J. R. Statist. Soc. B (2001)
63, *Part* 2, *pp.* 325–338

Fast sampling of Gaussian Markov random fields

Håvard Rue

Norwegian University for Science and Technology, Trondheim, Norway

[Received January 2000. Final revision November 2000]

Summary. This paper demonstrates how Gaussian Markov random fields (conditional autoregressions) can be sampled quickly by using numerical techniques for sparse matrices. The algorithm is general and efficient, and expands easily to various forms for conditional simulation and evaluation of normalization constants. We demonstrate its use by constructing efficient block updates in Markov chain Monte Carlo algorithms for disease mapping.

Keywords: Block sampling; Conditional autoregressive model; Divide-and-conquer strategy; Gaussian Markov random field; Markov chain Monte Carlo methods

1. Introduction

Gaussian Markov random fields (GMRFs), or conditional autoregressions as they are also known, are popular and commonly used (sub)models within spatial statistics in a broad sense; see for example Cressie (1993), Besag and Kooperberg (1995), Winkler (1995), Besag and Higdon (1999) and the references therein, Heikkinen and Arjas (1998) and Wikle *et al.* (1998). GMRFs are convenient models from both a computational and a theoretical point of view; they have a Markov property and are jointly Gaussian. The Markov property is also important for models relying on inference based on Markov chain Monte Carlo (MCMC) sampling as it will ensure rapid computation of the conditional densities.

Sampling from a GMRF is straightforward in theory as a GMRF is multivariate Gaussian and hence well-known general algorithms apply; see for example Cressie (1993), section 3.6.1. None of these algorithms take advantage of the Markov property for the GMRF, so they will be slow and computationally inconvenient for larger problems. For a GMRF with a circulant block Toeplitz covariance matrix, sampling is particularly efficient using fast Fourier transforms (Wood and Chan, 1994; Dietrich and Newsam, 1997). However, the Toeplitz assumption only applies to the simplest problems. Recently, algorithms have been constructed that make specific use of the Markov property of the GMRF, either by running a special dynamic model with artificial observed data (Lavine, 1998) or by using the propagation algorithm of Lauritzen (1992) with extensions (Dawid, 1992). However, neither of these seems particularly useful nor fast except in simple cases, e.g. for a GMRF involving the four nearest neighbours on a lattice. For more complicated and realistic neighbourhood patterns, algorithmic complexity increases very rapidly both with respect to central processor unit time and coding; see Wilkinson and Yeung (2000) for an overview.

In this paper, we show how to apply numerical techniques for sparse matrices to construct an efficient sampler for a general GMRF defined on a lattice or a graph. The algorithm is

Address for correspondence: Håvard Rue, Department of Mathematical Sciences, Norwegian University for Science and Technology, N-7491 Trondheim, Norway.
E-mail: Havard.Rue@math.ntnu.no

simple and surprisingly efficient and expands easily to complicated neighbourhood structures and various forms for conditional simulation and evaluation of the likelihood.

In addition to the traditional applications of GMRFs in spatial statistics referenced above, there are also potential new uses of GMRFs. Rue and Tjelmeland (1999) studied how to approximate a stationary Gaussian field as a GMRF (on a lattice) with somewhat surprising results: a Gaussian field with a Gaussian correlation function of range 50 could be approximated with a GMRF by using a 7×7 neighbourhood with a maximum error in the correlation function of about 0.01, and similarly with other families of correlation functions. Thus, a local GMRF can in practice replace a conventionally specified Gaussian field in applications, so posterior analyses or other tasks can benefit from the new fast algorithms and analytical simplifications for GMRFs.

In spatial models using MCMC sampling for inference, the new algorithms are useful for block sampling or block proposals depending on whether the full conditional is a GMRF or contains additional non-quadratic terms. Block sampling often improves the properties of an MCMC algorithm (Liu *et al.*, 1994). The full generality in the algorithms is important as conditioning on all other variables often results in a non-homogeneous GMRF. Previous examples in the literature include dynamic models (Frühwirth-Schnatter, 1994; Carter and Kohn, 1994; Shephard and Pitt, 1997; Knorr-Held, 1999) and expert systems (Jensen *et al.*, 1995). Knorr-Held and Rue (2000) discuss disease mapping models and find that schemes involving block updates can give misleading estimates even for long runs on some data sets for statistics where the tail behaviour is important, and that more sophisticated block updates are needed to obtain correct results. On the theoretical side, Roberts and Sahu (1997) studied the convergence rate for Gibbs block sampling of a GMRF.

A rapid evaluation of the likelihood for GMRFs is important for estimating (hyper-) parameters by using either a likelihood or Bayesian approach. For example, Griffith and Sone (1995) investigated numerical approximation formulae for the likelihood for which the neighbourhood structure is constructed by using spatial neighbouring regions, and Kiiveri (1992) numerically approximated the integral in Whittle's approximation in the stationary case. Our new algorithms can efficiently compute the likelihood for large dimensions, even in the general non-homogeneous case, which makes a numerical approximation even for simpler cases less important. Rue and Tjelmeland (1999) demonstrated that likelihood-based inference for a GMRF can be quite sensitive to model error.

The paper is organized as follows. Section 2 presents the basic algorithm for sampling from a GMRF. Section 3 discusses extensions including various forms for conditioning and evaluation of the likelihood. Section 4 discusses blocking and a divide-and-conquer strategy approach to large graphs. In Section 5, we demonstrate how the new algorithm can be used in MCMC sampling to construct a block update for a near GMRF full conditional. Computational details are described in Appendix A.

2. The algorithm

This section defines a GMRF and describes the basic algorithm for producing samples from it. We use lower-case characters for vectors and random vectors, upper-case characters for matrices and script characters for sets and graphs.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a finite undirected graph where $\mathcal{V}$ is the set of nodes (or vertices) and $\mathcal{E}$ is the set of edges. For simplicity, we number the nodes from 1 to $n = |\mathcal{V}| < \infty$. We use the following notation: $\mathbf{x}_\mathcal{A} = \{x_i: i \in \mathcal{A}\}$ for $\mathcal{A} \subseteq \mathcal{V}$, and $-\mathcal{A}$ for the set $\mathcal{V} - \mathcal{A}$. Define ∂_i to be the set of neighbours of node i, i.e. the set of all nodes adjacent to node i in the graph. A zero-mean GMRF $\mathbf{x}$ on $\mathcal{G}$ is Gaussian with zero mean and precision (inverse covariance) matrix $\mathbf{Q}$, such that Q_{ij} is 0 if and only if $j \notin \{\partial_i \cup i\}$. A GMRF is often specified through the local characteristics

$$E(x_i|\mathbf{x}_{-i}) = -\sum_{j\in\partial_i} \frac{Q_{ij}}{Q_{ii}} x_j,$$
$$\mathrm{var}(x_i|\mathbf{x}_{-i}) = Q_{ii}^{-1},$$

which illustrates the Markov property.

