IMAGE DEBLURRING, GAUSSIAN MARKOV RANDOM FIELDS, 
AND NEUMANN BOUNDARY CONDITIONS

JOHNATHAN M. BARDSELY, MARYLESA HOWARD, AND JAMES G. NAGY

Abstract. In this paper we consider the inverse problem of image deblurring with Neumann boundary conditions. Regularization is incorporated by using Gaussian Markov random fields (GM-RFs) to model an appropriate prior on the image pixel values. We provide a linear algebraic framework for GMRFs, and we establish an important connection between GMRFs studied in the statistical literature, and negative-Laplacian-based Tikhonov regularization used in the inverse problems and imaging communities. This connection allows us to show that the negative Laplacian Tikhonov regularization method corresponds to concrete statistical assumptions about the unknown pixel values conditioned on those of its neighbors. For image reconstruction, we implement a Markov Chain Monte Carlo (MCMC) method that yields samples of the unknown image $x$ and of the regularization parameter $\alpha$. From the samples, we compute the reconstructed image and quantify uncertainty in both $x$ and $\alpha$. Furthermore, we show that the approach can be implemented very efficiently by exploiting matrix structure, through the use of the discrete cosine transform.

Key words. image deblurring, discrete cosine transform, Bayesian inference, Gaussian Markov random fields, Markov chain Monte Carlo methods, Neumann boundary conditions, Laplacian, Tikhonov regularization

AMS Subject Classifications: 15A29, 62F15, 65F22, 94A08.

1. Introduction. In applications such as astronomy, medicine, physics and biology, digital images are used by scientists and practitioners to record and analyze unique events. Environmental effects and imperfections in the imaging system can cause the recorded images to be degraded by blurring and noise. Unfortunately, it is not always possible to repeat the process used to record the image to obtain “better pictures”; for example, it could be too costly to repeat a particular experiment, or it may not be physically possible to repeat the event that was observed. In such cases, computational post processing techniques, called image deblurring, are used to improve the resolution of the image.

Image deblurring is typically modeled as a linear inverse problem. Suppose $x(t)$, \( t \in \mathbb{R}^d \), is a function describing the true $d$-dimensional image; e.g., for a plane image containing pixels, $d = 2$. The image formation process, which includes blurring and noise, is modeled by an integral equation,

$$b(s) = \int_{\Omega} k(s, t)x(t)dt + \eta(s),$$

where $s \in \mathbb{R}^d$, $b(s)$ is a function that represents the observed image, and $\eta(s)$ represents additive noise. The kernel, $k(s, t)$, is a function that specifies how the points in the image are distorted, and is therefore called the point spread function (PSF). The inverse problem of image deblurring is: given $k$ and $b$, compute an approximation of $x$. If the kernel has the property that $k(s, t) = k(s - t)$, then the PSF is said to be spatially invariant; otherwise it is said to be spatially variant. In the spatially invariant case, the blurring operation, $\int k(s - t)x(t)dt$, is a convolution operation,
and thus the corresponding inverse problem of computing an approximation of $x$ from $b$ and $k$, is called deconvolution.

In a realistic problem, images are collected only at discrete points (i.e., pixels for 2-D images and voxels for 3-D images), and are also only available in a finite bounded region. It is, therefore, typical to work directly with the discrete linear model,

$$b = Ax + \eta,$$

(1.1)

where $x$, $b$ and $\eta$ are $n \times 1$ vectors obtained by discretizing functions $x$, $b$, and $\eta$, and $n$ is the number of pixels (or voxels) in the discrete image. $A$ is an $n \times n$ matrix that arises when approximating the integration operation with a quadrature rule, and it usually has structure (e.g., Toeplitz, circulant, Hankel, etc.) that can be exploited in computations. The specific structure of $A$ depends, in part, on what kind of boundary conditions are incorporated when transforming a continuous mathematical model (i.e., the integral equation) to the discrete linear model given in equation (1.1). Boundary conditions are needed because pixels near the edges of a blurred image are likely to have been affected by information outside the field of view [7].

Our approach to solving the discrete inverse problem (1.1) is statistically motivated. Specifically, we assume that $\eta$ is an $n \times 1$ independent and identically distributed (iid) Gaussian random vector with variance $\lambda^{-1}$ ($\lambda$ is known as the precision) across all pixels, and that the probability density function for (1.1) is given by

$$p(b|x, \lambda) \propto \exp \left( -\frac{\lambda}{2} \|Ax - b\|^2 \right),$$

(1.2)

where ‘$\propto$’ denotes proportionality. However, it is important to note that when attempting to solve inverse problems, the maximizer of the likelihood $L(x|b, \lambda) = p(b|x, \lambda)$ with respect to $x$, is unstable with respect to the noise contained in $b$. This instability is a characteristic of inverse problems, such as deconvolution, and it has to do with the fact that the forward mapping (convolution) is a compact operator defined on a function space [4]. The standard technique for overcoming such instability is regularization, which is treated in detail in several references [4, 6, 7, 9, 15].

