Chapter 1

Characteristics of Inverse Problems

The fields of inverse problems and statistics have significant overlap, though their literatures remain largely disjoint. Our goal in this manuscript is to present an introduction to the field of inverse problems with statistical content introduced, and connections between the two fields highlighted, wherever appropriate, in the hopes that this will facilitate the exchange of ideas between these two closely tied fields.

1.1 Preliminaries

We begin with the following linear statistical model with Gaussian noise:

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}, \tag{1.1}$$

where $\mathbf{b}$ is the $m \times 1$ model output, or observation, vector; $\mathbf{x}$ is the $n \times 1$ vector of unknown model parameters; $\mathbf{A}$ is the $m \times n$ forward model matrix, which sometimes requires a separate set of observations; and $\boldsymbol{\epsilon}$ is an $m \times 1$ independent and identically distributed (iid) Gaussian random vector with variance σ^2 across all entries, which we denote $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. We will assume that $m \geq n$ and that $\mathbf{A}$ has full column rank.

In the statistics literature, $\mathbf{b}$ is called the *response vector*, $\mathbf{A}$ the *design matrix*, and $\mathbf{x}$ the *unknown parameter vector*. Moreover, for statistical problems it is typically the case that $m \gg n$ and that the measured data consists both of the response vector $\mathbf{b}$ and of the elements of $\mathbf{A}$, which are called the *explanatory variables*.

The term *inverse problem* is used in many contexts, however, classically, inverse problems are *ill-posed*, which means not *well-posed*, a term first defined by Jacques Hadamard [11]. The definition, adjusted for our notation and situation, is given as follows.

Definition 1.1. *The discrete mathematical model* $\mathbf{b} = \mathbf{A}\mathbf{x}$ *is said to be* well-posed *if*

1. *for all model output* $\mathbf{b}$ *there exists a solution* $\mathbf{x}$ *to* $\mathbf{A}\mathbf{x} = \mathbf{b}$*;*

2. *the solution* $\mathbf{x}$ *is unique; and*

3. *the solution* $\mathbf{x}$ *is stable with respect to perturbations in* $\mathbf{b}$*; that is, if* $\tilde{\mathbf{b}}$ *is a perturbation of* $\mathbf{b}$ *with corresponding solution* $\tilde{\mathbf{x}}$*, then* $\|\mathbf{x}-\tilde{\mathbf{x}}\|/\|\mathbf{x}\|$ *is of the same order of magnitude as, or is smaller than,* $\|\mathbf{b}-\tilde{\mathbf{b}}\|/\|\mathbf{b}\|$.

Condition 3 gives a suitable definition of stability for the examples that we consider in this text, however it may not be a suitable definition for all discrete inverse problems. In the continuous setting, where the focus is instead on the underlying continuously defined mathematical model $b = Ax$, a precise definition of stability can be given. In this case, Hadamard's condition 3 becomes

3. *the solution* x *is stable with respect to perturbations in* b*; that is, the inverse mapping* $b \mapsto x$ *is continuous.*

In an introductory course on linear algebra or statistics, students study linear regression, which often leads to a linear system $\mathbf{b} = \mathbf{A}\mathbf{x}$ that fails condition 1 of Definition 1.1. To see this consider the next example.

Example 1.2 Suppose as a scientist you want to find a formula to estimate the weight b in kilograms of a lion by using its length ℓ in meters. You have five samples: (b, ℓ) = (420,2.4), (350,2.0), (310,2.1), (280,1.8), and (75,1.3). We assume a model of the form

$$b_i = x_1 + x_2\ell_i + \epsilon_i, \tag{1.2}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Gathering the data in matrix form $\mathbf{b} = \mathbf{A}\mathbf{x}$ yields

$$\begin{bmatrix} 420 \\ 350 \\ 310 \\ 280 \\ 75 \end{bmatrix} = \begin{bmatrix} 1 & 2.4 \\ 1 & 2.0 \\ 1 & 2.1 \\ 1 & 1.8 \\ 1 & 1.3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{1.3}$$

We plot this data in Figure 1.1. Visually, you can see that the data has a linear trend, but also that there is no single line that interpolates all data points. Or in other words, (1.3) does not have a solution. This fact can be verified by performing Gaussian elimination on the augmented system $[\mathbf{A}|\mathbf{b}]$. Thus (1.3) is an ill-posed problem because it fails conditions 1 and 2. We can overcome this deficiency by computing the least squares solution, or equivalently by solving the normal equations $\mathbf{A}^T\mathbf{A}\widehat{\mathbf{x}} = \mathbf{A}^T\mathbf{b}$, yielding the best-fit line $b = -303.08 + 307.34\ell$, also plotted with the data in Figure 1.1. We will discuss least squares in more detail in a moment. ∎

The last example would not be considered a classical inverse problem. As was mentioned above, it is typical in inverse problems that the noise free equation $\mathbf{b} = \mathbf{A}\mathbf{x}$ in (1.1) is obtained from the numerical discretization of a continuously defined physical model $b = Ax$, where b and x are functions and A is an operator between function spaces. In many important applications, and in the examples

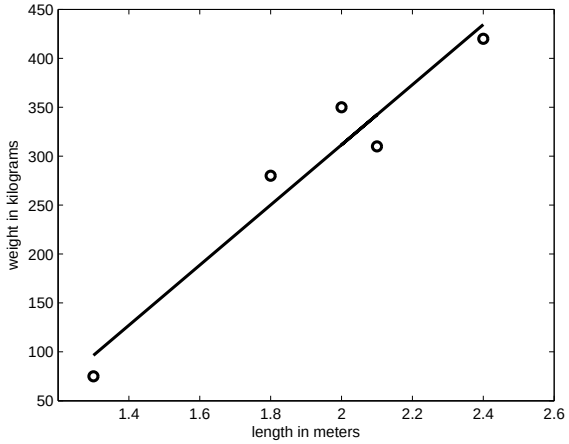

Figure 1.1. *Scatter plot of lion length and weight data* (ℓ_i, b_i) *together with a plot of the best fit line.*

considered in this book, this continuous model takes the form of a Fredholm first kind integral equation

$$b(s) = Ax(s) \stackrel{\text{def}}{=} \int_\Omega a(s,s')x(s')\,ds', \quad s \in \Omega \tag{1.4}$$

where Ω is the domain over which x is defined, and a is known as the *kernel*. In the examples considered in this text, either $\Omega = [0,1]$ or $\Omega = [0,1] \times [0,1]$.

To obtain a matrix-vector equation $\mathbf{b} = \mathbf{A}\mathbf{x}$ from (1.4), we must perform a numerical discretization. We assume $\Omega = [0,1]$ and use the midpoint quadrature rule, which for a general function f defined on $[0,1]$ has the form

$$\int_0^1 f(s')\,ds' = h\sum_{j=1}^{n} f(s'_j) + E_n, \tag{1.5}$$

where $h = 1/n$, $s'_j = (j - \frac{1}{2})/n$, and E_n is the quadrature error. Assuming the same grid for the s variable, i.e. $s_j = s'_j$, if we define $b_i \stackrel{\text{def}}{=} b(s_i)$, $x_j \stackrel{\text{def}}{=} x(s'_j)$, and $a_{ij} = a(s_i, s'_j)$, and apply (3.2) to (1.4), ignoring E_n, we obtain

$$b_i = h\sum_{j=1}^{n} a_{ij}x_j, \quad i = 1,\ldots,m. \tag{1.6}$$

Equation (1.6) can be written in matrix-vector form $\mathbf{b} = \mathbf{A}\mathbf{x}$, where $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{A} \in \mathbb{R}^{m\times n}$ with entries $[\mathbf{A}]_{ij} = a_{ij}$.

For many examples in imaging, $m = n$ and the matrix $\mathbf{A}$ is invertible. However, the resulting model $\mathbf{b} = \mathbf{A}\mathbf{x}$ will fail condition 3 of Definition 1.1, which will be the case for the next example.

Example 1.3 An application that we will repeatedly turn to in this manuscript, and that can be modeled as a Fredholm first kind integral equation, is image deblurring. In this section, we present a one-dimensional (1D) test problem. The ease of computation and visualization in the 1D case makes the illustration of certain concepts more straightforward. We will discuss the two-dimensional image deblurring problem later.

Often in image deblurring problems, the blur is the same everywhere in the image, in which case we say that it is spatially invariant. The assumption of a spatially invariant blur simplifies (1.4) so that it has convolution form

$$b(s) = \int_0^1 a(s - s')x(s')\, ds'. \tag{1.7}$$

Applying the mid-point quadrature rule to (1.7), as above, with kernel function

$$a(s) = \frac{1}{\sqrt{2\pi\gamma^2}} \exp\left(-\frac{s^2}{2\gamma^2}\right), \quad \gamma > 0, \tag{1.8}$$

yields the system of linear equations

$$b_i = h \sum_{j=1}^{n} a_{i-j} x_j, \quad i = 1, \ldots, m, \tag{1.9}$$

where $a_{i-j} = a((i-j)h)$. Compare with (1.6).

As with (1.6), equation (1.9) can be written as $\mathbf{b} = \mathbf{A}\mathbf{x}$ with $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{A} \in \mathbb{R}^{m\times n}$ with entries

$$[\mathbf{A}]_{ij} = \frac{h}{\sqrt{2\pi\gamma^2}} \exp\left(-\frac{((i-j)h)^2}{2\gamma^2}\right). \tag{1.10}$$

From this model, we can generate noisy data using (1.1) with $m = n = 80$ and σ^2 chosen so that the signal-to-noise ratio (SNR), $\|\mathbf{A}\mathbf{x}\|/\sqrt{n\sigma^2}$, is 50. Both $\mathbf{x}$ and $\mathbf{b}$ are plotted on the left in Figure 1.2.

∎

Example 1.4 The relationship between integration and it's inverse, differentiation, also illustrates a failure of condition 3 of Definition 1.1 nicely. The Second Fundamental Theorem of Calculus states that if

$$b(s) = \int_0^s x(s')\, ds', \tag{1.11}$$

then $\frac{d}{ds} b(s) = x(s')$. Unknown to most is that the inverse problem of obtaining x from b defined in (1.11) is ill-posed. To see this, let's proceed as in the last example and discretize (1.11) using midpoint quadrature. We have

$$b_i = \int_0^{s_i} x(s')\, ds' \approx h \sum_{j=1}^{i} x(s'_j), \quad i = 1, \ldots, n.$$

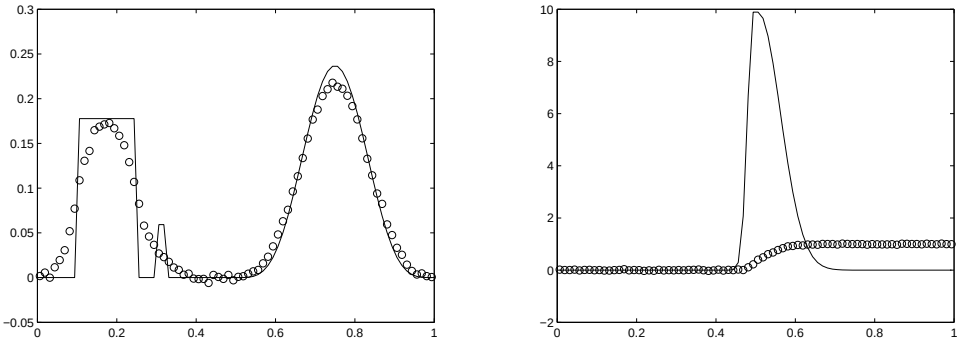

Figure 1.2. *One dimensional deblurring (left) and kernel reconstruction (right) examples. The true image* $\mathbf{x}$ *is given by the solid line and the blurred and noisy data* $\mathbf{b}$ *is given by the circles.*

Letting $\mathbf{x}$ be the $n \times 1$ vector with jth entry $x(s'_j)$ yields $\mathbf{b} = \mathbf{A}\mathbf{x}$ with

$$\mathbf{A} = h \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \cdots & 1 \end{bmatrix}_{n \times n} . \tag{1.12}$$

We will leave it as an exercise to show that

$$\mathbf{A}^{-1} = h^{-1} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \ddots & 0 \\ 0 & -1 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}_{n \times n} . \tag{1.13}$$

Next, let's tie (1.11) to a specific application. Note that in the previous example, the kernel a was assumed to be known. However, in any image deblurring problem involving a kernel, an estimate of the kernel is needed before the inverse problem can be solved. In radiography, the kernel is estimated by passing a step function through the imaging instrument. If a convolution model is assumed, and the step function has the form

$$\chi_t(s') = \begin{cases} 1 & s' > t, \\ 0 & s' \leq t, \end{cases}$$

with $t \in [0, 1]$, then the output has the form (ignoring for a moment the finite

computational domain):

$$\begin{aligned} b(s) &= \int_{-\infty}^{\infty} a(s-s')\chi_t(s')\,ds' \\ &= \int_{t}^{\infty} a(s-s')\,ds' \\ &= \int_{-\infty}^{s} a(s'-t)\,ds' \end{aligned}$$

Restricting to the computational domain [0,1], we use the model

$$b(s) = \int_{0}^{s} a(s'-t)\,ds',$$

which is equivalent to (1.11) with $x(s') = a(s'-t)$.

As a numerical test problem, we synthetically generates a kernel, which defines $\mathbf{x}$, and then, as in the previous example, generated synthetic data $\mathbf{b}$ with an SNR of 50 via (1.1) with $\mathbf{A}$ defined by (1.12). The result is plotted in Figure 1.2. ∎

1.2 The Least Squares Estimator

In the 1D examples above, the goal is to estimate the true image $\mathbf{x}$ given data $\mathbf{b}$ and the model matrix $\mathbf{A}$. First we recall that the probability density function for a general Gaussian random variable $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ is given by

$$p(\mathbf{w}) = \frac{1}{\sqrt{(2\pi)^n \det(\mathbf{C})}} \exp\left(-\frac{1}{2}(\mathbf{v}-\boldsymbol{\mu})^T\mathbf{C}^{-1}(\mathbf{v}-\boldsymbol{\mu})\right). \tag{1.14}$$

Then the statistical model defined by (1.1) implies that

$$p(\mathbf{b}|\mathbf{x},\sigma^2) = \frac{1}{\sqrt{(2\pi)^n\sigma^{2n}}} \exp\left(-\frac{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2}{2\sigma^2}\right). \tag{1.15}$$

A common approach for estimating $\mathbf{x}$ given $\mathbf{b}$ is to maximize the likelihood function, which is defined $L(\mathbf{x}\,|\,\mathbf{b},\sigma^2) = p(\mathbf{b}|\mathbf{x},\sigma^2)$, with respect to $\mathbf{x}$. Equivalently, we can minimize the negative log-likelihood $-\ln L(\mathbf{x}\,|\,\mathbf{b})$, whose minimizer is also the minimizer of

$$\ell(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2, \tag{1.16}$$

and hence is also the least squares solution. The least squares solution can be expressed as the solution of the normal equations

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}, \tag{1.17}$$

a fact whose proof we relegate to the exercises. We assume that $\mathbf{A}$ has full column rank, so that $\mathbf{A}^T\mathbf{A}$ is invertible and the least squares solution is given by

$$\mathbf{x}_{\mathrm{LS}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{1.18}$$

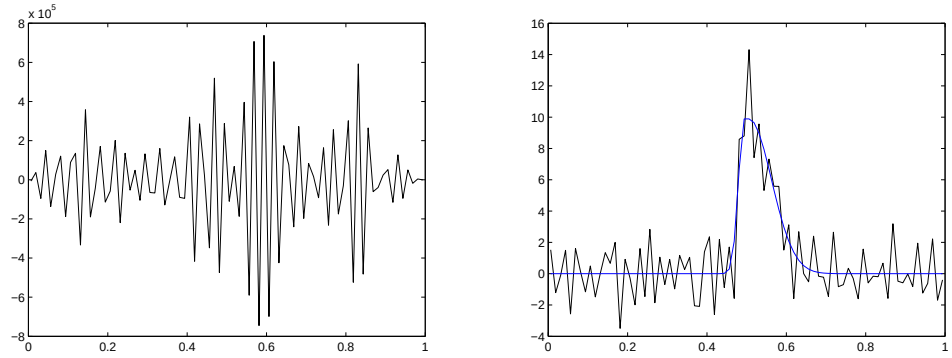

Figure 1.3. *The least squares solution* $\mathbf{x}_{\text{LS}}$ *for the one-dimensional deblurring example.*

Thus if $\mathbf{A}$ is invertible, then $\mathbf{x}_{\text{LS}} = \mathbf{A}^{-1}\mathbf{b}$.

The reader with any experience studying inverse problems will not be surprised that the least squares solutions for the above examples are poor. To illustrate, we plot the least squares solutions in Figure 1.3, and note that in both cases, highly oscillatory components degrade the reconstruction. In addition, for the deblurring example, the oscillatory components completely dominate, making the magnitude of the least squares solution more than 10^7 times larger than for the true image. To see that both examples fail condition 3 of Definition 1.1, in the deblurring case $\|\mathbf{Ax} - \mathbf{b}\|/\|\mathbf{Ax}\| = 0.024$ and $\|\mathbf{x} - \mathbf{x}_{\text{LS}}\|/\|\mathbf{x}\| = 5.25 \times 10^{13}$, while for the PSF reconstruction case $\|\mathbf{Ax} - \mathbf{b}\|/\|\mathbf{Ax}\| = 0.021$ and $\|\mathbf{x} - \mathbf{x}_{\text{LS}}\|/\|\mathbf{x}\| = 0.60$. Given these results, we say that the deblurring example is severely ill-posed, while the PSF reconstruction problem is only mildly ill-posed.

From a statistical viewpoint, we note that (1.1) implies $\mathbf{b} \sim \mathcal{N}(\mathbf{Ax}, \sigma^2\mathbf{I})$. Moreover, for a general $n \times 1$ normal random vector $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ if $\mathbf{w} = \mathbf{Bv}$, where $\mathbf{B}$ is an $m \times n$ matrix, $\mathbf{w} \sim \mathcal{N}(\mathbf{B}\boldsymbol{\mu}, \mathbf{BCB}^T)$. Thus from (1.18) we have

$$\mathbf{x}_{\text{LS}} \sim \mathcal{N}(\mathbf{x}, \sigma^2(\mathbf{A}^T\mathbf{A})^{-1}), \tag{1.19}$$

and hence, while the variance of each pixel of the data $\mathbf{b}$ is σ^2, for the i^{th} component of $\mathbf{x}_{\text{LS}}$ the variance is $[\sigma^2(\mathbf{A}^T\mathbf{A})^{-1}]_{ii}$, which will be large for large i in typical examples in inverse problems. As a result, the least squares solution will typically be far from the true image $\mathbf{x}$, which is again supported by the plots in Figure 1.3.