A common application in spatial statistics is when data are observed and the model is defined on the regions of a country, such as the map of Germany in Fig. 1(a) which has 544 regions. A common prior for the spatial component (which is discussed further in Section 5) is an intrinsic GMRF $\mathbf{x}$ where, for each region i, ∂_i is the set of all regions j geographically adjacent to i (Besag and Kooperberg, 1995)

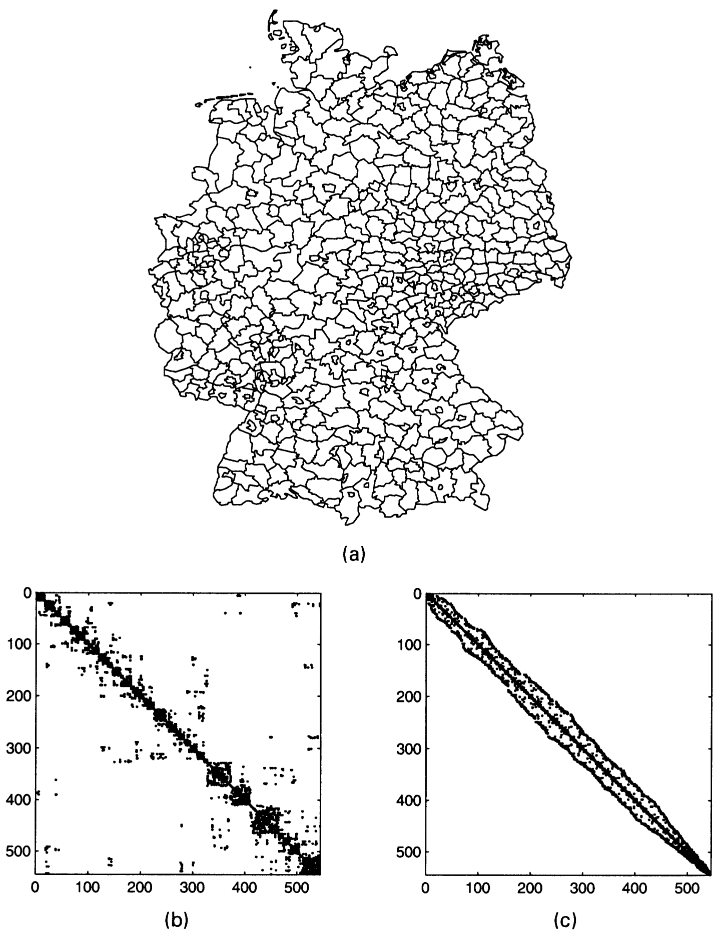

Fig. 1. (a) Map of Germany with 544 regions, (b) non-zero entries in the precision matrix **Q** for the GMRF using the numbering of the regions as in the definition of the map and (c) the same precision matrix after a permutation of the nodes to minimize the bandwidth: fast numerical algorithms exist for band matrices, and these are the basis for our algorithms

$$\pi(\mathbf{x}|\kappa) \propto \kappa^{n/2} \exp\left\{ -\tfrac{1}{2}\kappa \sum_{i\sim j} (x_i - x_j)^2 \right\}. \tag{1}$$

The graph for the GMRF is thus derived from the map. Fig. 1(b) shows the non-zero pattern in the corresponding precision matrix using the numbering of the regions as in the definition of the map. Fig. 1(c) shows the same precision matrix after a permutation of the nodes to minimize the bandwidth (we shall describe later how this is done). The bandwidth for the precision matrix is defined as $b_w = \max_{i\sim j} |i-j|$. The permuted precision matrix is a band matrix (with a smaller bandwidth) and is the basis for our algorithms; fast algorithms for most tasks exist for band matrices and their efficiency depends on b_w. Permutation of the nodes reduces b_w from 522 to 43 in this example, which allows for a potentially large acceleration.

The situation is similar for lattices where the neighbours ∂_i are the sites closest to i using some appropriate norm. Assume for simplicity that ∂_i is the $(2m+1) \times (2m+1)$ square centred at i and that the lattice has dimension n with n_1 rows and n_2 columns, where $n_1 \leqslant n_2$. In this case the precision matrix using columnwise ordering is a band matrix with bandwidth $b_w = m(n_1 + 1)$ assuming that $n_1 > m$. In this case there is no need to permute the nodes. The case $n_1 = 1$ is similar to an autoregressive process of order m and $b_w = m$.

The banded precision matrix for the GMRF (after permutation) is the basis for our sampling algorithm. There are no additional assumptions about the coefficients in the precision matrix. The algorithm goes as follows: first note that computing the Cholesky factorization of the band matrix $\mathbf{Q}$ with bandwidth b_w

$$\mathbf{Q} = \mathbf{L}\mathbf{L}^{\mathrm{T}} \tag{2}$$

is a standard problem in numerical linear algebra. By theorem 4.3.1 in Golub and van Loan (1996), $\mathbf{L}$ has lower bandwidth b_w (and is 0 above the diagonal) and is efficiently computed using only nb_w^2 floating point operations (flops); see Golub and van Loan (1996), section 4.3.5, for details and the algorithm. The matrix $\mathbf{L}$ requires nb_wU bytes of storage, where U is the number of bytes needed to store a floating point number. In the next step let $\mathbf{z}$ be a vector of independent standard Gaussian variables, and note by direct calculation that $\mathbf{x}$ defined by the solution of

$$\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{z} \tag{3}$$

has mean 0 and precision matrix $\mathbf{Q}$. Again, solving equation (3) efficiently is a standard problem in numerical linear algebra and the solution can be computed in $2nb_w$ flops by using band back-substitution; see Golub and van Loan (1996), section 4.3.2, for details. How to permute the nodes to minimize the bandwidth is also a classical numerical problem, for which there are clever and fast algorithms: see Appendix A.

Some additional notes to the algorithm are as follows.

2.1. *Repeated samples*

Generating more than one sample from the same GMRF is especially efficient. The band Cholesky factorization in equation (2) is computed only once and then equation (3) is solved for each sample.

2.2. *Computational cost*

Assume for simplicity an $n_1 \times n_2$, $n_1 \leqslant n_2$, lattice with a $(2m+1) \times (2m+1)$ neighbourhood. Computing the band Cholesky decomposition (2) costs $n_2 n_1^3 m^2$ flops, and solving equation (3) costs $2n_2n_1^2m$ flops. Note that the cost is linear in the largest dimension n_2, and cubic or quadratic in the smallest dimension n_1. Hence, it is far more efficient to sample from rectangular than from square lattices. A similar comment is also valid for graphs. For an autoregressive process with $b_w = O(1)$, the algorithm requires $O(n)$ flops.