In the context of Bayesian statistics, regularization corresponds to the choice of the prior probability density function $p(x|\delta)$, where $\delta > 0$ is a scaling parameter. Bayes’ Theorem states that given $p(b|x, \lambda)$ and $p(x|\delta)$, the posterior probability density function $p(x|b, \lambda, \delta)$ can be written as

$$p(x|b, \lambda, \delta) \propto p(b|x, \lambda)p(x|\delta).$$

(1.3)

In this paper we use Gaussian Markov random fields (GMRFs) to model the prior, which will have the form

$$p(x|\delta) \propto \exp \left( -\frac{\delta}{2} x^T L x \right),$$

(1.4)

where the precision (inverse-covariance) matrix $\delta L$ is sparse and encodes distributional assumptions regarding the values of $x_i$ conditioned on the values if its neighbors, for all $i$.

Maximizing (1.3) with respect to $x$ is equivalent yields the so-called maximum a posteriori (MAP) estimator, which is also the solution of

$$\arg\min_x \{ \|Ax - b\|^2 + (\delta/\lambda)x^T L x \}.$$
Thus we see that $L$ corresponds to the regularization matrix in classical inverse problems, while $\delta/\lambda$ corresponds to the regularization parameter [15].

When constructing both $A$ and $L$, it is necessary to choose boundary conditions (BCs) for the unknown $x$. BCs correspond to assumptions about the values of the unknown outside of the computational domain. In this paper, our focus is on Neumann BCs and fast computations via the discrete cosine transform (DCT), which has been extensively studied for image deblurring; see, e.g., [7, 13]. For GMRFs, Neumann BCs are mentioned in [10], and fast computations via the DCT are discussed briefly in [11] but are not fully developed. In contrast, the connection between GMRFs with periodic BCs and efficient computations using the discrete Fourier transform is treated extensively in [14].

The main contributions of this paper are as follows: (i) we provide a full development of the use of the DCT for general GMRFs with Neumann BCs; (ii) we establish the important connection between negative-Laplacian Tikhonov regularization, GMRFs, and concrete distributional assumptions (autoregressive models) for $x_i$ conditioned on its neighbors; and (iii) we describe a computationally efficient Markov Chain Monte Carlo (MCMC) method for estimating $x$ and for quantifying uncertainty. Of note is that our MCMC method samples, in addition to $x$, the precision parameters $\lambda$ and $\delta$, which can be used to obtain samples of the regularization parameter $\alpha = \delta/\lambda$. Hence regularization parameter selection is built into the MCMC method.

Finally, we remark that, although the focus of this paper is on image deblurring, the techniques and results we present can also be applied to examples from spatial statistics, in the spirit of [14]. In this setting, $b$ is the response vector and $A$ the design matrix.

The paper is organized as follows. We begin with a discussion of Neumann boundary conditions, both for image deblurring and GMRFs, in Section 2. Then in Section 3, we present an efficient MCMC-based image reconstruction method that makes use of the discrete cosine transform for an efficient implementation. Finally, we show numerical results in Section 4 and end with conclusions.

2. The Neumann boundary condition. In this section, we discuss the Neumann boundary condition, first for convolution problems and then for GMRFs.

2.1. Image deblurring. As has been stated, we are interested in the case in which the image $x$ is assumed to have Neumann, or reflective, boundary conditions. To illustrate what we mean by this, we begin by considering 1D image deconvolution. In this case, the unknown image $x = (x_1, \ldots, x_n)$ can be extended spatially to create the vector

$$\tilde{x} = [x_{-n+1}, \ldots, x_0, x_1, \ldots, x_n, \ldots, x_{2n}]^T.$$ 

The matrix $A$ is defined in terms of the convolution kernel

$$a = [a_{-n}, a_{-n+1}, \ldots, a_0, a_1, \ldots, a_n]^T,$$

and the noise-free response vector is obtained via discrete convolution:

$$b_i = \sum_{j=-n}^{i+n} a_{i-j}x_j, \quad \text{for} \quad i = 1, \ldots, n$$

(2.1)
or in matrix-vector notation,

\[
\begin{bmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_n
\end{bmatrix} = 
\begin{bmatrix}
  a_n & \cdots & a_0 & \cdots & a_{-n} \\
  a_n & \cdots & a_0 & \cdots & a_{-n} \\
  \vdots & \ddots & \vdots & \ddots & \vdots \\
  a_n & \cdots & a_0 & \cdots & a_{-n} \\
  a_n & \cdots & a_0 & \cdots & a_{-n}
\end{bmatrix}
\begin{bmatrix}
  x_{-n+1} \\
  \vdots \\
  x_0 \\
  x_1 \\
  \vdots \\
  x_{n} \\
  \vdots \\
  x_{2n}
\end{bmatrix}.
\] (2.2)

The key observation here is that the response value \( b_i \) for \( i \) near 1 will depend upon the region of \( \bar{x} \) to the left of the computational domain, i.e. \( (x_{-n+1}, \ldots, x_0) \); while for \( i \) near \( n \), \( b_i \) will depend upon the region of \( \bar{x} \) to the right of the computational domain, i.e. \( (x_{n+1}, \ldots, x_{2n}) \).

Rather than estimating these extra values by solving the underdetermined system (2.2), the standard approach is to make assumptions about these values based on a priori knowledge, or by relating the values to those within the computational domain. These assumptions are called boundary conditions. For example, a zero (or Dirichlet) boundary condition corresponds to the assumption that \( (x_{-n+1}, \ldots, x_0) = (x_{n+1}, \ldots, x_{2n}) = 0 \), which yields a Toeplitz matrix \( A \) [15]; while a periodic boundary condition corresponds to \( (x_{-n+1}, \ldots, x_0) = (x_1, \ldots, x_n) \) and \( (x_{n+1}, \ldots, x_{2n}) = (x_1, \ldots, x_n) \), which yields a circulant matrix \( A \) that can be diagonalized by the discrete Fourier transform (DFT) [15].