1.3 The Singular Value Decomposition and Ill-Conditioning

In order to understand why the least squares solutions are so poor, we turn to the singular value decomposition (SVD) of the design matrix:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{1.20}$$

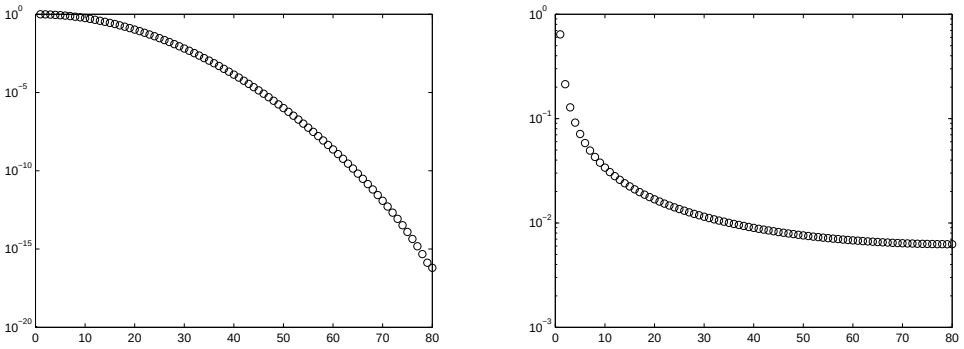

Figure 1.4. *A plot of singular values of* $\mathbf{A}$ *for the one-dimensional deblurring example (left) and the PSF reconstruction example (right). The y-axis is on the* log *scale.*

where $\mathbf{U}$ is an $m \times m$ matrix with columns given by the orthonormal eigenvectors of $\mathbf{A}\mathbf{A}^T$; $\mathbf{V}$ is an $n \times n$ matrix with columns given by the orthonormal eigenvectors of $\mathbf{A}^T\mathbf{A}$; and $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix with diagonal entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, which are called the singular values of $\mathbf{A}$ and are the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$. The columns of $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ are called the left and right singular vectors of $\mathbf{A}$, respectively, and are orthonormal so that $\mathbf{U}^T\mathbf{U} = \mathbf{I}_m$and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_n$.

Although $\mathbf{A}$ may not have an inverse, its pseudo-inverse always exists and is defined

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T, \tag{1.21}$$

where $\mathbf{\Sigma}^\dagger$ is the $n \times m$ matrix with diagonal values $\sigma_1^{-1} \geq \sigma_2^{-1} \geq \cdots \geq \sigma_r^{-1} > 0$, where r is the smallest positive singular value. Detail on the SVD and the pseudo-inverse can be found in [10]. For discretized first kind Fredholm integral equations (called discrete inverse problems in [14]), the singular values decay continuously to 0 as i increases, with a large number clustered near 0. To illustrate, we plot the singular values of $\mathbf{A}$ in both cases in Figure 1.4. Note that the singular values drop to zero much more rapidly in the deblurring example, which partially explains the much worse results for the least squares solutions presented in Figure 1.3. The lower bound of the singular values in the simple integration (kernel reconstruction) example isn't much below the grid size h.

We can also say something about typical behavior of the left and right singular vectors, $\mathbf{u}_i$ and $\mathbf{v}_i$, respectively, for discretized first kind Fredholm integral equations; namely, they behave like the Fourier basis in the sense that as i increases, $\mathbf{u}_i$ and $\mathbf{v}_i$ become more oscillatory. This fact is demonstrated for several examples in [14]. We leave its verification for the 1D deblurring example considered above to the exercises.

1.3.1 Statistical Properties of the Least Squares Estimator

We now return to the least squares solution and view it in light of the above discussion. Given our assumptions regarding $\mathbf{A}$, it can be shown (see Exercise 4) that

$$\mathbf{x}_{\text{LS}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \tag{1.22}$$

$$= \mathbf{x} + \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \boldsymbol{\epsilon}}{\sigma_i} \mathbf{v}_i. \tag{1.23}$$

From (1.23) and the above discussion, we see that due to the spectral properties of a typical noise realization $\boldsymbol{\epsilon}$, $(\mathbf{v}_i^T \boldsymbol{\epsilon})/\sigma_i$ will be large for large i, which serves to amplify the more oscillatory components in the second term in (1.23). This coincides with what we observe in the least squares solutions in Figure 1.3, where high frequency components dominate.

A more detailed analysis of the interplay between the noise $\boldsymbol{\epsilon}$, the data $\mathbf{b}$, and the spectral properties of $\mathbf{A}$, through its SVD, is possible using the discrete Picard condition; the interested reader should see [14] for details, as well as the exercises for a Picard analysis of the deblurring example considered in this chapter.

We can also use the SVD to shed further light on the distributional statement (1.19) for $\mathbf{x}_{\text{LS}}$. Replacing $\mathbf{A}$ with its the SVD in (1.19) yields

$$\mathbf{x}_{\text{LS}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{V} \boldsymbol{\Sigma}^{-2} \mathbf{V}^T), \tag{1.24}$$

where $\boldsymbol{\Sigma}^{-2} = (\boldsymbol{\Sigma}^T \boldsymbol{\Sigma})^{-1}$.

From (1.24), we can see that $E[\mathbf{x}_{\text{LS}}] = \mathbf{x}$, and hence the least squares estimator is *unbiased*. However, in the orthonormal basis defined by the columns of $\mathbf{V}$ (the right singular vectors), (1.24) can be restated in a component-wise fashion as follows:

$$\mathbf{v}_i^T \mathbf{x}_{\text{LS}} \sim \mathcal{N}\left(\mathbf{v}_i^T \mathbf{x}, \sigma^2/\sigma_i^2\right), \quad i = 1, \ldots, n, \tag{1.25}$$

which directly shows that for $0 < \sigma_i^2 \ll \sigma^2$ (i large), one can expect high variability in the corresponding oscillatory mode $\mathbf{v}_i$. Thus although the least squares estimator is unbiased, its *total variance*, which is given by

$$\text{tr}(\sigma^2 \mathbf{V} \boldsymbol{\Sigma}^{-2} \mathbf{V}^T) = \sigma^2 \sum_{i=1}^{n} 1/\sigma_i^2, \tag{1.26}$$

where 'tr' denotes the matrix trace, will be extremely large, which is undesirable.

Example 1.5 In order to make the above concepts more concrete, we consider a two dimensional example. First we define the matrix $\mathbf{A}$ via its SVD: let $\mathbf{v}_1 = [1/\sqrt{2}, 1/\sqrt{2}]^T$, $\mathbf{v}_2 = [-1/\sqrt{2}, 1/\sqrt{2}]^T$, and

$$\mathbf{A} = \mathbf{v}_1 \mathbf{v}_1^T + 10^{-2} \mathbf{v}_2 \mathbf{v}_2^T$$

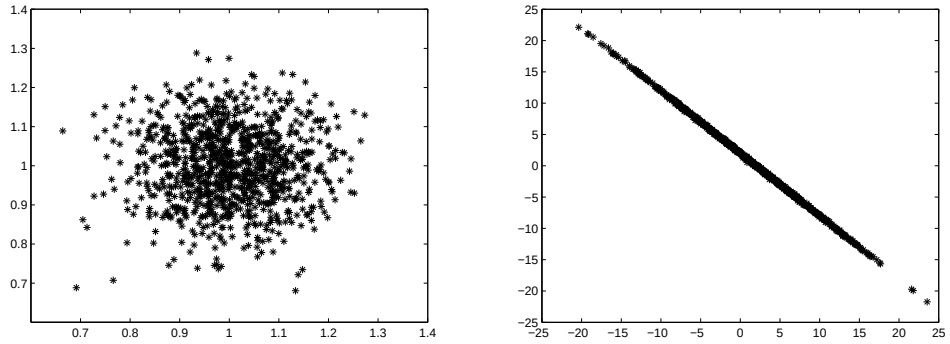

Figure 1.5. *On the left are 1000 realizations from the data model* $\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$*, and on the right are plots of the corresponding 1000 least squares solutions* $\mathbf{A}^{-1}\mathbf{b}$*. Both the true solution* $\mathbf{x}$ *and the noise-free data* $\mathbf{A}\mathbf{x}$ *are* $[1,1]^T$*.*

Note that we have used the summation form of the SVD (see Exercise 1.3). Hence $\mathbf{v}_1$ and $\mathbf{v}_2$ are both the left and right singular vectors and the singular values are $\sigma_1 = 1$ and $\sigma_2 = 10^{-2}$. Furthermore, as in the above cases, the smaller singular value corresponds to the 'higher frequency' vector.

Now suppose $\mathbf{x} = [1,1]^T$, then $\mathbf{b} = \mathbf{A}\mathbf{x} = [1,1]^T$ and $\mathbf{A}^{-1}\mathbf{b} = [1,1]^T$. However, if we add a small amount of noise to $\mathbf{b}$ using (1.1) with $\boldsymbol{\epsilon} = [0.026, 0.075]^T$ (a truncated realization of an iid Gaussian distribution with variance $\sigma^2 = 0.01$) then $\mathbf{A}^{-1}\mathbf{b} = [-1.400, 3.501]^T$, which is far from $[1,1]^T$.

Let us use the SVD discussion above to explore this problem more deeply. First, we plot 1000 realizations from the statistical model $\mathbf{b} = \mathbf{A}\mathbf{x}+\boldsymbol{\epsilon}$ with $\sigma^2 = 0.01$ on the left in Figure 1.5. Note that the $\mathbf{b}$ samples have a relatively small variance centered around $[1,1]^T$. Next, on the right in Figure 1.5, we plot $\mathbf{x}_{\text{LS}}$ for each realization of the data. Note that the noise free solution $[1,1]^T$ is contained within the data cloud, but that the variance is high in the direction of $\mathbf{v}_2$, while in the direction of $\mathbf{v}_1$ it is relatively small. This coincides with (1.25), which says that the variance of $\mathbf{x}_{\text{LS}}$ in the direction $\mathbf{v}_1$ is $\sigma^2/\sigma_1^2 = 0.01$, while in the direction $\mathbf{v}_2$ it is $\sigma^2/\sigma_2^2 = 100$. We computationally verified this for our experiment by computing the sample variance of $\mathbf{v}_1^T\mathbf{x}_{\text{LS}}$ and $\mathbf{v}_2^T\mathbf{x}_{\text{LS}}$ for all least squares solutions, which were 0.01 and 99.73, respectively. Additionally, note that

$$\mathbf{A} = \begin{bmatrix} 0.505 & 0.495 \\ 0.495 & 0.505 \end{bmatrix},$$

which, though non-singular, has columns that are 'nearly' linearly dependent, both well-approximating the singular vector $\mathbf{v}_1$. This agrees with our observations above since variance values for $\mathbf{v}_1^T\mathbf{x}_{\text{LS}}$ and $\mathbf{v}_2^T\mathbf{x}_{\text{LS}}$ of 0.01 and 100, respectively, suggest that the least squares solution is well-resolved in the direction of $\mathbf{v}_1$ and poorly resolved in the direction of $\mathbf{v}_2$. ∎

Exercises

1.1. *This simple linear regression problem was written in the fall of 2011 by former University of Montana Math PhD students (and now PhD's) John Hossler, Marylesa Howard, and Jordan Purdy*: In baseball two commonly collected statistics for pitchers are batting average against (AVG) and walks plus hits per inning pitched (WHIP). Suppose we believe that a linear relationship exists between AVG and WHIP, and in particular, we believe we can predict a pitcher's WHIP from their AVG. AVG and WHIP data for 30 Major League Baseball (MLB) pitchers from the 2011 regular season (through September 2) are provided in `BaseballData.m`. Use this data to answer the following questions. You may find it easiest to modify `SimpleLinearModel.m` used for Example 1.2.

(a) Plot the batting average (AVG) against corresponding walks plus hits per inning pitched (WHIP) such that AVG predicts WHIP (AVG on x-axis, WHIP on y-axis).

(b) Write out by hand the first four rows of the design matrix. Construct the entire design matrix in MATLAB. Include an intercept in your model.

(c) Compute the least squares estimate. Note that $\boldsymbol{x}_{LS}$ is a 2×1 vector.

(d) Using the model created in part (c), plot the model on the data plot from part (a). This model predicts a pitcher's WHIP given his batting average against.

(e) For each observed data point, the residual is that observed value (WHIP) minus the corresponding predicted value (predicted WHIP) from the model. Calculate the thirty residuals and plot them against their corresponding AVG value. On the same graph, plot the line $y = 0$. Comment on whether a pattern is distinguishable or whether there is random scatter in the residual plot. A pattern in the residuals may suggest a different model or a linear model with different error structure is more appropriate, for this data set. While random scatter does not verify our choice of this linear model is correct, it does suggest this model may be useful for analysis.

1.2. The *midpoint quadrature rule* for a function f defined on $[a, b]$ is defined by

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) + \frac{f''(\eta)}{24}(b-a)^3, \tag{1.27}$$

where $a \leq \eta \leq b$.

a. Write your own MATLAB code applying midpoint quadrature with $n = 100$ grid points to approximate $\int_0^{2\pi} \cos x \, dx = 1$.

b. Determine an upper bound for the quadrature error in part (a) using (1.27) (n times) and then verify that the inequality holds.

c. Modify `Deblur1d.m` so that it implements midpoint quadrature on the convolution model (1.7) with kernel

$$a(s) = \begin{cases} 100s + 10, & -\frac{1}{10} \le s \le 0, \\ -100s + 10, & 0 \le s \le \frac{1}{10}, \\ 0, & \text{otherwise.} \end{cases}$$

to obtain $\mathbf{b} = \mathbf{A}\mathbf{x}$. Plot a on $[-1, 1]$. Is the deblurring problem ill-posed with this kernel? Support your answer.

1.3. *More on the simple integration/kernel reconstruction inverse problem.*

a. Show that $\mathbf{A}$ defined by (1.12) has inverse $\mathbf{A}^{-1}$ defined by (1.13). Use induction on n.

b. Show that $\mathbf{A}^{-1}$ corresponds to the discretization of $\frac{d}{ds}$ on the grid $\{s'_j\}_{j=1}^n$, where recall that $s'_j = (j - \frac{1}{2})/n$, with a zero Dirichlet boundary condition on the left (i.e., $x(s'_0) = 0$) and a zero Neumann boundary condition on the right (i.e., $x(s'_{n+1}) = x(s'_n)$). Use the forward difference approximation of the derivative:

$$\frac{d}{ds}x(s'_j) \approx \frac{x(s'_j) - x(s'_{j-1})}{h}.$$

c. Replace $\mathbf{A}$ by $\mathbf{A}^{-1}$ in `PSFrecon.m`. How do the results change? Is $\mathbf{b} = \mathbf{A}^{-1}\mathbf{x}$ an ill-posed problem, i.e. does it fail any of the three conditions of Definition 1.1?

1.4. *The Singular Value Decomposition.* Let $\mathbf{A}$ be $m \times n$, with $m \ge n$, and suppose it has singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Let $\mathbf{u}_i$ be the i^{th} column of $\mathbf{U}$, $\mathbf{v}_i$ be the i column of $\mathbf{V}$, and σ_i the i diagonal element of $\mathbf{\Sigma}$.

a. If the rank of $\mathbf{A}$ is $r \le n$, show that

$$\mathbf{A}\mathbf{v}_i = \begin{cases} \sigma_i \mathbf{u}_i, & i = 1, \ldots, r \\ \mathbf{0}, & i = r+1, \ldots, n, \end{cases} \quad \mathbf{A}^T\mathbf{u}_i = \begin{cases} \sigma_i \mathbf{v}_i, & i = 1, \ldots, r \\ \mathbf{0}, & i = r+1, \ldots, m. \end{cases}$$

b. Show that (1.20) can be written $\mathbf{A} = \sum_{i=1}^n \mathbf{u}_i \sigma_i \mathbf{v}_i^T$ and write down the equivalent expression for $\mathbf{A}^\dagger$ defined in (1.21).

c. Substitute (1.20) into (1.18) to prove (1.22) and hence that $\mathbf{x}_{\text{LS}} = \mathbf{A}^\dagger \mathbf{b}$.

1.5. *The Singular Value Decomposition.* Define $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.

a. Compute the singular value decomposition of $\mathbf{A}$ by hand: $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{3\times 3}$ is orthogonal, $\mathbf{S} \in \mathbb{R}^{3\times 2}$ is diagonal, and $\mathbf{V} \in \mathbb{R}^{2\times 2}$ is orthogonal.

b. Using the columns of $\mathbf{U}$ and $\mathbf{V}$ as basis elements, write down bases for the four subspaces: the column space $C(\mathbf{A})$, the row space $C(\mathbf{A}^T)$, the null space $N(\mathbf{A})$, and the left null space $N(\mathbf{A}^T)$.

c. Using the SVD you computed in (a), compute the pseudo-inverse $\mathbf{A}^\dagger$ of $\mathbf{A}$, and then use it to compute the least squares solution $\mathbf{x}_{\text{LS}} = \mathbf{A}^\dagger\mathbf{b}$ of

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

1. Compare the solution in (c) with the solution of the normal equations, $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$. The two solutions should agree.