2.3. Unified framework

Our algorithm defines a unified framework where only one algorithm and one set of computer code is need for sampling a GMRF defined either on a line, a lattice or a general graph.

2.4. A counter-example

It is not difficult to construct a graph for which the bandwidth equals the dimension of the problem, i.e. $b_w = n - 1$ for all permutations: let every node have all other nodes as neighbours. In this case, there is little to gain computationally. In spatial applications the bandwidth is most often small compared with the size of the problem $b_w = o(n)$, and often $b_w = O(\sqrt{n})$.

3. Conditioning and likelihood

We shall now discuss generalizations of the simulation algorithm to conditional simulation and evaluation of the log-likelihood.

3.1. Conditional simulation

This section discusses three common forms of conditional sampling for a GMRF:

(a) conditioning on an observed subset of $\mathbf{x}$,
(b) conditioning on 'the rest' as is common in MCMC algorithms and
(c) conditioning on some linear combinations of $\mathbf{x}$.

3.1.1. Sampling from $\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{-\mathcal{A}}$

Assume that we want to sample a subset $\mathbf{x}_{\mathcal{A}}$ of a zero-mean GMRF conditioned on the rest $\mathbf{x}_{-\mathcal{A}}$. For notational convenience, let $\mathcal{B}$ denote $-\mathcal{A}$. Then $\mathbf{x} = (\mathbf{x}_{\mathcal{A}}^{\mathrm{T}}, \mathbf{x}_{\mathcal{B}}^{\mathrm{T}})^{\mathrm{T}}$ and

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_{\mathcal{A}\mathcal{A}} & \mathbf{Q}_{\mathcal{A}\mathcal{B}} \\ \mathbf{Q}_{\mathcal{B}\mathcal{A}} & \mathbf{Q}_{\mathcal{B}\mathcal{B}} \end{pmatrix}.$$

The task is to sample from $\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{\mathcal{B}}$. This conditional density is Gaussian with mean $\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}}$ given by $\mathbf{Q}_{\mathcal{A}\mathcal{A}}\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}} = -\mathbf{Q}_{\mathcal{A}\mathcal{B}}\mathbf{x}_{\mathcal{B}}$ and precision $\mathbf{Q}_{\mathcal{A}|\mathcal{B}} = \mathbf{Q}_{\mathcal{A}\mathcal{A}}$. These simple expressions for the conditional densities are one of the great advantages of using GMRFs. The proof uses the block equations derived from $\mathbf{Q}\mathbf{Q}^{-1} = \mathbf{I}$ and standard results about conditional means and variances. Conditioning on observed values is particularly easy for a GMRF, as the conditional precision matrix is explicitly known as the precision matrix for the graph after removing all observed nodes. $\mathbf{Q}_{\mathcal{A}\mathcal{A}}$ cannot have a larger bandwidth than $\mathbf{Q}$, so it is still a band matrix (after permutation) and the algorithm in Section 2 can be used. The only difference is the need to compute $\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}}$ by solving $\mathbf{Q}_{\mathcal{A}\mathcal{A}}\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}} = \mathbf{b}$, where the right-hand side is easily computable as $\mathbf{b} = -\mathbf{Q}_{\mathcal{AB}}\mathbf{x}_{\mathcal{B}}$. The vector $\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}}$ can be obtained from the band Cholesky decomposition of $\mathbf{Q}_{\mathcal{AA}} = \mathbf{LL}^{\mathrm{T}}$, by solving efficiently by forward (band) substitution $\mathbf{Lu} = \mathbf{b}$ and a back- (band) substitution $\mathbf{L}^{\mathrm{T}}\boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}} = \mathbf{u}$. To obtain a sample, we only need to add the solution of $\mathbf{L}^{\mathrm{T}}\mathbf{y} = \mathbf{z}$ (where $\mathbf{z}$ is a vector of independent standard Gaussian variables) to the conditional mean, $\mathbf{x}_{\mathcal{A}|\mathcal{B}} = \boldsymbol{\mu}_{\mathcal{A}|\mathcal{B}} + \mathbf{y}$. The computational cost for a conditioned sample is then $n_{\mathcal{A}}b_{\mathrm{w}\mathcal{A}}^2 + 2 \times 2n_{\mathcal{A}}b_{\mathrm{w}\mathcal{A}}$ flops, where $n_{\mathcal{A}} = \dim(\mathbf{x}_{\mathcal{A}})$ and $b_{\mathrm{w}\mathcal{A}}$ is the bandwidth of the permuted $\mathbf{Q}_{\mathcal{AA}}$-matrix. Repeated samples cost $2n_{\mathcal{A}}b_{\mathrm{w}\mathcal{A}}$ flops.

The case $\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{\mathcal{B}}$ also covers the case when some observations are observed exactly and others have Gaussian errors. This is because the density for $\mathbf{x}$ conditioned on the noisy observations is again a GMRF with precision matrix $\mathbf{Q} + \mathbf{Q}^{\mathrm{obs}}$, where $\mathbf{Q}^{\mathrm{obs}}$ is the precision matrix for the observations. A common situation is to observe (conditionally independent) $y_i|\mathbf{x}$ with mean x_i and precision τ_i for $i \in \mathcal{I} \subset \mathcal{V}$. Then Q_{ii}^{obs} equals τ_i, $i \in \mathcal{I}$, and 0 otherwise.

3.1.2. *Sampling from* $\pi(\mathbf{x}) \propto \exp(-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Qx} + \mathbf{b}^{\mathrm{T}}\mathbf{x})$

A typical form of the density when deriving full conditionals for GMRFs in MCMC algorithms (examples are shown in Section 5) is

$$\pi(\mathbf{x}|\mathrm{rest}) \propto \exp(-\tfrac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Qx} + \mathbf{b}^{\mathrm{T}}\mathbf{x}). \tag{4}$$

$\mathbf{Q}$ is the precision matrix and the mean is implicitly given as $\mathbf{Q}^{-1}\mathbf{b}$, similarly to the case in Section 3.1.1. To sample from distribution (4) we therefore compute the band Cholesky decomposition $\mathbf{Q} = \mathbf{LL}^{\mathrm{T}}$, solve the systems $\mathbf{Lv} = \mathbf{b}$, $\mathbf{L}^{\mathrm{T}}\boldsymbol{\mu} = \mathbf{v}$ and $\mathbf{L}^{\mathrm{T}}\mathbf{y} = \mathbf{z}$, and sum to obtain the sample $\mathbf{x} = \boldsymbol{\mu} + \mathbf{y}$.

