsvik, J. (2010). Analysis of prior models for a blocky inversion of seismic AVA data. *Geophysics*, 75(3):C25–C35.

Tierney, L. and Kadane, J. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, pages 82–86.

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

Tyrtyshnikov, E. E. (1990). Optimal and Superoptimal Circulant Preconditioners. *SIAM Journal on Matrix Analysis and Applications*, 13:459–473.

Ulrych, T., Sacchi, M., and Woodbury, A. (2001). A Bayes tour of inversion: A tutorial. *Geophysics*, 66(1):55–69.

van den Eshof, J., Frommer, A., Lippert, T., Schilling, K., and van der Vorst, H. (2002). Numerical methods for the QCDd overlap operator. I. Sign-function and error bounds. *Computer Physics Communications*, 146(2):203–224.

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

van der Vorst, H. and Melissen, J. (1990). A Petrov-Galerkin type method for solving Axk= b, where A is symmetric complex. *Magnetics, IEEE Transactions on*, 26(2):706–708.

Whittle, P. (1963). Stochastic processes in several dimensions. *Bull. Inst. Internat. Statist*, 40:974–994.

Youzwishen, C. and Sacchi, M. (2006). Edge preserving imaging. *Journal of Seismic Exploration*, 15(1):45–57.

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.