1.6. *The discrete Picard condition.* For parts (a)-(d), use `Deblur1d.m`.

a. Create plots verifying that the singular vectors of the design matrix $\mathbf{A}$ become more oscillatory as i increases. Plot, for example, right singular vectors 1-8.

b. A Picard plot for a discretized inverse problem is a plot of the values of σ_i, $|\mathbf{u}_i^T\mathbf{b}|$, and $|\mathbf{u}_i^T\mathbf{b}|/\sigma_i$. Picard plots are used extensively in [14] for analyzing discrete inverse problems. For the 1D deblurring example, create the Picard plot with a legend, first using the noise-free data and then using noisy data. Verify in both cases the *discrete Picard condition*: Ignoring the part of the Picard plot where $|\mathbf{u}_i^T\mathbf{b}|$ levels off due either to numerical round-off (in the noise-free case) or to the presence of noise in $\mathbf{b}$, the discrete Picard condition is satisfied if the remaining values of $|\mathbf{u}_i^T\mathbf{b}|$, on average, decay faster than σ_i, i.e. $|\mathbf{u}_i^T\mathbf{b}|/\sigma_i$ is decreasing. Motivate the discrete Picard condition by appealing to (1.22).

c. Plot the variance values in (1.25). What do they tell you about the statistical properties of the least squares solution.

d. Show that even in the absence of noise, the least squares solution for the deblurring example remains poor. This is due to the fact that the deblurring example is so ill-conditioned that even round-off errors, which are on the order of machine precision (approximately 10^{-16}), are amplified.

e. Repeat a-c using `PSFrecon.m`.

1.7. *Multivariate Gaussian distribution.*

a. Suppose $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is an independent and identically distributed standard Normal random vector, i.e. $v_i \sim \mathcal{N}(0,1)$ for $i = 1, \ldots, n$. Given $p(v_i) = 1/\sqrt{2\pi}\exp\left(-v_i^2/2\right)$, show that

$$p_{\mathbf{v}}(\mathbf{v}) = \frac{1}{\sqrt{(2\pi)^n}}\exp\left(-\frac{1}{2}\|\mathbf{v}\|^2\right)$$

b. Let $\mathbf{w} = \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$. Note that $\mathbf{w} = \boldsymbol{\mu} + \mathbf{C}^{1/2}\mathbf{v}$, where $\mathbf{v}$ is as in (a). Then the inverse transformation yields $\mathbf{v} = \mathbf{C}^{-1/2}(\mathbf{w} - \boldsymbol{\mu})$ and

$$p(\mathbf{w}) = p_{\mathbf{v}}(\mathbf{C}^{-1/2}(\mathbf{w} - \boldsymbol{\mu})) \cdot \det(\mathbf{C}^{-1/2}).$$

Use this to derive the multivariate Gaussian probability density function (1.14).

c. Use (b) to deriving the probability density for $p(\mathbf{b}|\mathbf{x}, \sigma^2)$ in (1.15).

1.8. *Deriving the least squares normal equations.*

a. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its Hessian $\nabla^2 f(\mathbf{x})$ is positive semi-definite for all $\mathbf{x}$ and is strictly convex if $\nabla^2 f(\mathbf{x})$ is positive definite for all $\mathbf{x}$. Moreover, the set of minimizers of a convex function f is the set on which the gradient of f is zero, i.e., those vectors $\mathbf{x}$ satisfying $\nabla f(\mathbf{x}) = \mathbf{0}$, and this set has a single member (and hence f a unique minimizer) if f is strictly convex. Use these facts to show that the minimizers of $\ell(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$, with respect to $\mathbf{x}$, are the solutions of (1.17) and derive conditions on $\mathbf{A}$ that guarantee that $\ell(\mathbf{x})$ has a unique minimizer.

b. Show that the projection of $\mathbf{b}$ onto the column space of $\mathbf{A}$, which we assume has full column rank, has the form $\mathbf{A}\mathbf{x}_{\text{LS}}$, where $\mathbf{x}_{\text{LS}}$ is the least squares solution. Note that the projection formula can be found in an elementary linear algebra text.

1.9. *Sampling the least squares solution.*

a. Prove (1.19) using the following result: if $\mathbf{B}$ is an $m \times n$ matrix, $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, and $\mathbf{w} = \mathbf{B}\mathbf{v}$, then $\mathbf{w} \sim \mathcal{N}(\mathbf{B}\boldsymbol{\mu}, \mathbf{B}\mathbf{C}\mathbf{B}^T)$.

b. In the m-file `TwoVarTest.m`, which was used to generate Figure 1.5, samples of $\mathbf{x}_{\text{LS}}$ are computed by first sampling $\mathbf{b} \sim \mathcal{N}(\mathbf{A}\mathbf{x}, \sigma^2\mathbf{I})$ and then computing the corresponding least squares solution $\mathbf{x}_{\text{LS}}$ via (1.18). Add a line of code that also samples $\mathbf{x}_{\text{LS}}$ using the approach in part (a), then plot the two collections of samples together in different color to verify that the sample distributions look roughly the same. Also, compare their sample mean and covariance matrices using MATLAB's `mean` and `cov` functions.

c. Create normalized histograms (using the `hist` function) from the samples of $\mathbf{v}_1^T\mathbf{x}_{\text{LS}}$ and $\mathbf{v}_2^T\mathbf{x}_{\text{LS}}$ computed in `TwoVarTest.m` and then graph the resulting sample probability densities together with the analytic true normal densities given by (1.25).

Chapter 2

Regularization by Filtering

In the last chapter, we introduced two examples, both obtained by numerically discretizing an integral equation, that characterize ill-posed inverse problems. We showed that for these problems the standard least squares solution is poor and, through an SVD analysis, why this is the case.

In this chapter, we present several methods which modify the least squares solution in such a way that the new solutions provide a significant improvement. Specifically, these methods filter out the singular vectors with low magnitude in the noise-free signal $\mathbf{Ax}$, but which lead to large variance in the least squares solution. These techniques are known as *regularization methods* and we interpret them as a 'filtered' SVD, or spectral filtering method.

The filters that we introduce are dependent upon a single unknown parameter, known as the *regularization parameter*. Hence the practical usefulness of the regularization methods depends upon there being effective techniques for estimating the regularization parameter. Thus we also present several *regularization parameter choice methods*.

2.1 Spectral Filtering Methods

To motivate our discussion, we return to Example 1.5, where we considered the problem of estimating $\mathbf{x} = [x_1, x_2]^T$ from data $\mathbf{b} = [b_1, b_2]^T$ generated by the model

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0.505 & 0.495 \\ 0.495 & 0.505 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix},$$

where $\epsilon_1, \epsilon_2 \overset{iid}{\sim} \mathcal{N}(0, 0.01)$.

Recall our example: suppose $\mathbf{x} = [1, 1]^T$, then in the noise-free case, $\mathbf{b} = \mathbf{Ax} = [1, 1]^T$ and $\mathbf{A}^{-1}\mathbf{b} = [1, 1]^T$, while if $\mathbf{b} = \mathbf{Ax} + \boldsymbol{\epsilon} = [1.026, 1.075]^T$, then $\mathbf{A}^{-1}\mathbf{b} = [-1.400, 3.501]^T$. Thus after adding a small amount of noise, the least squares solution is far from the solution $[1, 1]^T$ in the noise-free case. From our discussion in the previous chapter, this is due to high variance in the solution in the direction of the singular vector $\mathbf{v}_2 = [-1/\sqrt{2}, 1/\sqrt{2}]$.

Given this, a natural idea is to remove (or filter) the singular vector $\mathbf{v}_2$ from the model, so that instead $\mathbf{A}$ is replaced by

$$\mathbf{A}_{\text{filt}} = \sigma_1 \mathbf{v}_1 \mathbf{v}_1^T = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}.$$

The filtered least squares solution for the data $\mathbf{b} = [1.026, 1.075]^T$ is then given by

$$\mathbf{x}_{\text{filt}} = \mathbf{A}_{\text{filt}}^{\dagger}\mathbf{b} = \sigma_1^{-1}(\mathbf{v}_1^T\mathbf{b})\mathbf{v}_1 = \begin{bmatrix} 1.0505 \\ 1.0505 \end{bmatrix},$$

where '†' denotes pseudo-inverse [10], which is much nearer to the noise free solution $[1, 1]^T$ than $\mathbf{A}^{-1}\mathbf{b}$.

2.1.1 The Truncated Singular Value Decomposition

The approach of removing the singular vectors from the model that lead to large variance in the least squares solution, as we did in the previous example, can be generalized to higher dimensional problems. For the problems of interest to us, if

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{n} \mathbf{u}_i \sigma_i \mathbf{v}_i^T, \tag{2.1}$$

the singular vectors that lead to high variance in the least squares solution correspond to the larger values of i. Removing the singular vectors for $i \geq k$ from the model amounts to replacing $\mathbf{\Sigma}$ in (2.1) by the $m \times n$ diagonal matrix $\mathbf{\Sigma}_k$ with diagonal entries $(\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0)$, yielding

$$\mathbf{U}\mathbf{\Sigma}_k\mathbf{V}^T = \sum_{i=1}^{k} \mathbf{u}_i \sigma_i \mathbf{v}_i^T.$$

This is known as the truncated singular value decomposition of $\mathbf{A}$ and it leads to the *TSVD regularized solution*

$$\mathbf{x}_k = \mathbf{V}\mathbf{\Sigma}_k^{\dagger}\mathbf{U}^T\mathbf{b} = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i, \tag{2.2}$$

where '†' denotes pseudo-inverse. Compare (2.2) with the analogous expression for the least squares solution (1.22).

The TSVD appears in many of the standard texts on inverse problems; see e.g., [13, 14, 15, 17, 28], and in the statistical literature, it is known as *principal component regression* [21]. The parameter k is known as the *regularization parameter*, and the choice of its value is a nontrivial task that we address in Section 2.2. Nonetheless, to illustrate the effectiveness of the TSVD, we plot $\mathbf{x}_k$ for $k = 24$ in the deblurring case and $k = 20$ in the kernel reconstruction case in Figure 2.1, which yields a good visual reconstruction of $\mathbf{x}$.

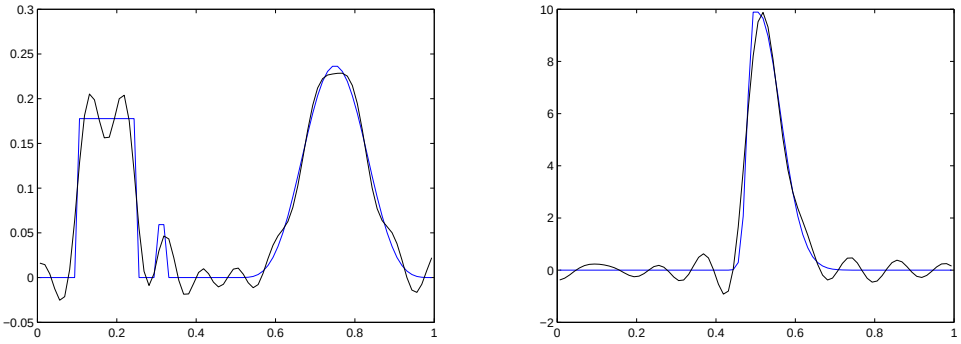

Figure 2.1. *A plot of the TSVD reconstruction* $\mathbf{x}_k$ *for* $k = 24$ *in the deblurring case (left) and for* $k = 20$ *in the kernel reconstruction case (right) together with the true image.*

The TSVD is an example of the general class of *spectral filtering methods*, which have the filtered SVD form

$$\mathbf{x}_\nu = \mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b} \tag{2.3}$$

where $\mathbf{\Phi}_\nu$ is an $n \times n$ diagonal matrix with diagonal entries $[\mathbf{\Phi}_\nu]_{ii} = \phi_i^{(\nu)}$ for $i = 1, \ldots, n$. The $\phi_i^{(\nu)}$'s are known as *filter factors*, are parameterized by the variable ν, and are chosen so that $\phi_i^{(\nu)} \approx 1$ for large singular values and $\phi_i^{(\nu)} \approx 0$ for small singular values.

For the TSVD, $\nu = k$ in (2.3) and

$$\phi_i^{(k)} = \begin{cases} 1 & i = 1, \ldots, k \\ 0 & i = k+1, \ldots, n. \end{cases}$$

We define $\mathbf{\Phi}_k = \mathrm{diag}(\phi_1^{(k)}, \ldots, \phi_n^{(k)})$, then (2.3) takes the form

$$\mathbf{x}_k = \mathbf{V}\mathbf{\Phi}_k\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b}, \tag{2.4}$$

which is equivalent to (2.2).

2.1.2 Tikhonov Regularization

Tikhonov regularization is probably the most commonly used form of regularization, and it appears in every text on the subject of inverse problems. These include [1, 4, 8, 13, 14, 15, 17, 28], as well as many others. The Tikhonov regularized solution is defined by

$$\mathbf{x}_\alpha = \arg\min_{\mathbf{x}} \left\{ \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \frac{\alpha}{2}\|\mathbf{x}\|^2 \right\}, \tag{2.5}$$

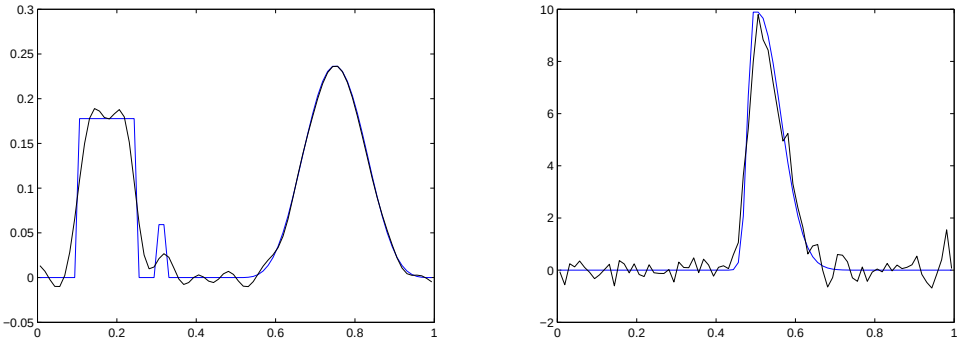

Figure 2.2. *A plot of the Tikhonov reconstruction* $\mathbf{x}_\alpha$ *for* $\alpha = 0.005$ *in the deblurring case (left) and* $\alpha = 0.0002$ *in the kernel reconstruction case (right), together with the true image.*

which can be equivalently expressed as the solution of the pseudo-normal equations for (2.5):

$$(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I})\mathbf{x} = \mathbf{A}^T\mathbf{b}. \tag{2.6}$$

Comparing (2.6) with the least squares normal equations (1.17), we see that the addition of the penalty term $\frac{\alpha}{2}\|\mathbf{x}\|^2$ in (2.5) has the effect of uniformly increasing the eigenvalues of the coefficient matrix; namely from σ_i^2 for $\mathbf{A}^T\mathbf{A}$ in (1.17) to $\sigma_i^2+\alpha$ for $\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}$ in (2.6). This makes the computation of $\mathbf{x}_\alpha$ more stable that the computation of the least squares solution. Statisticians might recognize (2.5), (2.6) as *ridge regression* [20].

Tikhonov regularization can also be written as a spectral filtering method with filter factors

$$\phi_i^{(\alpha)} = \sigma_i^2 / (\sigma_i^2 + \alpha).$$

Hence, $\mathbf{x}_\alpha$ can be written in the form (2.3) with $\nu = \alpha$ and

$$\mathbf{\Phi}_\alpha = \operatorname{diag}\left(\frac{\sigma_1^2}{\sigma_1^2+\alpha}, \ldots, \frac{\sigma_n^2}{\sigma_n^2+\alpha}\right). \tag{2.7}$$

The proof of the equivalence of (2.5), (2.6), and (2.3) with $\nu = \alpha$ and (2.7) is left to the exercises.

As was the case for the TSVD, we choose a value of the regularization parameter α that yields a good visual reconstruction of $\mathbf{x}$. For $\alpha = 0.005$ in the deblurring case and $\alpha = 0.0002$ in the kernel reconstruction case, the corresponding Tikhonov regularized solution $\mathbf{x}_\alpha$ is plotted in Figure 2.2. Less subjective methods for choosing α are presented in Section 2.2.

2.1.3 Iterative Regularization

For inverse problems, iterative methods applied to the problem of minimizing $\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2$ often exhibit *semi-convergence*; that is, if $\mathbf{x}_k$ is the k^{th} iteration of the algorithm

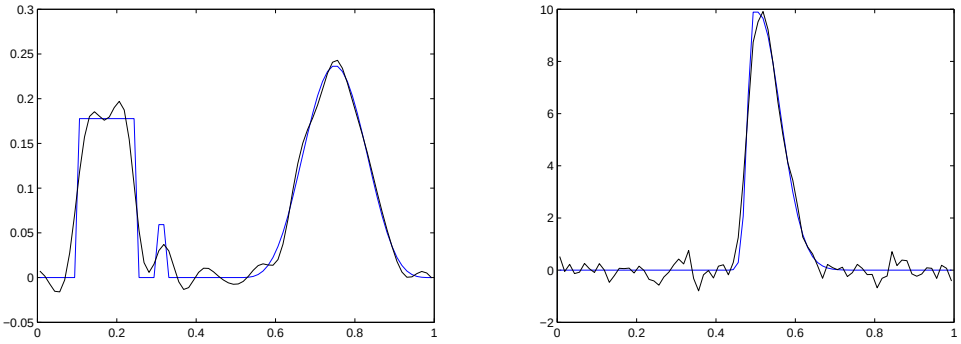

Figure 2.3. *The Landweber reconstruction* $\mathbf{x}_k$ *is plotted, together with the true image, on the left in the deblurring case with* $k = 300$*, and on the right in the kernel reconstruction case with* $k = 3000$*.*

– not to be confused with $\mathbf{x}_k$ defined by the TSVD – in early iterations $\|\mathbf{x}_k - \mathbf{x}\|^2$ decreases, but then eventually begins to increase as $\mathbf{x}_k \to \mathbf{x}_{\text{LS}} = \mathbf{A}^{-1}\mathbf{b}$, which you'll recall is noise corrupted. The early stopping of such iterations is known as *iterative regularization*, where the iteration count k is the regularization parameter.

To illustrate, we consider the simplest iterative regularization method, which is known as the Landweber iteration [4, 8]:

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \tau \mathbf{A}^T(\mathbf{A}\mathbf{x}_{k-1} - \mathbf{b}), \tag{2.8}$$

where $0 < \tau < 1/\sigma_1$ to guarantee convergence.