3.1.3. *Sampling from* $\mathbf{x}|\mathbf{Ax} = \mathbf{b}$

A more general problem is to sample $\mathbf{x}$ conditioned on $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a $k \times n$ matrix with rank k. A conditional sample can be obtained by using the fact that

$$\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}}(\mathbf{AQ}^{-1}\mathbf{A}^{\mathrm{T}})^{-1}(\mathbf{Ax} - \mathbf{b}), \tag{5}$$

where $\mathbf{x}$ is an unconditional sample, has the correct conditional density. Equation (5) is commonly referred to as conditioning using kriging (Cressie (1993), section 3.6.2). It is easy to modify equation (5) when $\mathbf{Ax} - \mathbf{b}$ is Gaussian noise and not zero. In equation (5) we need to compute $\mathbf{V} = \mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}}$, which requires solving $\mathbf{QV}_i = (\mathbf{A}^{\mathrm{T}})_i$ for each of the k columns. Since the band Cholesky decomposition is already available as $\mathbf{Q} = \mathbf{LL}^{\mathrm{T}}$, we solve $\mathbf{QV}_i = (\mathbf{A}^{\mathrm{T}})_i$ by a forward substitution $\mathbf{Lu} = (\mathbf{A}^{\mathrm{T}})_i$ and a back-substitution $\mathbf{L}^{\mathrm{T}}\mathbf{V}_i = \mathbf{u}$. We compute $\mathbf{W} = \mathbf{V}(\mathbf{AV})^{-1}$ only once, and for each sample we then need to compute $\mathbf{Ax} - \mathbf{b}$, to premultiply by $-\mathbf{W}$ and to add $\mathbf{x}$.

3.2. *Evaluation of the likelihood*

It is often important to compute the likelihood for a sample $\mathbf{x}$ from a zero-mean GMRF. The band Cholesky factorization of the (permuted) precision matrix allows a fast evaluation of the likelihood as $|\mathbf{Q}|^{1/2} = \Pi_i L_{ii}$. Hence, the log-likelihood $l_{\mathbf{x}}(\mathbf{x})$ is

$$l_{\mathbf{x}}(\mathbf{x}) = -\frac{n}{2}\log(2\pi) + \sum_i \log(L_{ii}) - \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Qx}. \tag{6}$$

If $\mathbf{x}$ is simulated by solving $\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{z}$, then $\mathbf{x}^{\mathrm{T}}\mathbf{Qx} = \mathbf{z}^{\mathrm{T}}\mathbf{z}$ and $l(\mathbf{x})$ can be evaluated at no extra costs. However, $\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}$ is fast to compute; the computation of the log-likelihood for the conditional sample in Section 3.1.1 and Section 3.1.2 is similar.

The likelihood is also often needed for $\tilde{\mathbf{x}}$, a sample from $\mathbf{x}|\mathbf{A}\mathbf{x} = \mathbf{b}$ computed by equation (5), and for an indirectly observed GMRF. Consider first the likelihood for $\tilde{\mathbf{x}}$. Denote by $l_{\tilde{\mathbf{x}}}(\cdot)$ the log-likelihood for the conditional density and $l_{\mathbf{A}\mathbf{x}}(\cdot)$ the log-likelihood for $\mathbf{A}\mathbf{x}$. By using the identity

$$\pi(\mathbf{x}|\mathbf{A}\mathbf{x}) = \pi(\mathbf{A}\mathbf{x}|\mathbf{x})\,\pi(\mathbf{x})/\pi(\mathbf{A}\mathbf{x})$$

assuming that $\pi(\mathbf{A}\mathbf{x}) > 0$, we obtain

$$l_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) = l(\tilde{\mathbf{x}}) - l_{\mathbf{A}\mathbf{x}}(\mathbf{b}) - \tfrac{1}{2}\log(|AA^{\mathrm{T}}|) \tag{7}$$

where $|\cdot|$ denotes the determinant. Note that $l_{\mathbf{A}\mathbf{x}}(\mathbf{b})$ does not depend on the values of $\mathbf{x}$ or $\tilde{\mathbf{x}}$. The term $l(\tilde{\mathbf{x}})$ is evaluated efficiently by using equation (6), whereas the determinant and $l_{\mathbf{A}\mathbf{x}}(\mathbf{b})$ only require operations on matrices of dimension $k \times k$, where usually $k \ll n$. An application of equation (7) is found in Knorr-Held and Rue (2000) where an efficient block update move is constructed under two linear constraints where parameters in the precision matrix are simultaneously updated. For an indirectly observed GMRF, we observe data $\mathbf{y} = \mathbf{x} + \boldsymbol{\epsilon}$ where $\mathbf{x}$ is a GMRF and the noise $\boldsymbol{\epsilon}$ is a GMRF and independent of $\mathbf{x}$. We can compute $\pi(\mathbf{y})$ efficiently by using that

$$\pi(\mathbf{y}) = \pi(\mathbf{y}|\mathbf{x})\,\pi(\mathbf{x})/\pi(\mathbf{x}|\mathbf{y}),$$

then choose $\mathbf{x} = \mathbf{0}$ and use that all terms on the right-hand side are GMRFs.

4. Sampling a Gaussian Markov random field on a large graph

For a GMRF on a very large graph it will still be inconvenient to sample $\mathbf{x}$ or to evaluate the likelihood. This is due to both the central processor unit time required and the amount of memory needed to store the band Cholesky factorization, although they are closely related. To give an idea of how large a 'large graph' is, our Sun Ultra 2 computer can at the time of writing handle lattice cases up to size 256×256 with a 5×5 neighbourhood. Beyond that, the performance drops because of memory problems.

We shall discuss two strategies to sample a GMRF on a large graph. The first approach reduces the computational burden by using blocking strategies, similarly to a Gibbs sampler for sampling, and pseudolikelihood as an approximation to the likelihood. The second approach reduces the memory requirement by using a divide-and-conquer strategy to split the problem into simpler problems by using iterative linear solvers for large linear systems. The strategies can of course be combined.

4.1. Blocking strategies

Sampling a GMRF on a large graph can be approached in a way that is similar to that in which MCMC methods are used to sample complicated non-Gaussian densities. It is easy to construct a (Gibbs) block sampler for a GMRF, by dividing the nodes $\mathcal{V}$ into (possible overlapping) sets $\{\mathcal{A}_i\}$ and sampling successively from $\pi(\mathbf{x}_{\mathcal{A}_i}|\mathbf{x}_{-\mathcal{A}_i})$ using the algorithm in Section 3.1. To illustrate the computational savings by using this strategy, consider an $n_1 \times n_1$ lattice with a $(2m+1) \times (2m+1)$ neighbourhood. Assuming disjoint sets where each $\mathcal{A}_i$ is an $n_1/k \times n_1/k$ lattice, the relative computational burden is $1/k^2$ for computing all the band Cholesky factorizations, and $1/k$ for doing repeated sampling.