The Neumann boundary condition corresponds to a reflection of the signal about the boundaries, i.e. \( (x_{-n+1}, \ldots, x_0) = (x_n, \ldots, x_1) \) and \( (x_{n+1}, \ldots, x_{2n}) = (x_n, \ldots, x_1) \). In this case, if the convolution kernel \( a \) is symmetric, i.e. \( a_i = a_{-i} \), the resulting matrix \( A \) has Toeplitz-plus-Hankel structure and can be diagonalized by the discrete cosine transform (DCT) [13, Theorem 3.2]. We note that while a Toeplitz matrix is one for which each descending diagonal from left to right is constant, a Hankel matrix is one for which each descending anti-diagonal from right to left is constant.

We started with the 1D example in order to illustrate concepts more simply, but our primary interest is two-dimensional (2D) image deblurring. In that case, \( b \) and \( x \) are obtained by column-stacking the \( N \times N \) arrays \( B \) and \( X \), which we denote as \( b = \text{vec}(B) \) and \( x = \text{vec}(X) \); and \( A \) is defined in terms of the \( N \times N \) convolution kernel \( a = \{a_{ij}\}_{i,j=1}^{N,N} \), with some assumed boundary condition. The noise-free \( N \times N \) response array then satisfies the 2D discrete convolution equation

\[
b_{r,s} = \sum_{i=-N}^{r+N} \sum_{j=-N}^{s+N} a_{r-i,s-j} x_{ij}, \quad \text{for} \quad r,s = 1, \ldots, N.
\] (2.3)

In the 2D case, for zero and periodic boundary conditions, the extensions of \( X \) are represented, respectively, by

\[
\begin{bmatrix}
  0 & 0 & 0 & X & X & X \\
  0 & X & 0 & & X & X & X \\
  0 & 0 & 0 & & X & X & X
\end{bmatrix}, \quad \text{and} \quad
\begin{bmatrix}
  X & X & X & X & X & X \\
  X & X & X & X & X & X \\
  X & X & X & X & X & X
\end{bmatrix}.
\]

In both cases the central \( X \) corresponds to the unknowns within the computational domain (i.e., the field of view). The assumption of zero boundary conditions results in
a matrix $A$ that is block Toeplitz with Toeplitz blocks \cite{15}, while periodic boundary conditions result in a matrix $A$ that is block circulant with circulant blocks and can be diagonalized by the 2D-DFT \cite{15}.

In instances where the zero and/or periodic extensions are poor approximations of reality, unnatural artifacts in reconstructions can result. This is particularly the case when $X$ and $B$ contain regions of relative high and variable intensity near the boundaries of the computational domain. In such instances, the reflective extension of $X$, corresponding to Neumann boundary conditions, works significantly better. It is represented by

\[
\begin{bmatrix}
X_{vh} & X_h & X_{vh} \\
X_v & X & X_v \\
X_{vh} & X_h & X_{vh}
\end{bmatrix},
\]

where $X_v$ is the image that results from flipping $X$ across its central vertical axis; $X_h$ is the image that results from flipping $X$ across its central horizontal axis; and $X_{vh}$ is the image that results from flipping $X$ across its vertical then horizontal axes. Then, provided the kernel $a$ is symmetric, i.e.

\[a_{i,j} = a_{-i,-j} = a_{i,-j} = a_{-i,-j},\]

the Neumann boundary condition leads to a matrix $A$ that is block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks (BTHTHB) and can be diagonalized by the 2D-DCT \cite{13, Theorem 3.3}. We will use such a BTHTHB matrix $A$ in our numerical experiments below.

2.2. Gaussian Markov Random fields. We now turn to the definition of the prior \cite{1.4}, and specifically, of the precision matrix $\delta L$. For this, we use Gaussian Markov random fields (GMRFs) with Neumann boundary conditions.

A GMRF $x = (x_1, \ldots, x_n)$ is a specific type of Gaussian random vector. Thus $x \sim N(\mu, Q^{-1})$ where $\mu \in \mathbb{R}^n$ is the mean of $x$, and $Q$ is the $n \times n$ symmetric positive definite (SPD) precision (inverse-covariance) matrix. Note that in \cite{1.4}, $\mu = 0$ and $Q = \delta L$.

To define $x$ more specifically, we need the notion of a labeled graph $G = (V, E)$. Here $V = \{1, \ldots, n\}$ is the set of nodes of the graph, and $E$ is the set of edges $\{i, j\}$, where $i, j \in V$ with $i \neq j$. We will use the notation $i \sim j$ to denote that $\{i, j\} \in E$, and in this case we will say that $i$ and $j$ are ‘neighbors’. Moreover, $\partial_i = \{j | j \sim i\}$, $n_i = |\partial_i|$, and $x_{\partial_i} = \{x_j | j \sim i\}$. We can now define a GMRF \cite[Definition 2.1]{14}.