If $\mathbf{x}_0 = \mathbf{0}$, it can be shown that (2.8) can be written in the form (2.3) with filter factors

$$\phi_i^{(k)} = 1 - (1 - \tau\sigma_i^2)^k \tag{2.9}$$

for $i = 1, \ldots, n$. Thus $\mathbf{x}_k$ can be written in the form of (2.3):

$$\mathbf{x}_k = \mathbf{V}\mathbf{\Phi}_k\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b}, \tag{2.10}$$

where $\mathbf{\Phi}_k = \text{diag}(\phi_1^{(k)}, \ldots, \phi_n^{(k)})$. Given our assumption regarding the value of τ, we see that $\phi_i \to 1$ as $i \to \infty$. Hence the filtering becomes less pronounced as the iteration proceeds and eventually converges to $\mathbf{x}_{\text{LS}}$. Moreover, the closer τ is to $1/\sigma_1$ the faster will be the convergence.

In Figure 2.3, we plot the Landweber solution for the deblurring example with $\tau = 0.9$ after 300 iterations. The choice of the stopping iteration k corresponds to the regularization parameter in this case. Objective methods for choosing k are discussed in Section 2.2.

Other iterative methods applied to the problem of minimizing $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ can be used in the same fashion, i.e., early stopping of their iterations can be used as a regularization method. Some even have the filtered SVD form (2.3), such as the steepest descent method (same as (2.8) but with $\tau = \tau_k$ iteration dependent), and

conjugate gradient iteration. However, even for such relatively simple algorithms, the corresponding SVD analysis quickly becomes complicated [22].

2.1.4 Statistical Properties of Filtered Solutions

Just as in the least squares case, we can make a distributional statement about estimators obtained from filtered solutions. A general filtered solution has the form $\mathbf{x}_\nu = \mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b}$, where $\nu = \alpha$ in the case of Tikhonov regularization and $\nu = k$ in the case of TSVD and truncated Landweber iterations. Recalling that $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and $\mathbf{b} \sim \mathcal{N}(\mathbf{A}\mathbf{x}, \sigma^2\mathbf{I})$, we therefore have

$$\mathbf{x}_\nu \sim \mathcal{N}\left(\mathbf{V}\mathbf{\Phi}_\nu\mathbf{V}^T\mathbf{x}, \sigma^2\mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^{-2}\mathbf{\Phi}_\nu\mathbf{V}^T\right), \tag{2.11}$$

where $\mathbf{\Sigma}^{-2} \stackrel{\text{def}}{=} (\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}$.

Projecting $\mathbf{x}_\nu$ onto the right singular vectors, as was done for the least squares solution in (1.25), we obtain

$$\begin{aligned}\mathbf{v}_i^T\mathbf{x}_\nu &\sim \mathcal{N}\left(\phi_i^{(\nu)}\mathbf{v}_i^T\mathbf{x}, (\phi_i^{(\nu)})^2\sigma^2/\sigma_i^2\right). \\ &\stackrel{d}{=} \phi_i^{(\nu)}\mathcal{N}\left(\mathbf{v}_i^T\mathbf{x}, \sigma^2/\sigma_i^2\right), \\ &\stackrel{d}{=} \phi_i^{(\nu)}\mathbf{v}_i^T\mathbf{x}_{\text{LS}}\end{aligned} \tag{2.12}$$

where '$\stackrel{d}{=}$' denotes 'equal in distribution', and the last equation follows from (1.25). Thus for $0 < \phi_i^{(\nu)} < 1$, in the direction $\mathbf{v}_i$, the filtered solution $\mathbf{x}_\nu$ has a smaller variance than the least squares solution $\mathbf{x}_{\text{LS}}$, as desired. However the mean of $\mathbf{v}_i^T\mathbf{x}_\nu$ is no longer $\mathbf{v}_i^T\mathbf{x}$, as is the case for $\mathbf{v}_i^T\mathbf{x}_{\text{LS}}$, which means that $\mathbf{x}_\nu$ is now biased.

We define the *bias* for the estimator $\mathbf{x}_\nu$ by

$$\text{bias}[\mathbf{x}_\nu] = \mathbf{x} - E[\mathbf{x}_\nu] = \mathbf{V}(\mathbf{I} - \mathbf{\Phi}_\nu)\mathbf{V}^T\mathbf{x} = \sum_{i=1}^{n}(1 - \phi_i^{(\nu)})(\mathbf{v}_i^T\mathbf{x})\mathbf{v}_i, \tag{2.13}$$

where E is the expected value function, and the total variance is given by

$$\text{tr}(\sigma^2\mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^{-2}\mathbf{\Phi}_\nu\mathbf{V}^T) = \sigma^2\sum_{i=1}^{n}(\phi_i^{(\nu)}/\sigma_i)^2. \tag{2.14}$$

For multivariate random vectors, the *mean square error* for $\mathbf{x}_\nu$ is defined [24]

$$\begin{aligned}\text{MSE}(\nu) &= E[\|\mathbf{x}_\nu - \mathbf{x}\|^2] \\ &= \text{tr}(\sigma^2\mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^{-2}\mathbf{\Phi}_\nu\mathbf{V}^T) + \|\text{bias}[\mathbf{x}_\nu]\|^2 \\ &= \sigma^2\sum_{i=1}^{n}(\phi_i^{(\nu)}/\sigma_i)^2 + \sum_{i=1}^{n}(1 - \phi_i^{(\nu)})^2(\mathbf{v}_i^T\mathbf{x})^2.\end{aligned} \tag{2.15}$$

Thus to minimize the MSE, the filter factor $\phi_i^{(\nu)}$ should be chosen to balance the competing desires for small variance and small bias in the estimator.

A similar treatment of the statistical properties of regularized solutions can be found in [24, 26]. Moreover, regularization methods exist that strive to balance the desire for small bias and variance, notably the method of Backus and Gilbert [2], which we do not present here, but which can also be found in [24, 26].

2.2 Regularization Parameter Selection Methods

The practical usefulness of the regularization approaches presented above hinges on there being methods for choosing the regularization parameter. For a given regularization technique with filter factors parameterized by ν, yielding regularized solution $\mathbf{x}_\nu$, a natural choice for ν is the value that minimizes $\text{MSE}(\mathbf{x}_\nu)$ defined in (2.15). Unfortunately, the computation of an estimator for $\text{MSE}(\mathbf{x}_\nu)$ is infeasible in general, since in practice we do not know $\mathbf{x}$.

2.2.1 The Unbiased Predictive Risk Estimator

A related approach is to instead minimize the *predictive risk*, which is defined

$$E[\|\mathbf{A}\mathbf{x}_\nu - \mathbf{A}\mathbf{x}\|^2], \tag{2.16}$$

where ν is the regularization parameter (α or k in our examples). Since the unknown $\mathbf{x}$ is in (2.16), we can't use it directly, but we can replace (2.16) by an unbiased estimator that does not contain $\mathbf{x}$. The resulting method is appropriately known as the *unbiased predicative risk estimator* (UPRE) method; it was first introduced in [19], but can also be found in [28].

To obtain the estimator for (2.16), we first define $\mathbf{A}_\nu$ to be the regularization matrix such that $\mathbf{x}_\nu = \mathbf{A}_\nu\mathbf{b}$, where $\mathbf{A}_\nu = \mathbf{U}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{V}^T$. Then, given that $\mathbf{A}\mathbf{A}_\nu$ is symmetric in the cases of interest to us,

$$\begin{aligned}
E[\|\mathbf{A}\mathbf{x}_\nu - \mathbf{A}\mathbf{x}\|^2] &= E[\|\mathbf{A}\mathbf{A}_\nu\mathbf{b} - \mathbf{A}\mathbf{x}\|^2] \\
&= E[\|(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{A}_\nu\boldsymbol{\epsilon}\|^2] \\
&= \|(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x}\|^2 + E[2\boldsymbol{\epsilon}^T\mathbf{A}\mathbf{A}_\nu(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}^T(\mathbf{A}\mathbf{A}_\nu)^2\boldsymbol{\epsilon}] \\
&= \|(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x}\|^2 + E[\boldsymbol{\epsilon}^T(\mathbf{A}\mathbf{A}_\nu)^2\boldsymbol{\epsilon}] \\
&= \|(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x}\|^2 + \text{tr}((\mathbf{A}\mathbf{A}_\nu)^2).
\end{aligned} \tag{2.17}$$

The last equality follows from the fact that for a symmetric matrix $\mathbf{B}$ and white noise vector $\mathbf{v}$, $E[\mathbf{v}^T\mathbf{B}\mathbf{v}] = \text{tr}(\mathbf{B})$ [28, Lemma 7.2].

A similar calculation to that for deriving (2.17) gives us that

$$E[\|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2] = \|(\mathbf{A}\mathbf{A}_\nu - \mathbf{I})\mathbf{A}\mathbf{x}\|^2 + \text{tr}((\mathbf{A}\mathbf{A}_\nu)^2) - 2\sigma^2\text{tr}(\mathbf{A}\mathbf{A}_\nu) + m\sigma^2,$$

and hence,

$$E[\|\mathbf{A}\mathbf{x}_\nu - \mathbf{A}\mathbf{x}\|^2] = E[\|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2] + 2\sigma^2\text{tr}(\mathbf{A}\mathbf{A}_\nu) - m\sigma^2. \tag{2.18}$$

Finally, we define

$$U(\nu) = \|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2 + 2\sigma^2\text{tr}(\mathbf{A}\mathbf{A}_\nu) - m\sigma^2, \tag{2.19}$$

which is an unbiased estimator of the predictive risk for given ν. Choosing the ν that minimizes $U(\nu)$ defines the UPRE regularization parameter choice method.

With the SVD of $\mathbf{A}$ in hand, we can simplify (2.19). In particular, note that since $\mathbf{A}_\nu = \mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T$ (see (2.3)), so that

$$\mathbf{A}\mathbf{A}_\nu = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T,$$

and hence

$$\begin{aligned} U(\nu) &= \|(\mathbf{U}\mathbf{\Sigma}(\mathbf{\Phi}_\nu - \mathbf{I})\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b}\|^2 + 2\sigma^2\mathrm{tr}(\mathbf{U}\mathbf{\Sigma}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T) - m\sigma^2 \\ &= \sum_{i=1}^{n}[((\phi_i^{(\nu)} - 1)\mathbf{u}_i^T\mathbf{b})^2 + 2\sigma^2\phi_i^{(\nu)}] - m\sigma^2, \end{aligned} \tag{2.20}$$

the verification of which we leave as an exercise. The '$-m\sigma^2$' term can be ignored since it does not depend upon ν.

For each of the filters above, we can write down the specific UPRE functions; for TSVD and Tikhonov regularization they are given, respectively, by

$$U(k) = \sum_{i=k+1}^{n}(\mathbf{u}_i^T\mathbf{b})^2 + 2\sigma^2 k \tag{2.21}$$

$$U(\alpha) = \sum_{i=1}^{n}\frac{\alpha^2(\mathbf{u}_i^T\mathbf{b})^2}{(\sigma_i^2 + \alpha)^2} + 2\sigma^2\sum_{i=1}^{n}\frac{\sigma_i^2}{\sigma_i^2 + \alpha}. \tag{2.22}$$

The expression for U for the Landweber iteration it is left to the exercises.

We apply UPRE to the problem of choosing k for TSVD in the deblurring example and α for Tikhonov regularization in the kernel reconstruction example. The reconstructions $\mathbf{x}_k$ and $\mathbf{x}_\alpha$ are plotted in Figure 2.4.

2.2.2 Generalized Cross Validation

One of the limitations of the UPRE selection method is that it requires knowledge of the noise variance σ^2.

A method similar to UPRE that does not require σ^2 is generalized cross validation (GCV). GCV was originally introduced in [30], is very commonly used in inverse problems (see, e.g., [1, 14, 15, 28]), and is an approximation of leave-one-out cross validation (LOOCV).

The idea behind LOOCV is as follows. First, we define $\mathbf{A}^{[i]}\mathbf{x} = \mathbf{b}^{[i]}$ to be the linear system that results after the i^{th} element of $\mathbf{b}$ and the i^{th} row of $\mathbf{A}$ are removed, and $\mathbf{x}_\nu^{[i]} = \mathbf{A}_\nu^{[i]}\mathbf{b}^{[i]}$, where $\mathbf{A}_\nu^{[i]}$ is the regularization matrix for the modified system. Finally, we define the LOOCV choice of regularization parameter ν to be the minimizer of

$$V(\nu) = \frac{1}{m}\sum_{i=1}^{m}([\mathbf{A}\mathbf{x}_\nu^{[i]}]_i - b_i)^2,$$

i.e., the value of ν that yields the best overall prediction of the removed data points.

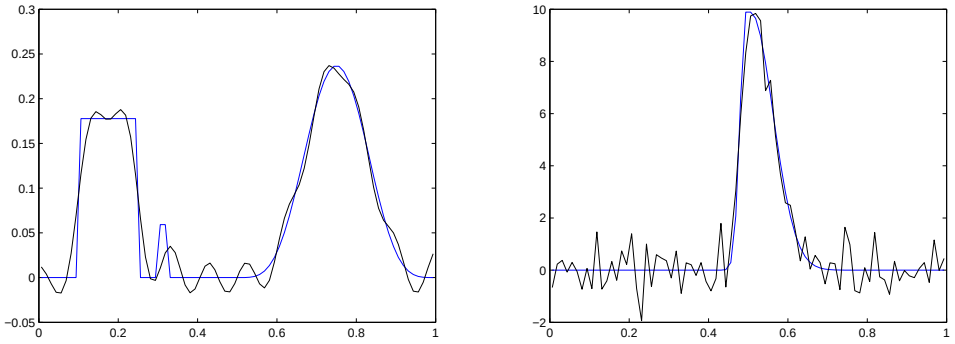

Figure 2.4. *On the left is a plot of the TSVD reconstruction* $\mathbf{x}_k$ *with the UPRE choice of* $k = 22$ *in the deblurring example. On the right is a plot of the Tikhonov reconstruction* $\mathbf{x}_\alpha$ *with the UPRE choice of* $\alpha = 0.000074$ *for the kernel reconstruction example.*

Evaluating the function V at a single ν involves solving m large-scale inverse problems, and hence, LOOCV is infeasible to implement for the problems of interest to us. However, a feasible approximation of V exists. We derive it by first noting that V can be equivalently written as (see [1] for details)

$$V(\nu) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{[\mathbf{A}\mathbf{x}_\nu - \mathbf{b}]_i}{1 - [\mathbf{A}\mathbf{A}_\nu]_{ii}}\right)^2.$$

For many large-scale problems, the diagonal values of $\mathbf{A}\mathbf{A}_\nu$ are infeasible to compute. Thus the GCV approximation of $V(\nu)$ replaces $1-[\mathbf{A}\mathbf{A}_\nu]_{ii}$ by $(m-\mathrm{tr}(\mathbf{A}\mathbf{A}_\nu))/m$ for all i to obtain

$$G(\nu) = \frac{m\|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2}{[\mathrm{tr}(\mathbf{I} - \mathbf{A}\mathbf{A}_\nu)]^2}. \tag{2.23}$$

The GCV choice of $\nu > 0$ is then the one that minimizes $G(\nu)$.

Simplifying (2.23) using the SVD as we did for UPRE above, we obtain

$$G(\nu) = \frac{\sum_{i=1}^{n}((\phi_i^{(\nu)} - 1)\mathbf{u}_i^T\mathbf{b})^2}{(m - \sum_{i=1}^{n}\phi_i^{(\nu)})^2}, \tag{2.24}$$

which for TSVD and Tikhonov regularization has the form, respectively, of

$$G(k) = \left(\sum_{i=k+1}^{n}(\mathbf{u}_i^T\mathbf{b})^2\right) \Bigg/ (m-k)^2 \tag{2.25}$$

$$G(\alpha) = \left(\sum_{i=1}^{n}\frac{\alpha^2(\mathbf{u}_i^T\mathbf{b})^2}{(\sigma_i^2+\alpha)^2}\right) \Bigg/ \left(m - \sum_{i=1}^{n}\frac{\sigma_i^2}{\sigma_i^2+\alpha}\right)^2, \tag{2.26}$$

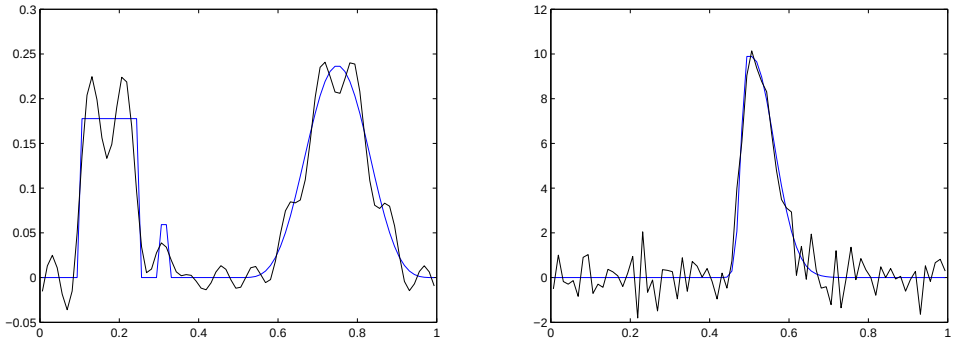

Figure 2.5. *On the left is a plot of the TSVD reconstruction* $\mathbf{x}_k$ *with the GCV choice of* $k = 25$*. On the right is a plot of the Tikhonov reconstruction* $\mathbf{x}_\alpha$ *with the GCV choice of* $\alpha = 0.000066$*.*

The expression for the Landweber iteration it is left to the reader.

We apply GCV to the problem of choosing k for TSVD in the deblurring example and α for Tikhonov regularization in the kernel reconstruction example. The reconstructions $\mathbf{x}_k$ and $\mathbf{x}_\alpha$ are plotted in Figure 2.4.