The pseudolikelihood approach uses $\mathrm{PL}(\mathbf{x}) = \Pi_i \, \pi(x_i|\mathbf{x}_{-i})$ as an approximation to the likelihood. It seems more sensible to use a block version of the pseudolikelihood: partition $\mathcal{V}$ into disjoint sets $\{\mathcal{A}_i\}$ and combine the conditional likelihoods into

$$\mathrm{PLBlock}(\mathbf{x}) = \prod_i \pi(\mathbf{x}_{\mathcal{A}_i}|\mathbf{x}_{-\mathcal{A}_i}).$$

PLBlock($\mathbf{x}$) can easily be computed by using the algorithm in Section 3.1 to compute the conditional densities, and equation (6) to evaluate each (conditional) likelihood.

One way to partition $\mathcal{V}$, which only depends on the graph $\mathcal{G}$, into disjoint sets $\{\mathcal{A}_i\}$ is to choose $\{\mathcal{A}_i\}$ such that, for all i, $\mathbf{Q}_{\mathcal{A}_i,-\mathcal{A}_i}$ has few non-zero elements. In this way, we hope that the dependence within $\mathbf{x}_{\mathcal{A}_i}$ is large and lower between $\mathbf{x}_{\mathcal{A}_i}$ and $\mathbf{x}_{-\mathcal{A}_i}$, but in general it depends on $\mathbf{Q}$ and not just on $\mathcal{G}$. Computing the permutation of the nodes remains feasible even for large graphs and this can be used automatically to define partitions. If we assume an autoregressive process of order b_w with non-zero coefficients and even n, then it is not difficult to show that choosing $\mathcal{A} = \{i \in \mathcal{V}\text{: } i \leqslant n/2\}$ will minimize the number of non-zero elements of $\mathbf{Q}_{\mathcal{A},-\mathcal{A}}$ when $\mathcal{A}$ and $-\mathcal{A}$ are of the same size. In general, we can think of the permuted precision matrix as the precision matrix for a one-dimensional non-stationary autoregressive process of order b_w where some of the coefficients are 0; see Fig. 1(c). A heuristic approach to defining partitions is therefore the following: to divide $\mathcal{V}$ into four disjoint sets of near equal size, we use the permuted set of nodes and let $\mathcal{A}_1$ consist of those nodes where the permuted node is in $[1, \ldots, n/4]$, and so on. If we require overlapping sets, we can choose $L > 0$ and use those in $[1, \ldots, n/4 + L]$ and $[n/4 - L + 1, \ldots, 2n/4 + L]$ as $\mathcal{A}_1$ and $\mathcal{A}_2$, and so on.

4.2. Using a divide-and-conquer strategy

For large graphs, the memory requirement for the band Cholesky factorization is nb_w and often $O(n^{3/2})$. The reason is that the sampling algorithm in Section 2 needs a split of the precision matrix $\mathbf{Q} = \mathbf{L}\mathbf{L}^\mathrm{T}$. Iterative solvers for large, sparse and positive definite linear systems can solve $\mathbf{Qx} = \mathbf{b}$ using only $O(n)$ memory. We shall show how we can construct a sampling algorithm based on a divide-and-conquer strategy (using the Markov property) that is mainly based on iterative solving systems such as $\mathbf{Qx} = \mathbf{b}$. This strategy can be used for dividing large graphs into smaller graphs such that the algorithm in Section 2 applies to each subgraph.

4.2.1. *Solving large, sparse and positive definite linear systems*

The sampling algorithms in Sections 2 and 3 use the band Cholesky factorization $\mathbf{Q} = \mathbf{L}\mathbf{L}^\mathrm{T}$ to sample and solve $\mathbf{Qx} = \mathbf{b}$. The divide-and-conquer strategy derives a sampling algorithm based on solving equations like $\mathbf{Qx} = \mathbf{b}$ for a large but sparse positive definite $\mathbf{Q}$. We can solve such equations by using iterative techniques using only $O(n)$ memory. Again, this is a standard problem in numerical linear algebra; see for example Golub and van Loan (1996). The basic method is an iterative conjugate gradient algorithm that seeks the minimum of $\frac{1}{2}\mathbf{x}^\mathrm{T}\mathbf{Qx} - \mathbf{b}^\mathrm{T}\mathbf{x}$ in a direction orthogonal (in $\mathbf{Q}$-norm) to all previous search directions. The solution of the minimization problem is the solution of $\mathbf{Qx} = \mathbf{b}$. Only matrix–vector products like $\mathbf{Qv}$ need to be computed. Additionally, the system is linearly transformed to improve the conditioning of $\mathbf{Q}$ and hence the convergence rate.

4.2.2. *The divide-and-conquer idea*

The divide-and-conquer idea is best illustrated when the graph is a lattice, and we assume for the moment a zero-mean GMRF defined on an $n_1 \times n_1$ lattice with a 3×3 neighbourhood.

Assume also that n_1 is odd, say $n_1 = 2c + 1$. Generalizations are straightforward once the idea is clear.

Let $\mathcal{C}$ be the nodes in the $(c + 1)$th column. Define $\mathcal{A}$ and $\mathcal{B}$ as the nodes to the left and right of the $(c + 1)$th column respectively. Then $\mathbf{x} = (\mathbf{x}_{\mathcal{A}}^{\mathrm{T}}, \mathbf{x}_{\mathcal{C}}^{\mathrm{T}}, \mathbf{x}_{\mathcal{B}}^{\mathrm{T}})^{\mathrm{T}}$ and

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_{\mathcal{AA}} & \mathbf{Q}_{\mathcal{AC}} & \mathbf{0} \\ \mathbf{Q}_{\mathcal{CA}} & \mathbf{Q}_{\mathcal{CC}} & \mathbf{Q}_{\mathcal{CB}} \\ \mathbf{0} & \mathbf{Q}_{\mathcal{BC}} & \mathbf{Q}_{\mathcal{BB}} \end{pmatrix};$$

$\mathbf{Q}_{\mathcal{AB}}$ is zero since $\mathcal{C}$ is a separating subset. Assume for a moment that we can simulate $\mathbf{x}_{\mathcal{C}}$ from its marginal density $\pi(\mathbf{x}_{\mathcal{C}})$. (We shall later show how to compute $\pi(\mathbf{x}_{\mathcal{C}})$.) A sample from the joint density $\pi(\mathbf{x})$ is then completed by sampling $\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{\mathcal{C}}$ and $\mathbf{x}_{\mathcal{B}}|\mathbf{x}_{\mathcal{C}}$ from their conditional densities. As in Section 3.1.1, the conditional precision matrix and mean for $\mathbf{x}_{\mathcal{A}}|\mathbf{x}_{\mathcal{C}}$ are $\mathbf{Q}_{\mathcal{A}|\mathcal{C}} = \mathbf{Q}_{\mathcal{AA}}$ and $\boldsymbol{\mu}_{\mathcal{A}|\mathcal{C}} = \mathbf{Q}_{\mathcal{AA}}^{-1}\mathbf{Q}_{\mathcal{AC}}\mathbf{x}_{\mathcal{C}}$, and similarly with $\mathbf{x}_{\mathcal{B}|\mathcal{C}}$. Both $\mathbf{Q}_{\mathcal{A}|\mathcal{C}}$ and $\mathbf{Q}_{\mathcal{B}|\mathcal{C}}$ have the same simple band structure as $\mathbf{Q}$, so we can solve the equations for the conditional means by using the iterative methods described in Section 4.2.1. Hence, we have divided our problem into two separate smaller problems of the same type as the original problem.