Definition 2.1. A random vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is called a GMRF with respect to a labeled graph $G = (V, E)$ with mean $\mu$ and SPD precision matrix $Q$ if and only if its probability density function has the form

\[
p(x) = (2\pi)^{-n/2} \det(Q)^{1/2} \exp \left( -\frac{1}{2} (x - \mu)^T Q(x - \mu) \right)
\]

where

\[Q_{ij} \neq 0 \iff \{i, j\} \in E, \text{ for all } i \neq j.\]

Note that this implies that every Gaussian random vector with SPD covariance matrix is a GMRF and vice versa. However, where GMRFs become most interesting
is in cases in which the neighborhood sizes $n_i$ are small and $Q$ is sparse. This is illustrated in the following theorem [14, Theorem 2.6].

**Theorem 2.2.** Given the $n$ normal full conditional distributions with conditional mean and precision

$$E(x_i|x_{\partial_i}) = \mu_i - \sum_{j \in \partial_i} \beta_{ij} (x_j - \mu_j),$$

$$\text{Prec}(x_i|x_{\partial_i}) = \kappa_i > 0,$$

then $x$ is a GMRF with respect to $G = (V, E)$ with mean $\mu$ and precision matrix $Q$, where

$$Q_{ij} = \begin{cases} 
\kappa_i, & j = i, \\
\kappa_i \beta_{ij}, & j \in \partial_i \\
0, & \text{otherwise},
\end{cases}$$

provided $\kappa_i \beta_{ij} = \kappa_j \beta_{ji}$, $i \neq j$, and $Q$ is positive definite.

The idea of building a joint density (prior) for $x$ from scalar conditional densities for each $x_i|x_{\partial_i}$, as in this theorem, is known as conditional autoregression, and was pioneered by Besag [2].

If we define $\beta_{ij} = -1/n_i$ for $j \in \partial_i$ and 0 for $j \notin \partial_i$, and $\kappa_i = \delta n_i$, we obtain the conditional densities

$$x_i|x_{\partial_i} \sim \mathcal{N}(\bar{x}_{\partial_i}, (\delta n_i)^{-1}),$$

where $\bar{x}_{\partial_i} = \frac{1}{n} \sum_{j \in \partial_i} x_j$ is the mean of the values in $x_{\partial_i}$. Note that this statistical model intuitively assumes that the value of $x_i$ is ‘near’ to the average of its neighbor values.

From Theorem 2.2, we have that (2.4) yields a Gaussian joint density for $x$ given by

$$p(x|\delta) \propto \delta^{-(n-1)/2} \exp \left( \frac{\delta}{2} x^T L x \right),$$

where

$$|L|_{ij} = \begin{cases} n_i, & i = j, \\
-1, & j \in \partial_i, \\
0, & \text{otherwise},
\end{cases}$$

and the $n - 1$ appears due to the fact that $L$ has rank $n - 1$ [14].

In 1D, we assume a uniform grid on $[0, 1]$ with $n$ vertices $\{1, \ldots, n\}$ at locations $\{s_i\}_{i=1}^n$, where $s_i = i/(n + 1)$. Moreover, we define $x_i$ to be the intensity value at $s_i$ and assume the first-order neighborhood system: $\partial_1 = \{2\}$, $\partial_i = \{i - 1, i + 1\}$ for $i = 2, \ldots, n - 1$, and $\partial_n = \{n - 1\}$. Thus from (2.6) we have

$$L_{1D} = \begin{bmatrix}
1 & -1 & 0 & \cdots & 0 \\
-1 & 2 & -1 & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & -1 & 2 & -1 \\
0 & \cdots & 0 & -1 & 1
\end{bmatrix}_{n \times n}$$

(2.7)
Note that multiplication by $L_{1D}$ is equivalent to discrete convolution (2.1) with kernel $\ell = [0, \ldots, -1, 2, -1, 0, \ldots, 0]^T$, assuming a Neumann boundary condition. Thus, since $\ell$ is symmetric, $L_{1D}$ is a Toeplitz-plus-Hankel matrix that can be diagonalized by the DCT.

In 2D, we assume a uniform grid on $[0, 1] \times [0, 1]$ with $n = N^2$ vertices at locations $\{(s_i, t_j)\}_{i,j=1}^N$, where $s_k = t_k = k/(N + 1)$. Moreover, we define $x_{ij}$ to be the intensity value at $(s_i, t_j)$ for $i, j = 1, \ldots, N$, and assume the first-order neighborhood system:

$$\partial_{ij} = \{(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)\}, \quad \text{for} \quad i, j = 2, \ldots, N - 1,$$

whereas if $i$ or $j$ is 1 or $N$, the vertices containing a 0 or $N + 1$ are removed from $\partial_{ij}$; for example, $\partial_{ij} = \{(2, j), (1, j - 1), (1, j + 1)\}$ and $\partial_{11} = \{(1, 2), (2, 1)\}$. Note then that $n_{ij} = |\partial_{ij}| \in \{2, 3, 4\}$. Moreover, the conditional autoregressive model has the form

$$x_{ij} | x_{\partial_{ij}} \sim N \left( \bar{x}_{\partial_{ij}}, (\delta n_{ij})^{-1} \right), \quad (2.8)$$

where $\bar{x}_{\partial_{ij}} = \frac{1}{n_{ij}} \sum_{(r,s) \in \partial_{ij}} x_{rs}$.