2.2.3 The Discrepancy Principle

Next, we present the *discrepancy principle* (DP), which is a standard regularization parameter selection method for inverse problems; see e.g., [1, 8, 14, 15, 17, 28]. DP advocates choosing ν so that the sum of squared residuals $\|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2$ is equal to the noise level $E[\|\boldsymbol{\epsilon}\|^2]$, and it is equivalent to the method of moments in statistics on the second moment of the distribution. Since given our assumptions, $E[\|\boldsymbol{\epsilon}\|^2] = m\sigma^2$, this amounts to solving to nonlinear equation

$$0 = D(\nu) \stackrel{\text{def}}{=} \|\mathbf{A}\mathbf{x}_\nu - \mathbf{b}\|^2 - m\sigma^2. \tag{2.27}$$

The function D simplifies using the SVD to

$$D(\nu) = \sum_{i=1}^{n} [(\phi_i^{(\nu)} - 1)\mathbf{u}_i^T\mathbf{b}]^2 - m\sigma^2. \tag{2.28}$$

which for TSVD and Tikhonov regularization has the form, respectively, of

$$D(k) = \sum_{i=k+1}^{n} (\mathbf{u}_i^T\mathbf{b})^2 - m\sigma^2 \tag{2.29}$$

$$D(\alpha) = \sum_{i=1}^{n} \frac{\alpha^2(\mathbf{u}_i^T\mathbf{b})^2}{(\sigma_i^2 + \alpha)^2} - m\sigma^2. \tag{2.30}$$

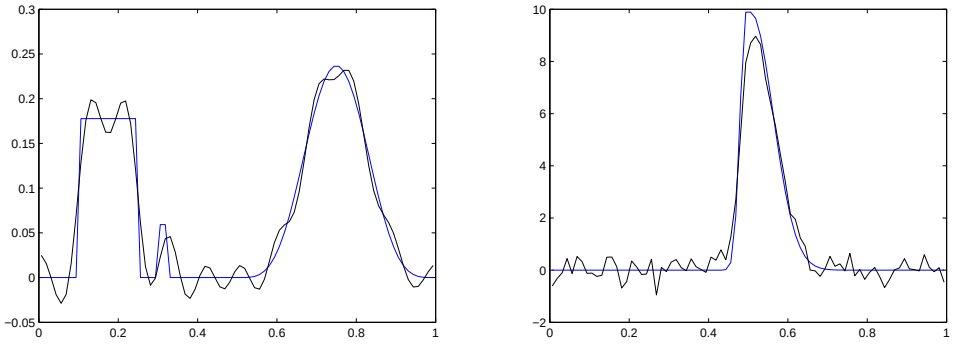

Figure 2.6. *On the left is a plot of the TSVD reconstruction* $\mathbf{x}_k$ *with the GCV choice of* $k = 23$*. On the right is a plot of the Tikhonov reconstruction* $\mathbf{x}_\alpha$ *with the GCV choice of* $\alpha = 0.00024$*.*

We apply DP to the problem of choosing k for TSVD in the deblurring example and α for Tikhonov regularization in the kernel reconstruction example. The reconstructions $\mathbf{x}_k$ and $\mathbf{x}_\alpha$ are plotted in Figure 2.4.

Note that when using DP for TSVD or Landweber iterations, you can choose the first k such that $D(k) \leq 0$. For Tikhonov regularization, on the other hand, a root finding algorithm such as the secant method or Newton's method can be used; note that $D(\alpha)$ is a differentiable function of α. Alternatively, a minimization algorithm can be used to compute the minimizer of $D(\nu)^2$, which will work even in cases where $D(\nu) > 0$ for all $\nu > 0$

The expression for D for the Landweber iteration it is left to the exercises.

2.2.4 The L-curve Criterion

The *L-curve method* is a heuristic developed in [12] that is widely used and that has been found to work well on many examples in inverse problems. We consider it here only as a method for choosing the Tikhonov regularization parameter α. It can also be used for choosing the TSVD parameter k, but it is a more involved procedure that we leave to [14] and the references therein.

The effectiveness of the L-curve method hinges on the fact that for a variety of inverse problems the parameterized curve

$$\left(\xi(\alpha) \stackrel{\text{def}}{=} \log \|\mathbf{x}_\alpha\|^2, \rho(\alpha) \stackrel{\text{def}}{=} \log \|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|^2\right), \tag{2.31}$$

has a distinct L-shape. The corner of the curve corresponds, roughly, to the transition point between the values of α near 0 for which $\|\mathbf{x}_\alpha\|$ is large and $\|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|$ is small (too little regularization), and the large values of α for which $\|\mathbf{x}_\alpha\|$ is small and $\|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|$ is large (too much regularization). The L-curve method chooses α corresponding to the corner, or point of maximum curvature, along the curve (2.31),

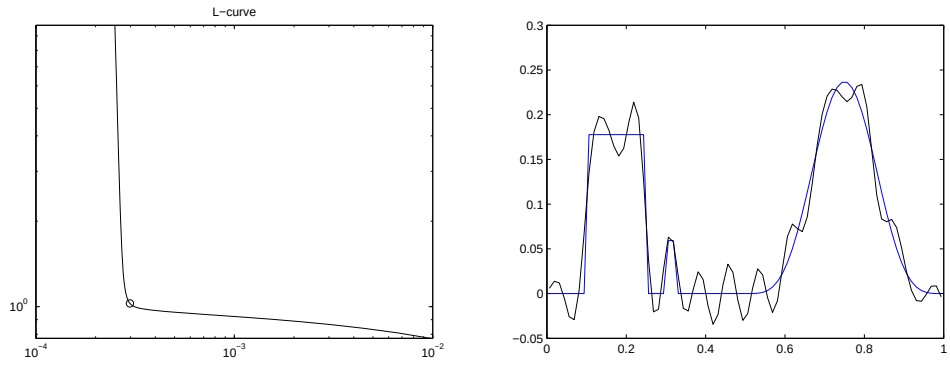

Figure 2.7. *On the left is a plot of the curve* $\left(\log\|\mathbf{x}_\alpha\|^2, \log\|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|^2\right)$ *for the 1D image deblurring test problem. The L-curve choice of the regularization parameter* $\alpha = 0.00035$ *corresponds to the circle at the corner of the curve. On the right is a plot of the corresponding Tikhonov reconstruction* $\mathbf{x}_\alpha$*.*

since this will often correspond to the value of α at which this transition occurs. We plot the L-curve for the 1D deblurring example on the left in Figure 2.7.

The equation for the curvature for a parametric curve of the form $(\xi(\alpha), \rho(\alpha))$ can be found in standard texts on analytic geometry and is given by

$$C(\alpha) = \frac{\rho(\alpha)'\xi(\alpha)'' - \rho(\alpha)''\xi(\alpha)'}{(\rho'(\alpha)^2 + \xi'(\alpha)^2)^{3/2}}.$$

For the parametric curve defined by (2.31), $C(\alpha)$ can be expressed entirely in terms of $s(\alpha) \stackrel{\text{def}}{=} \|\mathbf{x}_\alpha\|^2$, $r(\alpha) \stackrel{\text{def}}{=} \|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|^2$ and $s'(\alpha)$ [28]:

$$C(\alpha) = -\frac{r(\alpha)s(\alpha)[\alpha r(\alpha) + \alpha^2 s(\alpha)] + [r(\alpha)s(\alpha)]^2/s'(\alpha)}{[r(\alpha)^2 + \alpha^2 s(\alpha)^2]^{3/2}}. \tag{2.32}$$

Using the SVD, we can derive the following formula for $s'(\alpha)$:

$$s'(\alpha) = \sum_{i=1}^{n} \frac{-2\sigma_i^2(\mathbf{u}_i^T\mathbf{b})^2}{(\sigma_i^2 + \alpha)^3}. \tag{2.33}$$

The L-curve choice for the regularization parameter α is the value that maximizes the curvature C defined by (2.32). For the 1D deblurring test case, the L-curve choice was $\alpha = 0.00035$, which corresponds to the circle in left-hand plot in Figure 2.7. The resulting Tikhonov solution $\mathbf{x}_\alpha$ is plotted on the right in Figure 2.7.

Remarks: A regularization method is said to be convergent if, roughly speaking, as the noise level (in our case represented by σ^2) goes to 0, the methods choose values of ν such that $\mathbf{x}_\nu$ converges to $\mathbf{x}$. An in-depth analysis of the convergence properties

of the above methods can be found in [28]. To summarize that analysis, UPRE, GCV, and DP are convergent methods, with UPRE having the best convergence properties, followed by GCV, and then DP. We remind the reader that both UPRE and DP require an estimate of σ^2, while GCV does not. The L-curve method is shown to be non-convergent in [28], an important result that one should weigh when choosing a regularization parameter selection method. Nonetheless, the L-curve method is frequently used and has been shown to work reasonably well on a large number of examples in inverse problems, including the one in this chapter.

It is also worth mentioning that since the data $\mathbf{b}$ is a random vector, the value of ν given by the above methods will also be random. For this reason, a Monte Carlo analysis of the above four regularization parameter selection methods is useful. Such an analysis can be found for GCV, DP, and the L-curve in [14].

Exercises

2.1. a. Define $\ell(\mathbf{x})$ to be the penalized least squares function in (2.5). Show that $\nabla\ell(\mathbf{x}) = \mathbf{0}$ yields (2.6) and hence that the solutions of (2.5) and (2.6) coincide. Argue, moreover, that the solution is unique.

b. Use the SVD of the matrix $\mathbf{A}$ to show that the solutions of (2.6) (and hence of (2.5)) can be written in filtered SVD form (2.3) with Tikhonov filter (2.7).

2.2. a. Create plots of the *relative error* function $f(\alpha) = \|\mathbf{x}_\alpha - \mathbf{x}_{\text{true}}\|/\|\mathbf{x}_{\text{true}}\|$ within both `DeblurTikhonov.m` and `PSFreconTikhonov.m` (use MATLAB's `logspace` function). Then compute the α on your computational grid that minimizes the relative error and plot the corresponding regularization solution $\mathbf{x}_\alpha$.

b. Repeat part (a) with `DeblurTSVD.m` and `PSFreconTSVD.m`.

c. Repeat part (a) with `DeblurLandweber.m` and `PSFreconLandweber.m`.

2.3. Show that (2.8) can be written as (2.3) with ϕ_i defined by (2.9). *Hint:* First show that for $\mathbf{x}_0 = \mathbf{0}$, (2.8) can be written $\mathbf{x}_k = \sum_{j=0}^{k-1}(\mathbf{I} - \tau\mathbf{A}^T\mathbf{A})^j(\tau\mathbf{A}^T\mathbf{b})$. Then substitute the SVD of $\mathbf{A}$ into the expression.

2.4. a. Verify that (2.11) holds assuming $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, $\mathbf{b} \sim \mathcal{N}(\mathbf{A}\mathbf{x}, \sigma^2\mathbf{I})$, and $\mathbf{x}_\nu = \mathbf{V}\mathbf{\Phi}_\nu\mathbf{\Sigma}^\dagger\mathbf{U}^T\mathbf{b}$. Use the fact that for a general $n \times 1$ normal random vector $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ if $\mathbf{w} = \mathbf{B}\mathbf{v}$, where $\mathbf{B}$ is an $m \times n$ matrix, $\mathbf{w} \sim \mathcal{N}(\mathbf{B}\boldsymbol{\mu}, \mathbf{B}\mathbf{C}\mathbf{B}^T)$.

b. Use (2.11) to derive the distribution for $\mathbf{v}_i^T\mathbf{x}_\nu$ akin to (1.25).

2.5. a. Find the expressions for the MSE defined by (2.15) for the TSVD, Tikhonov, and Landweber regularized solutions.

b. Create plots of MSE(α), using MATLAB's `logspace` function, within both `DeblurTikhonov.m` and `PSFreconTikhonov.m`. Then compute the α on your computational grid that minimizes the MSE and plot the corresponding regularization solution $\mathbf{x}_\alpha$.

c. Repeat part (a) with `DeblurTSVD.m` and `PSFreconTSVD.m`.

d. Repeat part (a) with `DeblurLandweber.m` and `PSFreconLandweber.m`.

e. If you have done Exercise 2, how do the MSE and relative error curves compare? Plot the curves for `DeblurTikhonov.m` and `PSFreconTikhonov.m` separately.

2.6. a. Modify `DeblurTikhonov.m` and `PSFreconTikhonov.m` so that (some subset of) the following curves are plotted together, (all on the same numerical grid created using `logspace`):

i. the UPRE curve $U(\alpha)$ defined by (2.22),

ii. the GCV curve $G(\alpha)$ defined by (2.26),

iii. the DP curve $D(\alpha)^2$ defined by (2.30), and

iv. the L-curve curvature function $-C(\alpha)$ defined by (2.32).

Compute the α that minimizes each curve on your computational grid and plot the corresponding regularization solution $\mathbf{x}_\alpha$.

b. Repeat part a, (i)-(iii), with `DeblurTSVD.m` and `PSFreconTSVD.m`.

c. Repeat part a, (i)-(iii), with `DeblurLandweber.m` and `PSFreconLandweber.m`.

d. If you have done Exercises 2 and/or 5, how do the curves you've plotted compare to the *relative error* and/or the MSE? Plot the curves for the deblurring and kernel reconstruction test cases separately.

2.7. a. Derive formulas for $U(k)$, $G(k)$, $D(k)$ defined by (2.20), (2.24), and (2.28), respectively, for the Landweber iteration.

b. Using `DeblurLandweber.m`, compute Landweber iterations and report the iterations at which $U(k)$ and $G(k)$ first increase, and $D(k)$ first drops below zero. Plot the corresponding reconstructions.

2.8. a. Use the SVD of $\mathbf{A}$ to verify the step (2.20).

b. Use the SVD of $\mathbf{A}$ to verify that (2.23) and (2.24) are equivalent.

c. Use the SVD of $\mathbf{A}$ to verify that (2.27) and (2.28) are equivalent.

2.9. In exercise 1.2, you were asked to modify `Deblur1d.m` so that the convolution kernel

$$a(s) = \begin{cases} 100s + 10, & -\frac{1}{10} \le s \le 0, \\ -100s + 10, & 0 \le s \le \frac{1}{10}, \\ 0, & \text{otherwise.} \end{cases}$$

is used instead to define $\mathbf{A}$.

a. Use Tikhonov regularization together with GCV and L-curve to reconstruct $\mathbf{x}$ from observations $\mathbf{b}$. What is the optimal regularization parameter α in each case? Which gives the better reconstruction in your opinion?

b. Use TSVD regularization together with UPRE and DP to reconstruct $\mathbf{x}$ from observations $\mathbf{b}$. What is the optimal regularization parameter k in each case? Which gives the better reconstruction in your opinion?

c. Use truncated Landweber together with DP to reconstruct $\mathbf{x}$ from observations $\mathbf{b}$. What is the optimal stopping iteration k?

Chapter 3

Boundary Conditions and Two-Dimensional Test Cases

In this chapter, we begin by first discussing the assumption of boundary conditions in one-dimension, which must be made when discretizing (1.4). We then present two-dimensional imaging test cases, which also require boundary conditions assumptions.

3.1 Boundary Conditions in One-Dimensional Deconvolution

Consider the one-dimensional convolution model

$$b(s) = \int_{-\infty}^{\infty} a(s - s')x(s')ds'.$$

In practice, the domain for b is defined by the measurement device. We'll assume that it is $s \in [0, 1]$. Moreover, the kernel a is typically centered and given on a finite interval of the form $[-\xi, \xi]$, where ξ is some positive number. Given these assumptions, and assuming that $a = 0$ outside of $[-\xi, \xi]$, the convolution model reduces to

$$b(s) = \int_{-\xi}^{1+\xi} a(s - s')x(s')ds', \quad s \in [0, 1]. \tag{3.1}$$

As in Chapter 1, we can apply mid-point quadrature to the integral (3.1), i.e.,

$$\int_{-\xi}^{1+\xi} f(s')\, ds' = h \sum_{j=-k+1}^{n+k} f(s'_j) + E_n, \tag{3.2}$$

where $h = 1/n$, $s'_j = (j - \frac{1}{2})/n$, for $j = -k+1, \ldots, n+k$, and E_n is the quadrature error. Here k is defined to be the largest positive integer such that $(k - \frac{1}{2})/n < \xi$. Assuming the same grid for the s variable within [0,1], i.e. $s_j = s'_j$ for $j = 1, \ldots, n$, if we define $b_i \stackrel{\text{def}}{=} b(s_i)$, $x_j \stackrel{\text{def}}{=} x(s'_j)$, $a_{i-j} = a(s_i - s'_j) = a((i-j)h)$, and apply (3.2)

to (3.1), then, ignoring the quadrature error E_n, we obtain

$$b_i = h \sum_{j=i-k}^{i+k} a_{i-j} x_j = \sum_{j=-k}^{k} a_{-j}\, x_{i+j}, \quad i = 1, \ldots, n, \tag{3.3}$$

Equation (3.3) follows from (3.2), since $a(s) = 0$ for $s \in [-\xi, \xi]$ implies $a_{i-j} = 0$ for $|i - j| > k$. In matrix-vector notation, (3.3) has the form

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_k & \cdots & a_0 & \cdots & a_{-k} & & & & \\ & a_k & \cdots & a_0 & \cdots & a_{-k} & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & a_k & \cdots & a_0 & \cdots & a_{-k} & \\ & & & & a_k & \cdots & a_0 & \cdots & a_{-k} \end{bmatrix} \begin{bmatrix} x_{-k+1} \\ \vdots \\ x_1 \\ \vdots \\ x_n \\ \vdots \\ x_{n+k} \end{bmatrix}. \tag{3.4}$$

It is evident from (3.4) that the values of b_i for i near 1 and n will depend upon the elements $(x_{-k+1}, \ldots, x_0)$ and $(x_{n+1}, \ldots, x_{n+k})$, respectively, all of which lie outside of [0,1]. Note that for $k > 0$, this system has more unknowns than equations, i.e. it is under-determined.

In practice, in order to reduce the number of unknowns to be equal to the number of equations, assumptions are made about the values of $(x_{-k+1}, \ldots, x_0)$ and $(x_{n+1}, \ldots, x_{n+k})$. These assumptions are called *boundary conditions* in the imaging literature [15]. Three standard boundary conditions are zero, periodic, and Neumann.

Zero boundary conditions correspond to assuming $(x_{-k+1}, \ldots, x_0) = \mathbf{0}$ and $(x_{n+1}, \ldots, x_{n+k}) = \mathbf{0}$. In this case, assuming $k = n - 1$, (3.4) takes the form of the $n \times n$ linear system (1.6) from Chapter 1, which has a coefficient matrix with Toeplitz (constant diagonals) structure [28]. We leave the derivation of this system to the reader in Exercise 3.

Periodic boundary conditions correspond to assuming $(x_{-k+1}, \ldots, x_0) = (x_{n-k+1}, \ldots, x_n)$ and $(x_{n+1}, \ldots, x_{n+k}) = (x_1, \ldots, x_k)$. Assuming $k = (n-1)/2$, the coefficient matrix in (3.4) can be modified to reflect these assumptions, yielding an $n \times n$ linear system with a circulant coefficient matrix [28]. We leave the derivation of this system to the reader in Exercise 2.