The remaining task is to compute the marginal density for $\mathbf{x}_{\mathcal{C}}$. As only the precision matrix $\mathbf{Q}$ is available, we need to compute the covariance matrix $\boldsymbol{\Sigma}_{\mathcal{CC}}$ for $\mathbf{x}_{\mathcal{C}}$. Starting with $\mathbf{Q}\boldsymbol{\Sigma} = \mathbf{I}$, we can write out the equations for block column $c + 1$ in $\boldsymbol{\Sigma}$ (with obvious changes in notation)

$$\mathbf{Q}\begin{pmatrix} \boldsymbol{\Sigma}_{1,c+1} \\ \vdots \\ \boldsymbol{\Sigma}_{c+1,c+1} \\ \vdots \\ \boldsymbol{\Sigma}_{n_1,c+1} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{I} \\ \vdots \\ \mathbf{0} \end{pmatrix}, \tag{8}$$

where $\boldsymbol{\Sigma}_{c+1,c+1}$ is $\boldsymbol{\Sigma}_{\mathcal{CC}}$. The right-hand side is zero except for an $n_1 \times n_1$ identity matrix $\mathbf{I}$. Note that equation (8) defines n_1 linear systems, each with coefficient matrix $\mathbf{Q}$, and hence is solved by using iterative techniques as described in Section 4.2.1. From the kth solution we can extract column k of $\boldsymbol{\Sigma}_{\mathcal{CC}}$. We simulate $\mathbf{x}_{\mathcal{C}}$ using standard simulation procedures by computing the Cholesky decomposition $\boldsymbol{\Sigma}_{\mathcal{CC}} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$, and then computing $\mathbf{x}_{\mathcal{C}} = \mathbf{L}\mathbf{z}$, where $\mathbf{z}$ is a Gaussian $n_1 \times 1$ vector with zero mean and covariance matrix $\mathbf{I}$.

4.2.3. Summarizing remarks

The divide-and-conquer step in Section 4.2.2 demonstrates how we can split a problem into two (or in general more) smaller problems of the same type, if we know the marginal density of the separating subset. We solve the linear systems by using iterative methods (which have no memory requirements) as described in Section 4.2.1. After the divide-and-conquer step has been performed, we continue to apply the divide-and-conquer step on each subproblem that is too large; otherwise we switch to the basic algorithm in Section 2.

The generalization from a lattice to a general graph is immediate. The only problem is how to extract automatically a separating subset which divides the problem into smaller problems of approximately the same size. As in Section 4.1, the computed permutation of the nodes is again useful; those nodes with a permuted node in the interval $[n/2 - b_w/2, n/2 + b_w/2]$ (with obvious integer corrections) will be a separating subset.

5. An application in Markov chain Monte Carlo sampling

In this section we shall demonstrate how the algorithm can be used to derive block updates in MCMC algorithms for disease mapping for near GMRF full conditionals. Knorr-Held and Rue (2000) have recently extended these ideas, and we refer the reader to this report for a thorough study.

The spatial mapping of risk for a particular disease, based on observed incidence/mortality in counties or other administrative zones, is of importance for the formulation and validation of aetiological hypotheses. The basic model is as follows (Besag *et al.*, 1991): suppose that the area of interest is divided into n regions, as shown in Fig. 1(a). Let $y_1, \ldots, y_n$ be the number of deaths from the disease of interest during the study period. The common assumption is that y_i is Poisson distributed with mean $e_i r_i$ where $\mathbf{e} = (e_1, \ldots, e_n)^{\mathrm{T}}$ are known constants accounting for the number of inhabitants etc. The relative risks $\mathbf{r}$ are assumed to be decomposed as $\log(r_i) = u_i + v_i$, where $\mathbf{v}$ are independent Gaussian zero-mean random effects with unknown precision λ and $\mathbf{u}$ is an improper GMRF with density (1) where κ is unknown.

A single-site MCMC algorithm for this model is easy to construct (Besag *et al.*, 1991). Using gamma priors for κ and λ, the corresponding full conditionals are still gamma distributed and hence easy to sample. For $\mathbf{v}$, we can use a componentwise random walk Metropolis step. The full conditional for the remaining term $\mathbf{u}$ is

$$\pi(\mathbf{u}|\text{rest}) \propto \exp\left[-\tfrac{1}{2}\kappa \sum_{i\sim j} (u_i - u_j)^2 + \sum_i \{u_i y_i - e_i \exp(u_i + v_i)\}\right]. \tag{9}$$

A natural choice is to use single-site updates for u_i, using an adaptive rejection sampling algorithm (Gilks, 1996) since $\pi(u_i|\text{rest})$ is log-concave. Armed with the new algorithms for GMRFs, we can easily construct efficient block updates of $\mathbf{u}$. Since distribution (9) is not a GMRF owing to the exponential term $\exp(u_i)$, we approximate the exponential term in a neighbourhood around u_i as

$$\exp(u_i') \approx A_i + B_i u_i' + \tfrac{1}{2} C_i u_i'^2 \tag{10}$$

where the coefficients depends on u_i. Hence, we can use the following (non-homogeneous) GMRF as the proposal density:

$$q(\mathbf{u}'|\mathbf{u}) = \frac{1}{Z(\mathbf{u})} \exp\left[-\tfrac{1}{2}\kappa \sum_{i\sim j} (u_i' - u_j')^2 + \sum_i \{u_i' y_i - e_i \exp(v_i)(B_i u_i' + \tfrac{1}{2} C_i u_i'^2)\}\right]$$

which is of form (4) with

$$Q_{ij} = \begin{cases} \sum_{k\sim i} \kappa + e_i \exp(v_i) C_i, & i = j, \\ -\kappa, & i \sim j, \\ 0, & \text{otherwise,} \end{cases}$$

and $b_i = y_i - e_i \exp(v_i) B_i$. We also need the normalization constant $Z(\mathbf{u})$ as it depends on $\mathbf{u}$ through the coefficients $\{C_i\}$; see Section 3.2. The block proposal is accepted with probability $\min\{1, R\}$, where

$$R = \frac{\pi(\mathbf{u}'|\text{rest})}{\pi(\mathbf{u}|\text{rest})} \frac{q(\mathbf{u}|\mathbf{u}')}{q(\mathbf{u}'|\mathbf{u})}. \tag{11}$$

Let us return to the quadratic approximation (10) to the exponential distribution. A natural (and common) choice is to use a Taylor expansion around u_i. Studying equation (11), it seems more important to provide an 'overall' good fit to the full conditional for $\mathbf{u}$ in the region where $\mathbf{u}'$ is expected to be located, rather than a precise fit around $\mathbf{u}$. This motivates the approximation given by

$$(A_i, B_i, C_i) = \arg\min \left[\int_{u_i-\Delta}^{u_i+\Delta} \{\exp(u_i') - (A_i + B_i u_i' + \tfrac{1}{2} C_i u_i'^2)\}^2 \, \mathrm{d}u_i' \right]$$

where Δ is a crude guess of the step length of u_i to u_i'.