After reordering the array $\{x_{ij}\}_{i,j=1}^N$ by stacking its columns to make the $n \times 1$ vector $x$, i.e. $x = \text{vec}(X)$, we obtain the precision matrix $Q = \delta L_{2D}$, with

$$L_{2D} = L_{1D} \otimes I + I \otimes L_{1D}, \quad (2.9)$$

where $L_{1D}$ is defined in (2.7) and `$\otimes$' denotes the Kronecker product.

Note that multiplication by $L_{2D}$ is equivalent to discrete convolution (2.3) with the $N \times N$ kernel $\ell$ defined by

$$\ell_{ij} = \begin{cases} 4 & (i, j) = (0, 0), \\ -1 & (i, j) \in \{(0, \pm 1), (\pm 1, 0)\}, \\ 0 & \text{otherwise}, \end{cases}$$

assuming a Neumann boundary condition. Thus, since the kernel $\ell$ is symmetric, $L_{2D}$ is block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks (BTHTHB) matrix that can be diagonalized by the 2D-DCT [7, 13].

We summarize the results in this section in the following theorem.

**Theorem 2.3.** The conditional autoregressive models (2.4) in 1D and (2.8) in 2D, with neighborhood systems as described above, yield a prior of the form (2.5), with $L$ defined by (2.7) in 1D and (2.9) in 2D. Moreover, in both cases $L$ is diagonalizable by the discrete cosine transform.

**2.3. Diagonalizing matrices with Toeplitz-plus-Hankel structure.** In both of the above instances, the Neumann boundary condition resulted in a matrix $B$ that was diagonalizable by the DCT. Specifically,

$$B = C^T AC,$$

where $A$ is the $n \times n$ diagonal eigenvalue matrix, and $C$ is the orthogonal DCT matrix. In 1D, $C = C_{1D}$ with

$$[C_{1D}]_{ij} = \sqrt{\frac{2 - \omega_i}{n}} \cos \left( \frac{(i-1)(2j-1)\pi}{2n} \right), \quad \omega_i = \begin{cases} 1 & \text{if } i = 1, \\ 2 & \text{if } i = 2, 3, \ldots, n, \end{cases}$$
where \( 1 \leq i, j \leq n \). In the 2-D case, \( C = C_{2D} \) with \( C_{2D} = C_{1D} \otimes C_{1D} \). We note that both \( C_{1D} \) and \( C_{2D} \) are orthogonal matrices.

In practice, multiplication by \( C \) and \( C^T \) is computed using the fast cosine transform function. In MATLAB, the syntax is as follows: in 1D, for \( v \in \mathbb{R}^n \)
\[
C_{1D}v = \text{dct}(v), \quad C_{1D}^Tv = \text{idct}(v),
\]
while in 2D, if \( V \) is an \( N \times N \) array, and we define \( v = \text{vec}(V) \) and \( V = \text{array}(v) \),
\[
\text{array}(C_{2D}v) = \text{dct2}(V), \quad \text{array}(C_{2D}^Tv) = \text{idct2}(V).
\]

It remains to define the diagonal eigenvalue matrix \( A \). In both cases, \( A = \text{diag}((Ca_1) \otimes (Ce_1)) \), where \( a_1 \) is the first column of \( A \), \( e_1 = [1, 0, \ldots, 0]^T \), ‘\( \otimes \)’ denotes component-wise division, and \( C \) is \( C_{1D} \) in 1D and \( C_{2D} \) in 2D.

3. **MCMC-based Image Reconstruction Method.** In this section, we present a sampling method for image reconstruction assuming the linear model (1.1) (and likelihood (1.2)) with BTHTHB matrix \( A \) and GMRF prior \( p(x|\delta) \) defined by (2.5) with BTHTHB matrix \( L_{2D} \) defined by (2.9), (2.7). We follow the general outline of the method found in [1, 8].

3.1. **The posterior probability density function.** First, as in [8], we assume that \( \lambda \) and \( \delta \) are random variables with Gamma distributed hyper-priors \( p(\lambda) \) and \( p(\delta) \):
\[
p(\lambda) \propto \lambda^{\alpha_\lambda - 1} \exp(-\beta_\lambda \lambda), \quad (3.1)
\]
\[
p(\delta) \propto \delta^{\alpha_\delta - 1} \exp(-\beta_\delta \delta). \quad (3.2)
\]

Following [8], we take \( \alpha_\lambda = \alpha_\delta = 1 \), and \( \beta_\lambda = \beta_\delta = 10^{-4} \) in (3.1) and (3.2). Then the hyper-priors can be deemed to be “uninformative”, since the mean and variance of the corresponding Gamma distributions is \( \alpha/\beta = 10^4 \) and \( \alpha/\beta^2 = 10^8 \), respectively. This is done so that the effect of the hyper-priors on the sampled values for \( \lambda \) and \( \delta \) are negligible. Note that these choices of parameters have required no tuning whatsoever on the variety of examples that we’ve used. Moreover, no other parameters remain to be defined. However, if one has a reasonable \textit{a priori} notion of what \( \lambda \) and/or \( \delta \) should be, the corresponding hyper-prior parameter choices could be modified accordingly.