Finally, Neumann boundary conditions correspond to a reflection of the signal about the boundaries, i.e.,$(x_{-k+1}, \ldots, x_0) = (x_k, \ldots, x_1)$ and $(x_{m+1}, \ldots, x_{m+k}) = (x_m, \ldots, x_{m-k+1})$. Similar to the periodic case, the coefficient matrix in (3.4) can be modified, but only if the kernel is symmetric, i.e., $a_i = a_{-i}$ for all i, yielding an $n \times n$ linear system with Toeplitz-plus-Hankel structure [15]. We leave the derivation of this system to the reader in Exercise 4.

As was stated, the boundary condition assumptions reduce the number of unknowns to be equal to the number of equations. At least as important, however, is the fact that the resulting coefficient matrices have structure: Toeplitz, circulant,

and toeptlitz-plus-Hankel [15]. This structure can be exploited: circulant matrices can be diagonalized by the discrete Fourier transform (DFT); toeplitz matrices can be embedded within circulant matrices, which can in turn be diagonalized by the DFT; and finally, toeptliz-plus-Hankel matrices can be diagonalized by the discrete cosine transform (DCT). In one dimension, exploiting this special structure is not so important, however when we move to two-dimensions, it will be necessary to exploit structure in the coefficient matrix in order to keep computations efficient.

The drawback to using the above boundary conditions is that they lack physical motivation, yielding artifacts in reconstructions when the true signal does not satisfy the underlying assumptions. An alternative to assuming boundary conditions is to discretize (3.1) with both s and s' defined on the extended domain $[-\xi, 1+\xi]$. This yields the following system of equations:

$$\begin{bmatrix} b_{-k+1} \\ \vdots \\ b_1 \\ \vdots \\ b_n \\ \vdots \\ b_{n+k} \end{bmatrix} = \begin{bmatrix} a_k & \cdots & a_0 & \cdots & a_{-k} & & & & \\ & a_k & \cdots & a_0 & \cdots & a_{-k} & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & a_k & \cdots & a_0 & \cdots & a_{-k} & \\ & & & & a_k & \cdots & a_0 & \cdots & a_{-k} \end{bmatrix} \begin{bmatrix} x_{-2k+1} \\ \vdots \\ x_1 \\ \vdots \\ x_n \\ \vdots \\ x_{n+2k} \end{bmatrix},$$

where $\widehat{\mathbf{b}}$, on the left, has size $(n+2k)\times 1$, the matrix on the right has size $(n+2k)\times(n+4k)$, and the x-vector on the right has $(n+4k)$ elements.

Now, if we make one of the boundary conditions assumptions above, we obtain a system of the form $\widehat{\mathbf{b}} = \widehat{\mathbf{A}}\mathbf{x}$, where $\widehat{\mathbf{A}}$ is $(n+2k)\times(n+2k)$ and $\mathbf{x}$ is $(n+2k)\times 1$. Then, finally, if $\mathbf{D}$ is the indicator matrix on $[0,1]$, given by rows $k+1$ to $k+n$ of the $(n+2k)\times(n+2k)$ identity matrix, we have $\mathbf{b} = \mathbf{D}\widehat{\mathbf{b}}$, and our linear model takes the form

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}, \quad \text{where} \quad \mathbf{A} = \mathbf{D}\widehat{\mathbf{A}},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}). \tag{3.5}$$

Note that $\mathbf{b}, \boldsymbol{\epsilon} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^{n+2k}$, and $\mathbf{A} \in \mathbb{R}^{n\times(n+2k)}$, thus estimating $\mathbf{x}$ from $\mathbf{b}$ is under-determined for $k > 0$. We will call this the data driven boundary condition approach.

In Figure 3.1, we plot the true image as well as data generated using $\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ with σ^2 chosen so that the signal-to-noise ratio, $\|\mathbf{A}\mathbf{x}\| \Big/ \sqrt{n\sigma^2}$, is 50. Note that here, [0,1] is the extended domain, while $[0.15, 0.85]$ is the image on which x is reconstructed and b is observed. We reconstruct $\mathbf{x}$ using Tikhonov regularization and both the date driven and zero boundary conditions, with α chosen in both cases using the discrepancy principle. These two solutions are also plotted in Figure 3.1, together with the true image. Due to the fact that the zero boundary condition is a bad approximation of reality in this example, the corresponding reconstruction is poor near the boundaries, $t = 0.15$ and 0.85.

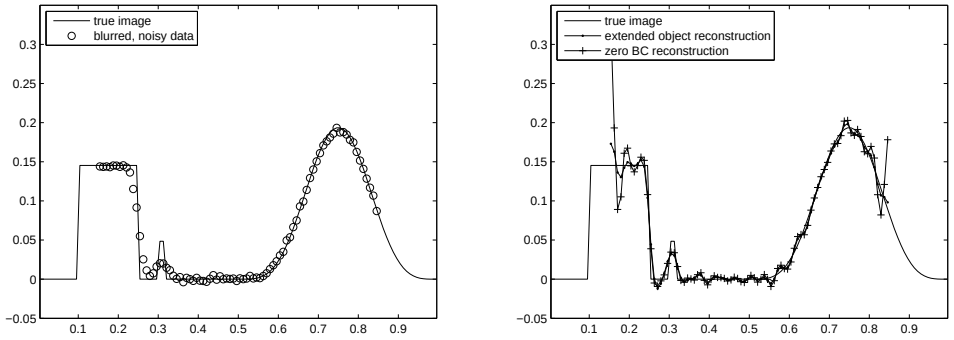

Figure 3.1. *One dimensional image deblurring example. On the left is a plot of the true image and the blurred, noisy data. On the right is a reconstruction of the Tikhonov reconstructions using both the data driven and zero boundary conditions, together with the true image.*

3.2 Two-dimensional image deblurring

We now discuss two-dimensional imaging. In two-dimensions, the convolution model has the form

$$b(s,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} a(s-s',t-t')x(s',t')\,ds'\,dt'. \tag{3.6}$$

We assume that $0 \le s,t \le 1$ and that a is defined on $[-\xi,\xi]\times[-\xi,\xi]$, so that the model becomes instead

$$b(s,t) = \int_{-\xi}^{1+\xi}\int_{-\xi}^{1+\xi} a(s-s',t-t')x(s',t')\,ds'\,dt', \quad 0 \le s,t \le 1. \tag{3.7}$$

We now discuss different boundary conditions assumptions and kernel structures. We will begin with the zero boundary case with a separable kernel, mainly because this case connects nicely to one-dimensional deblurring. We then present the periodic boundary condition case, because it is both very flexible and very computationally efficient. And finally, we present the extension of the data driven boundary condition to two-dimensions.

3.2.1 Zero Boundary Conditions and Separable Blur

We first consider the zero boundary condition case, in which $x = 0$ outside of $[0,1]\times[0,1]$. Moreover, we assume that a is defined on $[-1,1]\times[-1,1]$. Then, we use midpoint quadrature to discretize (3.7) with $\xi = 1$, assuming the computational grid given by the ordered pairs $\{(s'_i,t'_j)\}_{i,j=-n+1}^{n} = \{(i-1/2)/n,(j-1/2)/n\}_{i,j=-n+1}^{n}$ and collocation of indices, i.e., $(s_l,t_m) = (s'_k,t'_l)$ for all $l,m = 1,\ldots,n$. Moreover, we define $b_{lm} = b(s_l,t_m)$, $x_{ij} = x(s'_i,t'_j)$, and $a_{l-i,\,m-j} = a(s_l - t_i, s'_m - t'_j) =$

$a((l-i)h,(m-j)h)$. Then, ignoring quadrature error, mid-point quadrature yields the following system of linear equations:

$$b_{lm} = \frac{1}{n^2}\sum_{i,j=1}^{n} a_{l-i,\,m-j}x_{ij}, \quad l,m = 1,\ldots,n. \tag{3.8}$$

In what follows, we define $\mathbf{B}$ and $\mathbf{X}$ to be the $n \times n$ arrays with components $[\mathbf{B}]_{lm} = b_{lm}$ and $[\mathbf{X}]_{ij} = x_{ij}$, respectively. Moreover, we define $\mathbf{b} = \text{vec}(\mathbf{B})$ and $\mathbf{x} = \text{vec}(\mathbf{X})$ to be the $n^2 \times 1$ vectors obtained by stacking the columns of $\mathbf{B}$ and $\mathbf{X}$, respectively, with the left column on top and right column on bottom. Doing this, we can express (3.8) in the form $\mathbf{b} = \mathbf{A}\mathbf{x}$.

As in the one dimensional case, the matrix $\mathbf{A}$ defined by (3.8) has special structure that can be exploited for efficient computations. Specifically, for a general kernel a, the zero boundary condition results in a matrix $\mathbf{A}$ that is block Toeplitz with Toeplitz blocks (BTTB) [28]. This structure can be exploited for fast computations. Specifically, $\mathbf{A}$ can be embedded within a block circulant with circulant blocks (BCCB) matrix [28], which can in turn be diagonalized by the two dimensional DFT (2D-DFT). Fast solutions of the resulting linear system of equations can then be obtained iteratively. This problem is discussed in detail in [28], so we do not pursue it here. However, if we assume additional structure on kernel a, i.e., that it is *separable*, we can extend our one-dimensional results to the two-dimensional case using the Kronecker product.

Separable kernel and the Kronecker product

The kernel a in (3.7) is *separable* if it can be written $a(s-s',t-t') = a_1(s-s')a_2(t-t')$ [15, 23]. We consider this as our first two dimensional example because the resulting discretization (3.8) can be expressed in terms of one-dimensional blurring matrices, by now familiar to the reader. An example of a separable kernel is a Gaussian with diagonal covariance, i.e.,

$$a(s,t) = \frac{1}{2\pi\sqrt{\gamma_1^2+\gamma_2^2}}\exp\left(-\frac{1}{2}(s^2/\gamma_1^2 + t^2/\gamma_2^2)\right), \quad \gamma_1,\gamma_2 > 0.$$

Then $a_1(x) = (2\pi\gamma_1^2)^{-1/2}\exp(-x^2/(2\gamma_1^2))$ and $a_2(t) = (2\pi\gamma_2^2)^{-1/2}\exp(-t^2/(2\gamma_2^2))$, which are both one-dimensional Gaussian kernels of the form (1.8) used in Chapter 1.

In the case where a is a separable kernel, equation (3.8) becomes

$$b_{lm} = \frac{1}{n^2}\sum_{j=1}^{n} a_2((m-j)h)\left(\sum_{i=1}^{n} a_1((l-i)h)x_{ij}\right), \quad l,m = 1,\ldots,n. \tag{3.9}$$

The corresponding 1D blurring matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ (defined by (1.10)) can then be used to express (3.9) in the following form:

$$\mathbf{B} = \mathbf{A}_1\mathbf{X}\mathbf{A}_2^T. \tag{3.10}$$

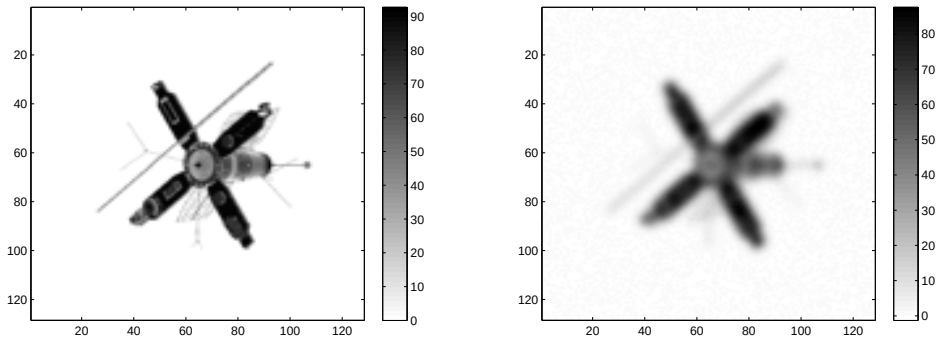

Figure 3.2. *Two dimensional deblurring example. The true image* $\mathbf{x}$ *is given on the left and the blurred and noisy data* $\mathbf{b}$ *is given on the right.*

Note that in (3.10), $\mathbf{A}_1$ blurs the columns of $\mathbf{X}$, while $\mathbf{A}_2$ blurs the rows; see [15] for more detail.

As we will soon see, equation (3.10) is handy for computations, however in order to unify our presentation with that of the previous chapters, and also to simplify our later discussions, we need to express (3.10) in the matrix-vector form $\mathbf{b} = \mathbf{A}\mathbf{x}$. This can be done using the Kronecker matrix product. In particular, if we define $\mathbf{x} = \text{vec}(\mathbf{X})$, then it can be shown that (see [15])

$$\text{vec}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2^T) = (\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{x}, \tag{3.11}$$

where

$$\mathbf{A}_1 \otimes \mathbf{A}_2 = \begin{bmatrix} a_{11}^{(1)}\mathbf{A}_2 & a_{12}^{(1)}\mathbf{A}_2 & \cdots & a_{1n}^{(1)}\mathbf{A}_2 \\ a_{21}^{(1)}\mathbf{A}_2 & a_{22}^{(1)}\mathbf{A}_2 & \cdots & a_{2n}^{(1)}\mathbf{A}_2 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)}\mathbf{A}_2 & a_{n2}^{(1)}\mathbf{A}_2 & \cdots & a_{nn}^{(1)}\mathbf{A}_2 \end{bmatrix},$$

with $a_{ij}^{(1)} = [\mathbf{A}_1]_{ij}$ for $i, j = 1, \ldots, n$. Thus if we define $\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2$ and $\mathbf{b} = \text{vec}(\mathbf{B})$, we can write (3.10) as $\mathbf{b} = \mathbf{A}\mathbf{x}$.

Adding iid Gaussian noise then yields $\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$. For a given $\mathbf{X}$ and matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, we can generate noisy data, choosing σ^2 so that the signal-to-noise ratio (SNR), $\|\mathbf{A}\mathbf{x}\|/\sqrt{n^2\sigma^2}$, is 50. A synthetically generate satellite image $\mathbf{X}$ and corresponding data $\mathbf{B}$ are plotted in Figure 3.2.

Before continuing, we need to present some properties of the Kronecker product. For matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{B} \in \mathbb{R}^{r\times s}$, $\mathbf{C} \in \mathbb{R}^{n\times p}$, and $\mathbf{D} \in \mathbb{R}^{s\times t}$, the following is true (see [15, 27]):

$$(\mathbf{A} \otimes \mathbf{B})^T = \mathbf{A}^T \otimes \mathbf{B}^T \tag{3.12}$$

$$(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger \tag{3.13}$$

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{A}\mathbf{C} \otimes \mathbf{B}\mathbf{D}. \tag{3.14}$$

Note that "†" denotes pseudo-inverse, which is the inverse if the matrix is invertible.

Using these formulas, one can show that if $\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2$, where $\mathbf{A}_1$ and $\mathbf{A}_2$ have full column rank, the least squares solution can be expressed

$$\mathbf{x}_{\text{LS}} = ((\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T) \otimes ((\mathbf{A}_2^T\mathbf{A}_2)^{-1}\mathbf{A}_2^T)\mathbf{b}. \tag{3.15}$$

Moreover, the SVD of $\mathbf{A}$ can be expressed in terms of the SVDs of $\mathbf{A}_1$ and $\mathbf{A}_2$. In particular, if $\mathbf{A}_1 = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$ and $\mathbf{A}_2 = \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T$, using properties (3.12) and (3.14), one can show that

$$\mathbf{A} = (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2)(\mathbf{V}_1 \otimes \mathbf{V}_2)^T. \tag{3.16}$$

Equation (3.16) is essentially the SVD of $\mathbf{A}$, except that the diagonal elements of $\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2$ are not necessarily in descending order.

Note that implementing the truncated SVD regularization method to (3.16) requires explicitly forming the diagonal of $\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2$ and reordering indices. Implementing Tikhonov regularization, on the other hand, requires very little extra work, and the result can be conveniently kept in array form. First, note that if $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are the $n \times n$ arrays such that $\mathbf{\Sigma}_1 = \text{diag}(\text{vec}(\boldsymbol{\sigma}_1))$ and $\mathbf{\Sigma}_2 = \text{diag}(\text{vec}(\boldsymbol{\sigma}_2))$, respectively, then (see Exercise 6) $\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2 = \text{diag}(\text{vec}(\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T))$, and hence

$$\mathbf{X}_\alpha = \mathbf{V}_1 \left(\left(\frac{\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T}{(\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T)^2 + \alpha\mathbf{1}} \right) \odot \mathbf{U}_1^T\mathbf{B}\mathbf{U}_2 \right) \mathbf{V}_2^T, \tag{3.17}$$

where the division and squaring of arrays is component-wise, '$\odot$' denotes component-wise multiplication, and $\mathbf{1}$ is an $n \times n$ array of 1s. Note that all operations are performed using the SVDs of $\mathbf{A}_1$ and $\mathbf{A}_2$, making (3.17) very efficient to implement.

Regularization parameter selection methods can also be efficiently implemented in the 2D separable kernel case. To illustrate, we present the UPRE function $U(\alpha)$ defined by (2.22), and the DP function $D(\alpha)$ defined by (2.30):

$$U(\alpha) = \alpha^2 \sum_{i,j=1}^{n} \left(\frac{[\mathbf{U}_1^T\mathbf{B}\mathbf{U}_2]_{ij}}{[\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T]_{ij}^2 + \alpha} \right)^2 + 2\sigma^2 \sum_{i,j=1}^{n} \frac{[\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T]_{ij}^2}{[\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T]_{ij}^2 + \alpha} \tag{3.18}$$

$$D(\alpha) = \alpha^2 \sum_{i,j=1}^{n} \left(\frac{[\mathbf{U}_1^T\mathbf{B}\mathbf{U}_2]_{ij}}{[\boldsymbol{\sigma}_2\boldsymbol{\sigma}_1^T]_{ij}^2 + \alpha} \right)^2 - n^2\sigma^2, \tag{3.19}$$

where $\mathbf{U}_1^T\mathbf{B}\mathbf{U}_2$ can be computed offline. Analogous expressions for the GCV and L-curve functions can also be derived and the corresponding regularization parameter selection methods can be efficiently implemented in this case, which we leave to the exercises.