To study the (computational) performance of the GMRF part of the MCMC algorithm, we ran the model on the oral cavity cancer mortality data for males in Germany (1986–1990) analysed by Knorr-Held and Raßer (2000). The map of Germany appears in Fig. 1(a), and the neighbours of each region i are all other regions j adjacent to i. This graph has $n = 544$ nodes with average 5.2, minimum 1 and maximum 11 neighbours. The sampling of the GMRF specified by expression (4) took about 0.008 s on a Sun Ultra 2 model 1296 computer with a 296-MHz SUNW UltraSPARC-II processor. The cost of one block proposal of $\mathbf{u}$ is about 2×0.008 s.

Table 1 shows the average acceptance rate and an overall step length $\{(\mathbf{u}' - \mathbf{u})^{\mathrm{T}} \mathbf{Q} (\mathbf{u}' - \mathbf{u})\}^{1/2}$ for accepted $\mathbf{u}$-proposals, where $\mathbf{Q}$ is the prior precision matrix for $\mathbf{u}$ for varying values of Δ. The tabulated values must be compared with the acceptance rate of 16.3% and step length 7.6 using the Taylor expansion. As the results show, the performance of the block update for $\mathbf{u}$ is quite robust for changes in Δ, and a reasonable Δ improves the acceptance rate compared with using a Taylor expansion, keeping the step length approximately constant. The average acceptance rate will tend to increase with increasing κ as our GMRF approximation to the full conditional becomes increasingly accurate.

Knorr-Held and Rue (2000) reported a thorough study of different blocking schemes in MCMC algorithms for different disease mapping models based on extending the ideas in this section. The basis is the fast general sampling algorithm for a GMRF. Their extension goes along the following lines. First observe that the full conditional for $\mathbf{w} = (\mathbf{u}, \mathbf{u} + \mathbf{v})$ is

$$\pi(\mathbf{w}|\lambda, \kappa, \mathbf{y}) \propto \exp\left\{ -\tfrac{1}{2} \mathbf{w}^{\mathrm{T}} \widetilde{\mathbf{Q}} \mathbf{w} + \mathbf{b}^{\mathrm{T}} \mathbf{w} - \sum_i c_i \exp(w_i) \right\},$$

where the precision matrix $\widetilde{\mathbf{Q}}$ and vector $\mathbf{b}$ depend on (λ, κ), and $\{c_i\}$ are some constants. Note that the form is similar to expression (9), but the graph itself is different. Hence they did block updates of $\mathbf{w}$ following the same idea as presented here; propose from a GMRF

Table 1. Average acceptance rate and average overall step length (for accepted proposals) for the block update of the $\mathbf{u}$ in the disease mapping example

Δ	*Acceptance rate (%)*	*Step length*
0.1	18.9	7.7
0.2	22.8	7.3
0.25	28.2	7.8
0.3	28.4	7.6
0.35	26.6	7.4
0.4	23.2	7.5

approximation around the current value of $\mathbf{w}$. They also constructed joint updates of the hyperparameters (λ, κ) and $\mathbf{w}$ by using the proposal

$$q(\mathbf{w}', \lambda', \kappa'|\mathbf{w}, \lambda, \kappa) = f_\lambda(\lambda'|\lambda)\, f_\kappa(\kappa'|\kappa)\, q(\mathbf{w}'|\mathbf{w}, \lambda', \kappa')$$

where f_λ and f_κ are, for example, log-random-walks around previous values, and $q(\mathbf{w}'|\mathbf{w}, \lambda', \kappa')$ is the GMRF approximation around $\mathbf{w}$ to the full conditional for $\mathbf{w}$ using the proposed new values (λ', κ'). The joint update of $\mathbf{w}$ and (λ, κ) gives an impressive mixing rate; see Knorr-Held and Rue (2000), who reported reasonable acceptance rates (greater than 10%) even when updating nearly 3000 variables in one block in a joint model analysing two diseases jointly. However, the acceptance rates will not be that large in general as the acceptance rate will decrease when the relative weight of the random effect compared with the spatial component increases.

Acknowledgements

The author thanks Merrilee A. Hurn, Oddvar K. Husby, Leonhard Knorr-Held, Arne Martinsen, Håkon Tjelmeland and Darren J. Wilkinson for stimulating discussions, and Leonhard Knorr-Held for providing the oral cavity cancer data and the regional map of Germany. This work has been supported by European Union training and mobility of researchers network ERB-FMRX-CT96-0095 on 'Computational and statistical methods for the analysis of spatial data'.

Appendix A: Computational and algorithmic details

A library written in C, `GMRFsim`, implementing the algorithms presented is available at

```
http://www.math.ntnu.no/~hrue/GMRFsim
```

The basic algorithm has three steps:

Step 1: permute the indices in $\mathbf{Q}$ by using a permutation matrix $\mathbf{P}$ such that $\mathbf{Q}^{\text{perm}} = \mathbf{PQP}^{\mathrm{T}}$ has a small bandwidth b_{w}.
Step 2: compute the band Cholesky factorization $\mathbf{Q}^{\text{perm}} = \mathbf{L}^{\mathrm{T}}\mathbf{L}$.
Step 3: use band solvers to solve $\mathbf{L}^{\mathrm{T}}\mathbf{x}^{\text{perm}} = \mathbf{z}$, where the z_i are independent standard Gaussians, and apply the inverse permutation $\mathbf{x} = \mathbf{P}^{\mathrm{T}}\mathbf{x}^{\text{perm}}$. Then, $\mathbf{x}$ is a sample from the zero-mean GMRF.

Although these steps are rather basic for a numerically trained researcher, we shall briefly describe each step for the statistically oriented reader.

A.1. Step 1

For step 1 we used the implementation that is available at

```
http://www.netlib.org/toms/582
```

The problem of how to permute the indices in a sparse matrix to obtain a small bandwidth is a standard problem in numeric linear algebra; see for example Saad (1996). Common algorithms are based on level sets: a level set is defined recursively as the set of all unmarked neighbours of all the nodes of a previous level set. Starting from a cleverly chosen initial node in the graph, the level sets are constructed and traversed in increasing numbers of neighbours. The order by which the nodes are traversed in this algorithm determines the permutation of the indices.