Finally, with (1.2), (2.5), (3.1), and (3.2) in hand, the posterior probability density can be defined:
\[
p(x, \lambda, \delta|b) \propto p(b|x, \lambda)p(\lambda)p(x|\delta)p(\delta) \quad (3.3)
\]
\[
= \lambda^{n/2 + \alpha_\lambda - 1} \delta^{(n-1)/2 + \alpha_\delta - 1} \exp \left( -\frac{\lambda}{2} \|Ax - b\|^2 - \frac{\delta}{2} x^T Cx - \beta_\lambda \lambda - \beta_\delta \delta \right).
\]

Before presenting our MCMC method for computing samples from (3.3), we draw a connection between (3.3) and classical inverse problems. Note that supposing the parameter \( \lambda \) and \( \delta \) are known the conditional probability density function for \( x|\lambda, \delta, b \) has the form
\[
p(x|\lambda, \delta, b) \propto \exp \left( -\frac{\lambda}{2} \|Ax - b\|^2 - \frac{\delta}{2} x^T Lx \right). \quad (3.4)
\]
Maximizing (3.4) with respect to $\mathbf{x}$ is equivalent to the minimization problem
\[
\min_{\mathbf{x}} \left\{ \| \mathbf{Ax} - \mathbf{b} \|^2 + (\delta/\lambda)\mathbf{x}^T \mathbf{Lx} \right\}.
\]

Thus we see that $\mathbf{L}$ corresponds to the regularization matrix, while $\delta/\lambda$ corresponds to the regularization parameter.

The MCMC method that we present next provides samples not only of $\mathbf{x}$, but also of $\lambda$ and $\delta$, and hence of the regularization parameter $\alpha = \delta/\lambda$. As a consequence, regularization parameter selection is built into the MCMC method.

### 3.2. The MCMC Method.

Following the general approach found in \([5, 8]\), we compute samples from (3.3). Our choice of a Gaussian prior for $\mathbf{x}$ and Gamma hyper-priors for $\lambda$ and $\delta$ was done with efficient sampling in mind. In particular, the prior and hyper-priors are conjugate \([5]\), which guarantees that the full conditional densities have the same form as the corresponding prior/hyper-prior. To see this, note that the full conditional densities have the form (3.4), as well as

\[
p(\lambda|\mathbf{x}, \delta, \mathbf{b}) \propto \lambda^{n/2+\alpha_\lambda-1} \exp \left( -\frac{1}{2} \| \mathbf{Ax} - \mathbf{b} \|^2 - \beta_\lambda \right) \lambda, \tag{3.5} \]
\[
p(\delta|\mathbf{x}, \lambda, \mathbf{b}) \propto \delta^{(n-1)/2+\alpha_\delta-1} \exp \left( -\frac{1}{2} \mathbf{x}^T \mathbf{Lx} - \beta_\delta \right). \tag{3.6} \]

and hence,
\[
\mathbf{x}|\lambda, \delta, \mathbf{b} \sim \mathcal{N} \left( (\lambda \mathbf{A}^T \mathbf{A} + \delta \mathbf{L})^{-1} \lambda \mathbf{A}^T \mathbf{b}, (\lambda \mathbf{A}^T \mathbf{A} + \delta \mathbf{L})^{-1} \right), \tag{3.7} \]
\[
\lambda|\mathbf{x}, \delta, \mathbf{b} \sim \Gamma \left( \frac{n}{2} + \alpha_\lambda, \frac{1}{2} \| \mathbf{Ax} - \mathbf{b} \|^2 + \beta_\lambda \right), \tag{3.8} \]
\[
\delta|\mathbf{x}, \lambda, \mathbf{b} \sim \Gamma \left( (n-1)/2 + \alpha_\delta, \frac{1}{2} \mathbf{x}^T \mathbf{Lx} + \beta_\delta \right), \tag{3.9} \]

where ‘$\mathcal{N}$’ and ‘$\Gamma$’ denote Gaussian and Gamma distributions, respectively.

The power in (3.7)-(3.9) lies in the fact that samples from these three distributions can be computed using standard statistical software. A Gibbsian approach applied to (3.7)-(3.9) yields the following Markov chain Monte Carlo (MCMC) sampling scheme for the posterior density function, which appears in various places in the Bayesian statistics literature, including \([5, 8]\).

#### A MCMC Method for Sampling from $p(\mathbf{x}, \delta, \lambda|\mathbf{b})$.

0. Initialize $\delta_0$, and $\lambda_0$, and set $k = 0$;
1. Compute $\mathbf{x}_k \sim \mathcal{N} \left( (\lambda_k \mathbf{A}^T \mathbf{A} + \delta_k \mathbf{L})^{-1} \lambda_k \mathbf{A}^T \mathbf{b}, (\lambda_k \mathbf{A}^T \mathbf{A} + \delta_k \mathbf{L})^{-1} \right)$;
2. Compute $\lambda_{k+1} \sim \Gamma \left( n/2 + \alpha_\lambda, \frac{1}{2} \| \mathbf{Ax}_k - \mathbf{b} \|^2 + \beta_\lambda \right)$;
3. Compute $\delta_{k+1} \sim \Gamma \left( (n-1)/2 + \alpha_\delta, \frac{1}{2} (\mathbf{x}_k)^T \mathbf{Lx}_k + \beta_\delta \right)$;
4. Set $k = k + 1$ and return to Step 1.