Minimizing $U(\alpha)$ and $D(\alpha)^2$ using MATLAB's `fminbnd` function for the example shown in Figure 3.2, we obtain values of $\alpha = 0.0011$ and $\alpha = 0.0069$, respectively. For details see the accompanying code `Deblur2dSeparable.m`. The corresponding Tikhonov regularized solutions are plotted in Figure 3.3.

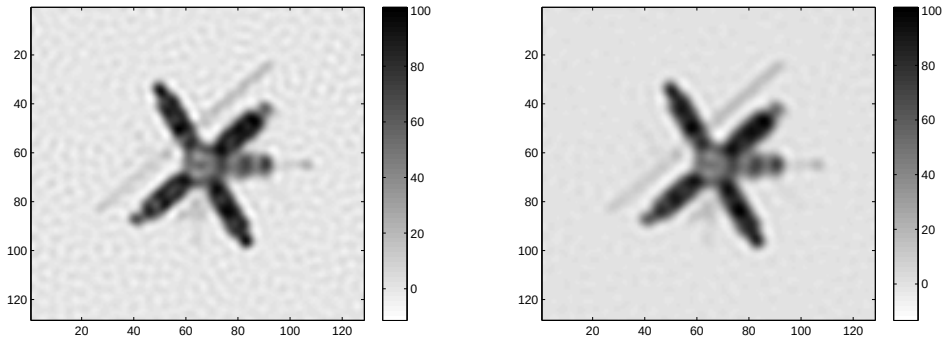

Figure 3.3. *The Tikhonov regularized solutions for two-dimensional deblurring examples. On the left is the reconstruction in the separable blur case with UPRE choice of* $\alpha = 0.0011$. *On the right is the reconstruction in the periodic boundary condition case with DP choice of* $\alpha = 0.0069$.

3.2.2 Periodic boundary conditions and the discrete Fourier transform

The assumption of a separable kernel will rarely be accurate in practice. An alternative that is often used is to assume that the unknown image x extends periodically outside of the domain $[0,1] \times [0,1]$, on which we assume b is measured; that is, the extended image is assumed to look like

x	x	x
x	x	x
x	x	x

with the central region corresponding to x on $[0,1] \times [0,1]$. We refer to this assumption as a periodic boundary condition.

Once again, we apply mid-point quadrature to (3.7), using the same computational grid as when deriving (3.8). In addition, we assume that $\xi = 1/2$, so that $a = 0$ outside of $[-1/2, 1/2] \times [-1/2, 1/2]$. As in the one dimensional case, we can express the discretized integral in summation form

$$b_{lm} = \frac{1}{n^2} \sum_{i=l-k}^{l+k} \sum_{j=m-k}^{m+k} a_{l-i,\, m-j} x_{ij}, \quad l, m = 1, \ldots, n. \tag{3.20}$$

Let $\mathbf{a}$ be the $n \times n$ discrete kernel array containing the elements $\{a_{r,s} = a(rh, sh)\}_{r,s=-n/2}^{n/2-1}$,

ordered as follows:

$$\mathbf{a} = \begin{bmatrix} a_{-n/2,n/2-1} & \cdots & a_{0,n/2-1} & \cdots & a_{n/2-1,n/2-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{-n/2,0} & \cdots & a_{0,0} & \cdots & a_{n/2-1,0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{-n/2,-n/2} & \cdots & a_{0,-n/2} & \cdots & a_{n/2-1,-n/2} \end{bmatrix}. \tag{3.21}$$

As in the one dimensional case, the periodic assumption on x allows us to construct an $n^2 \times n^2$ matrix vector equation $\mathbf{b} = \mathbf{A}\mathbf{x}$ from (3.20), with $\mathbf{A}$ a block circulant with circulant blocks (BCCB) matrix; see [15, 28] for details. Because $\mathbf{A}$ is BCCB, it can be diagonalize as follows:

$$\mathbf{A} = \mathbf{F}^*\mathbf{S}\mathbf{F}, \tag{3.22}$$

where $\mathbf{F}$ is the unitary two dimensional *discrete Fourier transform* (2D-DFT) matrix; $\mathbf{F}^*$ is the inverse 2D-DFT (2D-IDFT) matrix; and $\mathbf{S}$ is the diagonal matrix containing the eigenvalues of $\mathbf{A}$. Note that "$*$" denotes complex conjugate transpose, since $\mathbf{F}$ is a complex matrix, and that $\mathbf{F}\mathbf{F}^* = \mathbf{F}^*\mathbf{F} = \mathbf{I}$.

The 2D-DFT and 2D-IDFT, which define $\mathbf{F}$ and $\mathbf{F}^*$, respectively, are defined by their action on $n \times n$ arrays as follows: let $\mathbf{V}$ be a $n \times n$ array with ij^{th} entry v_{ij}, then

$$[\mathrm{DFT}(\mathbf{V})]_{lm} = \frac{1}{n}\sum_{i,j=1}^{n} v_{ij} e^{-\widehat{\imath}2\pi((i-1)(l-1)/n+(j-1)(m-1)/n)}$$

where $\widehat{\imath} = \sqrt{-1}$, and

$$[\mathrm{IDFT}(\mathbf{V})]_{lm} = \frac{1}{n}\sum_{i,j=1}^{n} v_{ij} e^{\widehat{\imath}2\pi((i-1)(l-1)/n+(j-1)(m-1)/n)}.$$

Then if $\mathbf{v} = \mathrm{vec}(\mathbf{V})$, $\mathbf{F}$ and $\mathbf{F}^*$ are defined by

$$\mathbf{F}\mathbf{v} = \mathrm{vec}(\mathrm{DFT}(\mathbf{V})), \quad \text{and} \quad \mathbf{F}^*\mathbf{v} = \mathrm{vec}(\mathrm{IDFT}(\mathbf{V})).$$

In MATLAB, the DFT and IDFT are implemented using the fast Fourier transform (FFT) algorithm (see [28]), with the syntax $\mathrm{DFT}(\mathbf{V}) = \frac{1}{n}\texttt{fft2}(\mathbf{V})$ and $\mathrm{IDFT}(\mathbf{V}) = n\texttt{ifft2}(\mathbf{V})$. The FFT requires that n is a power of 2, so we make this assumption, but it has the huge advantage that $\mathbf{F}\mathbf{v}$ is implemented in order $n^2 \log_2 n$ operations, as apposed to n^4 operations for standard matrix-vector multiplication. Hence a huge computational savings is gained, in addition to the fact that only the n^2 eigenvalues – and not all n^4 entries – of $\mathbf{A}$ need to be stored.

It remains to derive the eigenvalues of $\mathbf{A}$, i.e. the diagonal values of $\mathbf{S}$. The full derivation is technical and rather lengthy, so we only provide an outline and leave the remaining detail to [28]. Note that the kernel array $\mathbf{a}$ has the block form

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_{-+} & \mathbf{a}{+}{+} \\ \mathbf{a}_{--} & \mathbf{a}_{+-} \end{bmatrix},$$

where $\mathbf{a}_{-+}$ contains the a_{ij}'s with $i = -n/2, \ldots, -1$ and $j = 0, \ldots, n/2-1$; $\mathbf{a}_{++}$ contains the a_{ij}'s with $i, j = 0, \ldots, n/2-1$; $\mathbf{a}_{--}$ contains the a_{ij}'s with $i, j = -n/2, \ldots, -1$; and $\mathbf{a}_{+-}$ contains the a_{ij}'s with $i = 0, \ldots, n/2-1$ and $j = -n/2, \ldots, -1$. If we define the *circular shift* of $\mathbf{a}$ by [28]

$$\mathbf{a}_s = \begin{bmatrix} \mathbf{a}_{+-} & \mathbf{a}_{--} \\ \mathbf{a}_{++} & \mathbf{a}_{-+} \end{bmatrix}, \tag{3.23}$$

which is $\mathbf{a}_s = \texttt{fftshift}(\mathbf{a})$ in MATLAB, the diagonal of $\mathbf{S}$ is given by

$$\text{diag}(\mathbf{S}) = \text{vec}(n^2 \text{DFT}(\mathbf{a}_s)).$$

The above discussion suggests that, just as in the case of a separable blur, we can perform multiplication by $\mathbf{A}$ operating entirely in terms of arrays, since if $\widehat{\mathbf{a}}_s = n^2 \text{DFT}(\mathbf{a}_s)$, then

$$\mathbf{A}\mathbf{x} = \text{vec}(\text{IDFT}(\widehat{\mathbf{a}}_s \odot \text{DFT}(\mathbf{X}))),$$

where '$\odot$' denotes component-wise multiplication. The noise-free $n \times n$ data array $\mathbf{B}$ can therefore be expressed

$$\mathbf{B} = \text{IDFT}(\widehat{\mathbf{a}}_s \odot \text{DFT}(\mathbf{X})),$$

which we can compare with (3.10).

Adding iid Gaussian noise then yields a linear statistical model of the form $\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Using the same satellite image $\mathbf{X}$ as in the separable blur case, the same kernel a, and a noise variance σ^2 yielding SNR=50, the noisy data $\mathbf{B}$ will is visually indistinguishable from what appears in Figure 3.2.

Moreover, note that (3.22) can be used in precisely the same fashion as the SVD in previous examples, and hence, the results of Chapter 2 can once again be directly applied. First, we note that the Tikhonov regularized solution takes the form $\mathbf{x}_\alpha = \mathbf{F}^*(\mathbf{S}^*\mathbf{S} + \alpha\mathbf{I})^{-1}\mathbf{S}^*\mathbf{F}\mathbf{b}$, or in array form

$$\mathbf{X}_\alpha = \text{IDFT}\left(\left(\frac{\text{conj}(\widehat{\mathbf{a}}_s)}{|\widehat{\mathbf{a}}_s|^2 + \alpha\mathbf{1}}\right) \odot \text{DFT}(\mathbf{B})\right), \tag{3.24}$$

where 'conj' denotes complex conjugate, and $|\widehat{\mathbf{a}}_s|$ is the vector whose components are the magnitudes of the complex elements of $\widehat{\mathbf{a}}_s$.

The GCV function $G(\alpha)$ defined by (2.26) takes the form

$$G(\alpha) = \left(\alpha^2 \sum_{i,j=1}^{n} \frac{\widehat{\mathbf{B}}_{ij}^2}{(|\widehat{\mathbf{a}}_s|_{ij}^2 + \alpha)^2}\right) \Bigg/ \left(n^2 - \sum_{i,j=1}^{n} \frac{|\widehat{\mathbf{a}}_s|_{ij}^2}{|\widehat{\mathbf{a}}_s|_{ij}^2 + \alpha}\right)^2, \tag{3.25}$$

where $\widehat{\mathbf{B}} = \text{DFT}(\mathbf{B})$. The expression for the L-curve function $C(\alpha)$ does not change (see (2.32)), however $s'(\alpha)$ given by (2.33) becomes

$$s'(\alpha) = \sum_{i,j=1}^{n} \frac{-2|\widehat{\mathbf{a}}_s|_{ij}^2 \widehat{\mathbf{B}}_{ij}^2}{(|\widehat{\mathbf{a}}_s|_{ij}^2 + \alpha)^3}. \tag{3.26}$$

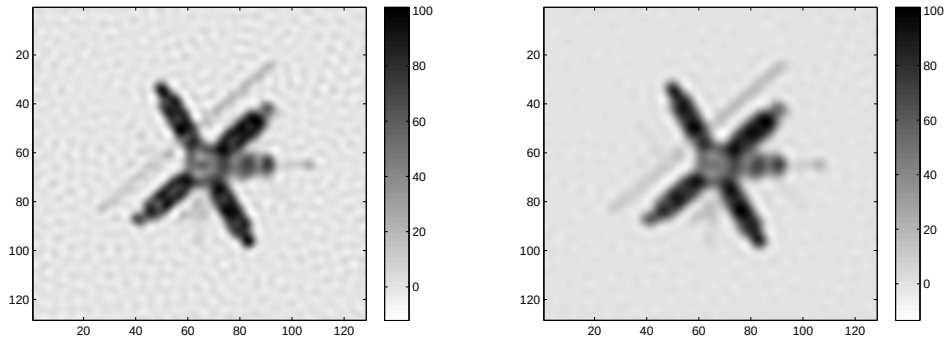

Figure 3.4. *The Tikhonov regularized solutions for two-dimensional deblurring examples. On the left is the reconstruction in the periodic boundary conditions case with GCV choice of* $\alpha = 0.0011$*. On the right is the reconstruction in the periodic boundary condition case with the L-curve choice of* $\alpha = 0.0034$*.*

Analogous expressions for the UPRE and DP functions can also be derived, and the corresponding regularization parameter selection methods can be efficiently implemented in this case. We leave this as an exercise.

Minimizing $G(\alpha)$ and $-C(\alpha)$ using MATLAB's `fminbnd` function for the example shown in Figure 3.2, we obtain values of $\alpha = 0.00034$ and $\alpha = 0.0034$, respectively. Details of this implementation can be found in `Deblur2dPeriodic.m`. The corresponding Tikhonov regularized solutions are plotted in Figure 3.4.

3.2.3 Data Driven Boundary Conditions

Assuming periodicity in the unknown x can be problematic, particularly if the true x is nonzero near the boundaries of $[0,1] \times [0,1]$. As an example, consider the deblurring problem presented by the example in Figure 3.5. The data was generated by first convolving a 256×256 true image with the same Gaussian kernel used in the previous example, then by extracting a 128×128 subimage, and finally, by adding white Gaussian noise with a variance σ^2 chosen so that the signal-to-noise is, once again, 50. We leave it to the reader in Exercise 11 to verify that the periodic boundary condition performs horribly on this example.

A way of dealing with the boundary artifact problem that still yields a matrix $\mathbf{A}$ diagonalizable by a fast transform is to assume a Neumann boundary condition. The Neumann boundary condition corresponds to the assumption that the unknown x is extended outside of $[0,1] \times [0,1]$ via a reflection about the boundary, i.e.

x_{lu}	x_u	x_{lu}
x_l	x	x_l
x_{lu}	x_u	x_{lu},

where x_l and x_u are the mirror images of x about the left and upper boundaries

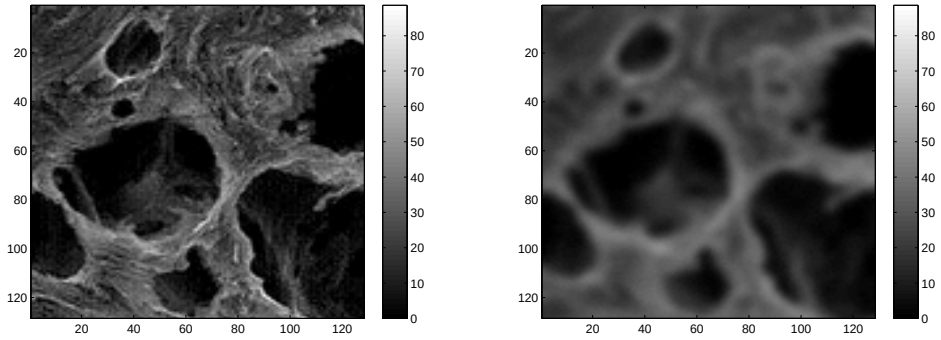

Figure 3.5. *Cell data, with regions of high relative intensity at the boundary. On the left is the true image and on the right is the blurred noisy data, generated using a larger image containing the true image on the left.*

of $[0,1] \times [0,1]$, respectively, and x_{lu} is the mirror image of x_l about its upper boundary. However, to use this approach the kernel must be symmetric, i.e., $a_{i,j} = a_{-i,j} = a_{i,-j} = a_{-i,-j}$ for all relevant i and j, which is very restrictive. If the kernel is symmetric, however, implementation of the resulting deblurring method is very similar to the periodic boundary conditions case, except that the two dimensional discrete cosine transform replaces the 2D-DFT. See [15] for details on this approach.

We propose dealing with the boundary artifact issue by instead using the data driven boundary condition idea presented for one dimensional deblurring in Section 3.1. This idea was first put forth for two dimensional imaging deblurring in [5]. Analogous to one dimension, we discretize the two dimensional convolution equation (3.7) on the extended domain $[-\xi, 1+\xi] \times [-\xi, 1+\xi]$, using the same computational grid as in the previous cases, to obtain

$$b_{lm} = \frac{1}{n^2} \sum_{i=l-k}^{l+k} \sum_{j=m-k}^{m+k} a_{l-i,\,m-j} x_{ij}, \quad l, m = 1-k, \ldots, n+k. \tag{3.27}$$

Compare (3.27) with (3.20), where the only difference is that here $-\xi \le s, t \le 1+\xi$, so that l and m range over more values.

At this point, we proceed just as in the periodic case, but on $[-\xi, 1+\xi] \times [-\xi, 1+\xi]$ rather than $[0,1] \times [0,1]$. Then the two dimensional arrays $\mathbf{X}$ and $\widehat{\mathbf{B}}$ have elements $\{x_{ij}\}_{i,j=-k+1}^{n+k}$ and $\{b_{ij}\}_{i,j=-k+1}^{n+k}$, respectively, and the kernel array $\mathbf{a}$ is of the form (3.21), but with n replaced by $n+2k$. We assume that $k = n/2$ so that, since n is a power of 2, so is $n+2k = 2n$. In this way, we obtain the matrix vector system

$$\widehat{\mathbf{b}} = \widehat{\mathbf{A}}\mathbf{x},$$

where $\widehat{\mathbf{b}}$ and $\mathbf{x}$ are $(2n)^2 \times 1$ vectors satisfying $\widehat{\mathbf{b}} = \text{vec}(\mathbf{B})$ and $\mathbf{x} = \text{vec}(\mathbf{X})$, and $\widehat{\mathbf{A}}$ is an $(2n)^2 \times (2n)^2$ BCCB matrix that is diagonalizable by the 2D-DFT. In array

form, this can be equivalently expressed

$$\widehat{\mathbf{B}} = \text{IDFT}(\widehat{\mathbf{a}}_s \odot \text{DFT}(\mathbf{X})), \tag{3.28}$$

where $\widehat{\mathbf{a}}_s = (2n)^2\text{DFT}(\texttt{fftshift}(\mathbf{a}))$. To relate the model to our observations $\mathbf{B}$ on $[0,1] \times [0,1]$, we define $\mathbf{D}$ to be the $n^2 \times (n+2k)^2$ diagonal matrix satisfying $\mathbf{D}\widehat{\mathbf{b}} = \mathbf{b}$. Then our model, in matrix-vector form and including observation noise, is given by

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}, \quad \mathbf{A} = \mathbf{D}\widehat{\mathbf{A}},$$

where as before $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

We note that the periodic boundary condition assumption on the extended domain makes it so that computing $\mathbf{A}\mathbf{x} = \mathbf{D}\widehat{\mathbf{A}}\mathbf{x}$ is highly efficient, since $\widehat{\mathbf{A}}\mathbf{x}$ is computed using (3.28). However the presence the $\mathbf{D}$ matrix makes it so that the pseudo inverse of $\mathbf{A} = \mathbf{D}\widehat{\mathbf{A}}$ is no longer efficiently computable. Thus we return to the variational formulation of Tikhonov regularization (2.5), which yields the normal equations

$$(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I})\mathbf{x} = \mathbf{A}^T\mathbf{b}. \tag{3.29}$$

Note that $\mathbf{A}^T\mathbf{A} = \widehat{\mathbf{A}}^T\mathbf{D}^T\mathbf{D}\widehat{\mathbf{A}}$, which is no longer diagonalizable by the 2D-DFT, because of the present of $\mathbf{D}$. However, we can solve the symmetric positive definite (SPD) system (3.29) using the preconditioned conjugate gradient (PCG) method.