A more modern package for computing bandwidth reducing permutations is METIS:

```
http://www-users.cs.umn.edu/~karypis/metis
```

A.2. Step 2

High quality routines for computing the band Cholesky factorization are available in the public domain Lapack library written in Fortran which can be downloaded from

```
http://www.netlib.org
```

The appropriate routines are `dpbtf2` (level 2) or `dpbtrf` (level 3).

Computing the band Cholesky factorization is a standard numerical problem. The knowledge that $\mathbf{L}$ in $\mathbf{Q}^{\text{perm}} = \mathbf{L}^{\text{T}}\mathbf{L}$ is a lower triangular matrix with (lower) bandwidth b_{w} simply says that we do not need to compute all the 0s as in a full Cholesky factorization. Hence, great computational savings are obtained.

There are alternatives to computing the band Cholesky factorization (but still $\mathbf{Q} = \mathbf{V}\mathbf{V}^{\text{T}}$ for some lower sparse triangular matrix $\mathbf{V}$) as the best permutation in terms of the number of non-zeros in $\mathbf{V}$ of the nodes does not result in a narrow bandwidth; the non-zeros may lie well off the diagonal. This is especially a benefit for large graphs, and there are several packages for this purpose, like `PSPASES`,

```
http://www-users.cs.umn.edu/~mjoshi/pspases
```

`SUPERLU`

```
http://www.nersc.gov/~xiaoye/SuperLU
```

or `S+`

```
http://www.cs.ucsb.edu/research/S+
```

`SUPERLU` and `S+` are for general sparse matrices whereas `PSPASES` is for positive definite matrices. There are also versions of these packages for shared memory and distributed memory parallel machines, which immediately give a parallel implementation of the sampling algorithm with high performance.

A.3. Step 3

To solve $\mathbf{L}^{\text{T}}\mathbf{x}^{\text{perm}} = \mathbf{z}$ we make use of a band linear solver which offers computational savings compared with a full back-substitution owing to the large number of known 0s. Here we used the band solver `dtbsv` in the Lapack library.

References

Besag, J. and Higdon, D. (1999) Bayesian analysis of agricultural field experiments (with discussion). *J. R. Statist. Soc.* B, **61**, 691–746.

Besag, J. and Kooperberg, C. (1995) On conditional and intrinsic autoregressions. *Biometrika*, **82**, 733–746.

Besag, J., York, J. and Mollié, A. (1991) Bayesian image restoration with two applications in spatial statistics (with discussion). *Ann. Inst. Statist. Math.*, **43**, 1–59.

Carter, C. K. and Kohn, R. (1994) On Gibbs sampling for state space models. *Biometrika*, **81**, 541–543.

Cressie, N. A. C. (1993) *Statistics for Spatial Data*, 2nd edn. New York: Wiley.

Dawid, A. P. (1992) Applications of a general propagation algorithm for probabilistic expert systems. *Statist. Comput.*, **2**, 25–36.

Dietrich, C. R. and Newsam, G. N. (1997) Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix. *SIAM J. Scient. Comput.*, **18**, 1088–1107.

Frühwirth-Schnatter, S. (1994) Data augmentation and dynamic linear models. *J. Time Ser. Anal.*, **15**, 183–202.

Gilks, W. R. (1996) Full conditional distributions. In *Markov Chain Monte Carlo in Practice* (eds W. R. Gilks, S. Richardson and D. J. Spiegelhalter), pp. 75–88. London: Chapman and Hall.

Golub, G. H. and van Loan, C. F. (1996) *Matrix Computations*, 3rd edn. Baltimore: Johns Hopkins University Press.

Griffith, D. A. and Sone, A. (1995) Trade-offs associated with normalizing constants computational simplifications for estimating spatial statistical models. *J. Statist. Computn Simuln*, **51**, 165–183.

Heikkinen, J. and Arjas, E. (1998) Non-parametric Bayesian estimation of a spatial Poisson intensity. *Scand. J. Statist.*, **25**, 435–450.

Jensen, C. S., Kong, A. and Kjærulff, U. (1995) Blocking-Gibbs sampling in very large probabilistic expert systems. *Int. J. Hum. Comput. Stud.*, **42**, 647–666.

Kiiveri, H. T. (1992) Fitting spatial correlation models: approximating a likelihood approximation. *Aust. J. Statist.*, **34**, 497–512.

Knorr-Held, L. (1999) Conditional prior proposals in dynamic models. *Scand. J. Statist.*, **26**, 129–144.

Knorr-Held, L. and Raßer, G. (2000) Bayesian detection of clusters and discontinuities in disease maps. *Biometrics*, **56**, 13–21.

Knorr-Held, L. and Rue, H. (2000) On block updating in Markov random field models for disease mapping. *Technical Report 210*. Institute of Statistics, University of Munich, Munich.

Lauritzen, S. L. (1992) Propagation of probabilities, means and variances in mixed graphical association models. *J. Am. Statist. Ass.*, **87**, 1098–1108.

Lavine, M. (1998) Another look at conditionally Gaussian Markov random fields. In *Bayesian Statistics 6* (eds J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith). Oxford: Oxford University Press.

Liu, J. S., Wong, W. H. and Kong, A. (1994) Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, **81**, 27–40.

Roberts, G. O. and Sahu, S. K. (1997) Updating schemes, correlation structure, blocking and parameterization for the Gibbs sampler. *J. R. Statist. Soc.* B, **59**, 291–317.

Rue, H. and Tjelmeland, H. (1999) Fitting Gaussian Markov random fields to Gaussian fields. *Statistics Report 16*. Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim.

Saad, Y. (1996) *Iterative Methods for Sparse Linear Systems*. Boston: PWS.

Shephard, N. and Pitt, M. K. (1997) Likelihood analysis of non-Gaussian measurement time series. *Biometrika*, **84**, 653–667.

Wikle, C. K., Berliner, L. M. and Cressie, N. A. (1998) Hierarchical Bayesian space-time models. *Environ. Ecol. Statist.*, **5**, 117–154.

Wilkinson, D. J. and Yeung, S. K. H. (2000) Conditional simulation from highly structured Gaussian systems, with application to blocking-MCMC for the Bayesian analysis of very large linear models. *Technical Report 9*. Department of Statistics, University of Newcastle, Newcastle upon Tyne.

Winkler, G. (1995) *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*. Berlin: Springer.

Wood, A. T. A. and Chan, G. (1994) Simulation of stationary Gaussian processes in $[0, 1]^d$. *J. Comput. Graph. Statist.*, **3**, 409–432.