Since the parameters $\lambda$ and $\delta$ are scalar, the scalar random draws required in Steps 2 and 3 are very efficient and easy to compute given the appropriate software. In this paper, we make Step 1 computationally efficient via the use of the 2D-DCT.

Note, particularly, that $\mathbf{x}_k$ can be expressed as follows:
\[
\mathbf{x}_k = (\lambda_k \mathbf{A}^T \mathbf{A} + \delta_k \mathbf{L})^{-1} (\lambda_k \mathbf{A}^T \mathbf{b} + \mathbf{w}), \quad \text{where~} \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \lambda_k \mathbf{A}^T \mathbf{A} + \delta_k \mathbf{L}). \tag{3.10} \]
Given our assumptions above regarding $A$ and $L$, $\lambda_k A^T A + \delta_k L$ can be diagonalized by the 2D-DCT, which makes computing solutions of (3.10) efficient. Specifically, letting $C = C_{2D}$, $A = C^T AC$, and $L = C^T SC$, we have

$$\lambda_k A^T A + \delta_k L = C^T (\lambda_k A^2 + \delta_k S) C.$$  

Substituting this into (3.10) yields

$$x^k = C^T (\lambda_k A^2 + \delta_k S)^{-1} \lambda_k A C b + C^T (\lambda_k A^2 + \delta_k S)^{-1} C w$$

$$= C^T (\lambda_k A^2 + \delta_k S)^{-1/2} ((\lambda_k A^2 + \delta_k S)^{-1/2} \lambda_k A C b + v),$$

where $v \sim N(0, I)$, which follows from the fact that if $B \in \mathbb{R}^{n \times n}$ and $y = B v$, then $y \sim N(0, BB^T)$. Since $A$ and $S$ are computed off-line, and multiplication by $C$ and $C^T$ is computed using the 2D-DCT, the computation of $x^k$ via (3.11) is very efficient.

### 3.3. Assessing MCMC chain convergence

Just as with an iterative method for optimization, a sampling method must be run to convergence. Convergence of an MCMC chain can be determined in a number of ways. The recommended approach presented in [5] requires the computation of multiple, parallel MCMC chains with randomly chosen starting points. With multiple chains in hand, a statistic for each sampled parameter is then computed whose value provides a measure of convergence. This statistic is defined as follows. Suppose we compute $n_r$ parallel chains, each of length $n_s$ (after discarding the first half of the simulations), and that $\{\psi_{ij}\}$, for $i = 1, \ldots, n_s$ and $j = 1, \ldots, n_r$, is the collection of samples of a single parameter. Then we define

$$B = \frac{n_s}{n_r - 1} \sum_{j=1}^{n_r} (\bar{\psi}_j - \bar{\psi})^2,$$

where $\bar{\psi}_j = \frac{1}{n_s} \sum_{i=1}^{n_s} \psi_{ij}$, and $\bar{\psi} = \frac{1}{n_r} \sum_{j=1}^{n_r} \psi_j$;

and

$$W = \frac{1}{n_r} \sum_{j=1}^{n_r} s_j^2,$$

where $s_j^2 = \frac{1}{n_s - 1} \sum_{i=1}^{n_s} (\psi_{ij} - \bar{\psi}_j)^2$.

Note that $\bar{\psi}_j$ and $\bar{\psi}$ are the individual chain mean and overall sample mean, respectively. Thus $B$ provides a measure of the variance between the $n_r$ chains, while $W$ provides a measure of the variance within individual chains.

The marginal posterior variance $\text{var}(\psi | b)$ can then be estimated by

$$\hat{\text{var}}^+(\psi | b) = ((n_s - 1)W + B)/n_s,$$

which is an unbiased estimate under stationarity [5]. The statistic of interest to us, however, is

$$\hat{R} = \sqrt{\frac{\hat{\text{var}}^+(\psi | b)}{W}},$$

(3.12)

which declines to 1 as $n_s \to \infty$.

Once $\hat{R}$ is sufficiently ‘near’ 1 for all sampled parameters, the $n_s n_r$ samples are treated as samples from the target distribution [5]. A value of 1.1 for $\hat{R}$ is deemed acceptable in [5]. In what follows, we stop the MCMC chain once $\hat{R}$ drops below a pre-specified tolerance.
4. Numerical Experiments. In this section, we implement the above MCMC method on an image deconvolution test case. Here the mathematical model is of the form

$$b(s, t) = \int_0^1 \int_0^1 k(s - s', t - t') x(s', t') ds' dt',$$

which we discretize using mid-point quadrature on an $128 \times 128$ uniform computational grid over $[0,1] \times [0,1]$. This yields a system of linear equations $b = Ax$. We assume that $k$ is a circular Gaussian convolution kernel, so that $A$ has BTHTHB structure and can be diagonalized by the 2D-DCT.

The data $b$ is generated using (1.1) with the noise variance $\lambda^{-1}$ chosen so that the noise strength is 2% that of the signal strength. In order to obtain noise-free data that is not corrupted by the Neumann BC assumption, we begin with an extended $256 \times 256$ true image, compute 2D discrete-convolution (2.3) assuming the Neumann BC, and then restrict to the central $128 \times 128$ sub-image to obtain $Ax$. The central $128 \times 128$ region of the image used to generate the data and the data $b$ are shown in Figure 4.1.

Next, we reconstruct the image by sampling from the posterior density function $p(x, \lambda, \delta | b)$ defined in (3.3) using the above MCMC method. We compute 5 parallel MCMC chains and reached an $\bar{R}$ value of 1.03 when the length of the chains was 400, which took approximately 21 seconds. The initial values $\delta_0$ and $\lambda_0$ in Step 0 were chosen randomly from the uniform distributions $U(5, 10)$ and $U(0, 1/2)$, respectively.

We plot the median of the sampled images, with negative values set to zero, as the reconstruction on the upper-left in Figure 4.2. From the samples for $\lambda$ and $\delta$, on the upper-right in Figure 4.2, we plot histograms for $\lambda$, $\delta$, and the regularization parameter $\alpha = \delta / \lambda$, which has a 95% credibility interval $[8.00 \times 10^{-4}, 9.15 \times 10^{-4}]$. Note that the noise precision used to generate the data, $\lambda = 4.72$, is contained within the sample 95% credibility interval for $\lambda$, $[4.61, 4.83]$. And finally, for this example, we also plot the MAP estimator computed with $\alpha$ taken to be the mean of the samples for $\alpha$. As with the sample mean, we set the negative values in the MAP estimator to zero.

It remains to quantify the uncertainty in $x$. First, we plot the standard deviation of the sampled values at each pixel in the lower-right in Figure 4.2; to give the reader
some sense of the variability suggested by these images, we note that for a Gaussian, the 95% confidence interval is approximately two standard deviations either side of the mean. Note that due to the fact that the Neumann boundary condition is only an approximation, boundary artifacts appear in the standard deviation plot, however, these are significantly less pronounced than in the case of periodic boundary conditions.

A more satisfactory approach for visualizing uncertainty in 2D is to create a movie of the image samples. We do this in MATLAB, taking every 10th sample as a frame after the first half of all of the chains have been discarded. The movie accompanies this paper, has the name twinkle1.avi, and can be viewed in Windows Media Player. Another possible approach is to use the computed pixel-wise mean $\mu_{ij}$ and variance $\sigma_{ij}$ from the samples and then let the frames of the movie be samples from $\mathcal{N}(\mu_{ij}, \sigma_{ij})$ for all $ij$. This is the approach taken in [12] and we present the results for our example in the movie twinkle2.avi, noting that since correlation between neighboring intensities is not modeled in this approach, the image appears more variable.
5. Conclusions. Our focus is on the problem of image deconvolution, which is an ill-posed inverse problem, and hence requires regularization. We take a Bayesian approach, in which case the negative-log of the prior probability density function corresponds to the regularization function. We construct our prior by assuming specific Gaussian conditional densities for $x_i|x_\partial$, where $x_\partial$ is the vector containing the ‘neighbor’ intensities $x_j$ of pixel $x_i$. This leads to a Gaussian prior with precision (inverse-covariance) matrix $\delta L$, where $L$ is the discrete negative-Laplacian matrix. This approach is known as conditional autoregression, and the random vector $x$ is called a Gaussian Markov Random Field (GMRF). This interpretation of negative-Laplacian regularization has the benefit that the underlying statistical assumptions are made apparent.

In addition, we assume that the unknown $x$ has Neumann boundary conditions (BCs), which corresponds to extending $x$ outside of the computational domain via a reflection about the boundary. For both convolution (assuming a symmetric kernel) and GMRF models, the resulting matrices $A$ and $L$ have Toeplitz-plus-Hankel structure and can be diagonalized by the discrete cosine transform (DCT). The use of the DCT in the context of GMRFs does not appear to be widespread, even though it has advantages over the oft-used DFT, which corresponds to the assumption of periodic BCs. Specifically, in terms of computational efficiency the DCT and DFT are comparable (both require $O(n \log_2 n)$ flops), while the use of the DCT yields better results when the unknown has relatively high intensity values near the boundary of the computational domain. On the other hand, the DCT requires a symmetric kernel, while the DFT does not.

For the estimation step, we present a Markov Chain Monte Carlo (MCMC) method for sampling from the posterior density function $p(x, \lambda, \delta|b)$. At every MCMC iteration, the primary work is the computation of the image sample $x^k$. This requires the solution of a matrix-vector equation with coefficient matrix $\lambda_k A^T A + \delta_k L$. Since $A$ and $L$ are diagonalizable by the DCT, computing $x^k$ is efficient, and hence, so is the MCMC method. We also present a statistical technique for determining the convergence of the MCMC chain, and test the method on an image deconvolution problem. The method is efficient, requiring only 22 seconds for 5 chains of length 400 each, and yields samples of $x$, $\lambda$, and $\delta$, from which a reconstructed image (sample mean), a pixel-wise variance image, and histograms of $\lambda$, $\delta$ and the regularization parameter $\alpha = \delta/\lambda$ are computed. The results compare favorably with standard MAP estimation.

Finally, we quantify uncertainty in the parameters $\lambda$, $\delta$, and $\alpha = \delta/\lambda$ by plotting sample histograms and, in the case of $\lambda$ and $\alpha$, presenting 95% confidence intervals. For the image $x$, we present both the pixel-wise standard deviation image as well as a movie with frames taken to be a sub-sample of the image samples.

REFERENCES