Algorithm 3.1. PCG method for solving $\mathbf{B}\mathbf{x} = \mathbf{c}$, where $\mathbf{B}$ is SPD.
Given $\mathbf{x}^0$, SPD $\mathbf{B}$, $\mathbf{c}$, and SPD preconditioner $\mathbf{M}$, let $\mathbf{r}^0 = \mathbf{B}\mathbf{x}^0 - \mathbf{c}$, $\mathbf{b}_0 = \mathbf{M}^{-1}\mathbf{r}_0$, $\mathbf{p}^0 = -\mathbf{b}^0$, and $k = 1$. Specify some stopping tolerance ϵ. Iterate:

1. $\gamma_{k-1} = \frac{\mathbf{r}_{k-1}^T\mathbf{b}_{k-1}}{\mathbf{p}_{k-1}^T\mathbf{B}\mathbf{p}_{k-1}}$;
2. $\mathbf{x}_k = \mathbf{x}_{k-1} + \gamma_{k-1}\mathbf{p}_{k-1}$;
3. $\mathbf{r}_k = \mathbf{r}_{k-1} + \gamma_{k-1}\mathbf{B}\mathbf{p}_{k-1}$;
4. $\mathbf{b}_k = \mathbf{M}^{-1}\mathbf{r}_k$;
5. $\beta_k = \frac{\mathbf{r}_k^T\mathbf{b}_k}{\mathbf{r}_{k-1}^T\mathbf{b}_{k-1}}$;
6. $\mathbf{p}_k = -\mathbf{b}_k + \beta_k\mathbf{p}_{k-1}$;
7. Quit if $||\mathbf{r}_k|| < \epsilon$. Else set $k := k+1$ and go to step 1.

For the problem at hand, $\mathbf{B} = (\widehat{\mathbf{A}}^T\mathbf{D}^T\mathbf{D}\widehat{\mathbf{A}} + \alpha\mathbf{I})$ and $\mathbf{c} = \mathbf{A}^T\mathbf{b}$. The preconditioner $\mathbf{M}$, in general, should be an approximation of $\mathbf{B}$ that can be efficiently inverted. A natural choice for us is $\mathbf{M} = \widehat{\mathbf{A}}^T\widehat{\mathbf{A}} + \alpha\mathbf{I}$, which is diagonalizable by the 2D-DFT.

Moreover, for the UPRE and GCV function U and G, the computation of the trace of $\mathbf{A}\mathbf{A}_\alpha$ is no longer feasible, and hence a trace approximation must be computed. This can be accomplished using *randomized trace estimation*, which

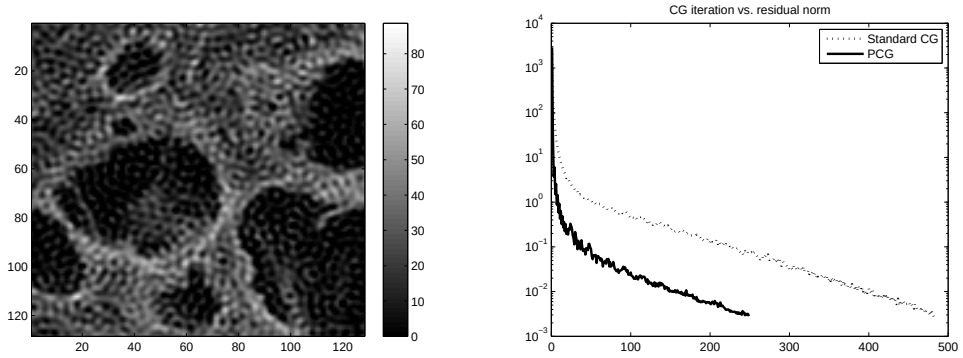

Figure 3.6. *Reconstruction and Residual norm plots for iterates from both CG and PCG applied to (3.29). The reconstruction with GCV choice of α is given on the left. On the right, the PCG iterates are denoted by the solid line, while the CG iterates are denoted by the dotted line.*

is motivated from the fact that if $\mathbf{v}$ is a white noise random vector and $\mathbf{C}$ is a symmetric matrix, then $E(\mathbf{v}^T\mathbf{C}\mathbf{v}) = \text{tr}(\mathbf{C})$ [28], where E denotes the expected value function. Thus if we choose $\mathbf{v}$ to be a realization of a white noise vector,

$$\text{tr}(\mathbf{A}\mathbf{A}_\alpha) \approx \mathbf{v}^T\mathbf{A}\mathbf{A}_\alpha\mathbf{v}. \tag{3.30}$$

The choice of $\mathbf{v}$ that minimizes the variance in this estimator is the one for which the components of $\mathbf{v}$ are independent and take on the values of 1 and -1 with equal probability (see [28] and the references therein). Note that to compute $\mathbf{A}_\alpha\mathbf{v}$ we must solve the linear system $(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I})\mathbf{w} = \mathbf{A}^T\mathbf{v}$ using PCG. Thus the computation of the trace approximation (3.30) requires an additional PCG run.

For the L-curve method, an auxiliary CG run is also necessary. In particular, in place of an SVD-type definition for $s'(\alpha)$ like (2.33), one can use

$$s'(\alpha) = \frac{2}{\alpha}\mathbf{x}_\alpha^T\mathbf{z}_\alpha, \qquad (\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I})\mathbf{z}_\alpha = \mathbf{A}^T(\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}), \tag{3.31}$$

where $\mathbf{z}_\alpha$ is computed using PCG. The proof that (3.31) and (2.33) are equivalent is left as an exercise.

In `Deblur2dDataDriven.m`, GCV is implemented and gives the value $\alpha = 4.03 \times 10^{-5}$. The corresponding Tikhonov reconstruction is shown in Figure 3.6. We leave the implementation of UPRE, DP, and L-curve in this case to the exercises. And finally, in order to illustrate the effect of the preconditioner on CG convergence, we plot the convergence history of CG ($\mathbf{M} = \mathbf{I}$) and PCG ($\mathbf{M} = \widehat{\mathbf{A}}^T\widehat{\mathbf{A}} + \alpha\mathbf{I}$) iterations, the latter of which converges in roughly half as many iterations.

3.3 Two-dimensional computed tomography

We end this chapter with the example of two dimensional computed tomography (CT). The CT inverse problem can be viewed as a two dimensional analogue of the one dimensional kernel reconstruction problem considered in Chapters 1 and 3, in the sense that the forward mathematical model involves the computation of a collection of integrals of an unknown object x.

In CT, the collection of integrals is along all lines in the plane intersecting x. Thus to define the CT forward model, we need a parametrization of all lines in the plane. A general line in the plane has the form $ax + by = c$, which can be equivalently expressed

$$\frac{a}{\sqrt{a^2+b^2}}s + \frac{b}{\sqrt{a^2+b^2}}t = \frac{c}{\sqrt{a^2+b^2}}.$$

From this we can see that if $\boldsymbol{\omega}_\theta = (\cos\theta, \sin\theta)$, and $z \in \mathbb{R}$, then the collection of all lines in the plane can be expressed

$$\boldsymbol{\omega}_\theta^T \mathbf{x} = z, \quad \mathbf{x} = (s,t) \in \mathbb{R}^2, \tag{3.32}$$

where θ need only vary over $[0, \pi)$. We will denote line (3.32) by $\ell_{z,\theta}$, and note that $\boldsymbol{\omega}_\theta$ is normal to $\ell_{z,\theta}$. Moreover, we orient $\ell_{z,\theta}$ by choosing its positive direction to be given by the parallel (perpendicular to $\boldsymbol{\omega}_\theta$) vector $\boldsymbol{\omega}_\theta^\perp = (-\sin\theta, \cos\theta)$. Then $\ell_{z,\theta} = \{z\boldsymbol{\omega}_\theta + \xi\boldsymbol{\omega}_\theta^\perp \,|\, \xi \in \mathbb{R}\}$.

We can now define the Radon transform of x. Suppose that x is a function defined on the plane, which we assume is continuous with bounded support. The Radon transform of x is then defined by

$$b(z,\theta) = \int_{-\infty}^{\infty} x(z\boldsymbol{\omega}_\theta + \xi\boldsymbol{\omega}_\theta^\perp)d\xi. \tag{3.33}$$

The inverse problem of computed tomography is to reconstruct x from its Radon transform b [7].

To discretize (3.33), we assume that the support of x is contained in in the unit circle. However, for simplicity we discretize x over $[-1,1] \times [-1,1]$, which we divide into a $2n \times 2n$ uniform pixel grid, with pixel centers given by

$$\{(s_i, t_j)\}_{i,j=1}^n = \{(i-1/2)/n, (j-1/2)/n\}_{i,j=-n+1}^n.$$

We denote $x_{ij} = x(s_i, t_j)$ and assume that x is piecewise constant within each pixel; specifically

$$x(s,t) = x_{ij}, \quad \text{for} \quad (s,t) \in P_{ij},$$

where $P_{ij} = \{(s,t) \in \mathbb{R}^2 \,|\, (i-1)/N \le s \le i/N, \text{ and } (j-1)/N \le t \le j/N\}$ and $-n+1 \le i,j \le n$.

Next we discretize the z and θ variables. We also do this uniformly, first with z, which varies between -1 and 1 since x is contained in the unit circle:

$$z_k = -1 + 2\left(\frac{k-1}{m_2}\right), \quad k = 1, \ldots, m_2.$$

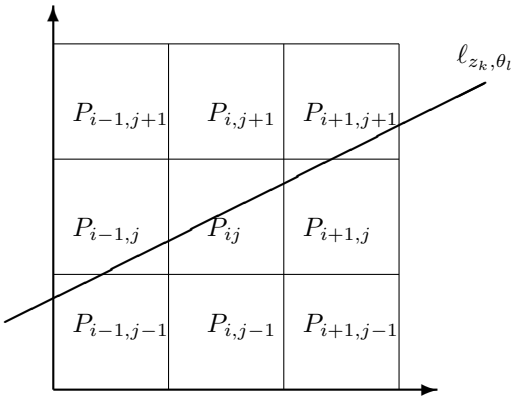

Figure 3.7. *The line ℓ_{z_l,θ_k} is z_l units from $(0,0)$ in the direction $\boldsymbol{\omega}_{\theta_k} = (\cos\theta_k, \sin\theta_k)$. Let $\Delta\ell_{kl}^{ij}$ be the intersection length of ℓ_{z_l,θ_k} with P_{ij} and x_{ij} be the value of x at the midpoint of P_{ij}. Then if $\mathbf{X}$ is the $n \times n$ array with ij^{th} entry x_{ij}, the discrete Radon transform of $\mathbf{X}$ is defined by $b_{kl} = \sum_{ij=1}^{n} \Delta\ell_{kl}^{ij} x_{ij}$.*

Next, assuming full angle data, $\theta \in [0, \pi)$, and hence

$$\theta_l = \frac{(l-1)}{m_1}\pi, \quad l = 1, \ldots, m_1.$$

Then, if we define $b_{kl} = b(z_k, \theta_l)$, for $k = 1, \ldots, m_1$ and $l = 1, \ldots, m_2$, from (3.33) we obtain

$$b_{kl} = \sum_{i,j=1}^{n} \Delta\ell_{kl}^{ij} x_{ij}, \quad k = 1, \ldots, m_1, \quad l = 1, \ldots, m_2, \tag{3.34}$$

where $\Delta\ell_{kl}^{ij}$ is the intersection length of line ℓ_{z_k,θ_l} with pixel P_{ij}. This is illustrated in Figure 3.7.

The system of equations defined by (3.34) characterizes the discrete Radon transform. Letting $\mathbf{B}$ be the $m_1 \times m_2$ array with kl^{th} element b_{kl}, $\mathbf{b} = \text{vec}(\mathbf{B})$, and $\mathbf{x} = \text{vec}(\mathbf{X})$, we obtain the $m_1 m_2 \times n^2$ system of equations $\mathbf{b} = \mathbf{A}\mathbf{x}$ from (3.34). Using MATLAB's built in functionality, we generate a synthetic true object $\mathbf{X}$, also called the Shepp-Logan phantom, on the left in Figure 3.8, and, using the `radon` function, blurred noisy data $\mathbf{B}$ (called a sinogram in CT problems) on the right in Figure 3.8. Independent and identically distributed white noise was added to $\mathbf{B}$ so that the SNR was 50.

We end the chapter with a description of Kaczmarz's method, which is often used for solving the CT inverse problem. The algorithm is straightforward and

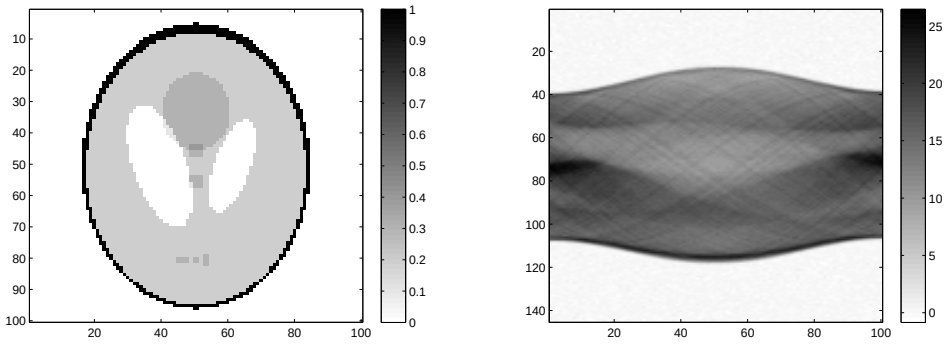

Figure 3.8. *On the left is the Shepp-Logan phantom* $\mathbf{x}$ *generated in MATLAB. On the right is the discrete Radon transform of* $\mathbf{x}$ *with Gaussian noise added.*

works well in practice. First, we express $\mathbf{Ax} = \mathbf{b}$ in the modified form

$$\begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_{n^2}^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m_1 m_2} \end{bmatrix},$$

where $\mathbf{a}_i$ is the i^{th} row of $\mathbf{A}$, from which we see the n^2 hyper-planes $\mathbf{a}_i^T\mathbf{x} = b_i$ for $i = 1, \ldots, n^2$.

The idea of standard Kaczmarz's method is to start with an approximate solution $\mathbf{x}_0$ and project it onto hyper-plane 1 to obtain $\mathbf{x}_1$, then project $\mathbf{x}_1$ onto hyper-plane 2 to obtain $\mathbf{x}_2$, etc., until you compute $\mathbf{x}_{n^2}$ and then repeat the process if necessary. It remains to compute the formula for projecting a vector $\widehat{\mathbf{x}}$ onto a hyper-plane $\mathbf{a}^T\mathbf{x} = b$. We note that $\mathbf{a}$ is the normal to the hyperplane, and that $(b/\|\mathbf{a}\|^2)\mathbf{a}$ is the vector that orthogonally projects vectors on $\mathbf{a}^T\mathbf{x} = 0$ onto $\mathbf{a}^T\mathbf{x} = b$. Moreover, the orthogonal projection of $\widehat{\mathbf{x}}$ onto $\mathbf{a}^T\mathbf{x} = 0$ is given by $\widehat{\mathbf{x}} - (\widehat{\mathbf{x}}^T\mathbf{a}/\|\mathbf{a}\|^2)\mathbf{a}$. Combining these two results, we obtain Kaczmarz's iteration: choose $\mathbf{x}_0$, then

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \frac{b_i - \mathbf{a}_i^T\mathbf{x}_{i-1}}{\|\mathbf{a}_i\|^2}\mathbf{a}_i, \quad \text{for } i = 1, \ldots, n^2. \tag{3.35}$$

The iteration can be continued by setting $\mathbf{x}_0 = \mathbf{x}_{n^2}$ and repeating (3.35). It can be shown that the iterates converge to the minimum norm least squares solution of $\mathbf{Ax} = \mathbf{b}$. We leave it to the reader to implement (3.35) for computing a solution to the image reconstruction problem presented by Figure 3.8 (see Excercise 14).

Exercises

3.1. Modify `OnedDeblurBCs.m` so that it implements GCV and UPRE regularization parameter selection methods. How do these parameter selection methods perform in terms of the visual quality of the regularized reconstructions?

3.2. *Circulant matrices.* These questions involve the matrix in (3.4) when a periodic boundary condition is assumed. Assume that n is odd and $k = (n-1)/2$.

a. Write down the $n \times n$ linear system $\mathbf{b} = \mathbf{A}\mathbf{x}$ that results when a periodic boundary condition is assumed in (3.4). Observe that the matrix $\mathbf{A}$ is circulant, i.e., each row of $\mathbf{A}$ is the circular shift of its previous row.

b. If $\mathbf{a} = (a_{-k} \ldots, a_{-1}, a_0, a_1, \ldots, a_k)$ is the kernel, show that the first column of $\mathbf{A}$ is $\mathbf{c} = \texttt{fftshift}(\mathbf{a}) = (a_0, a_1, \ldots, a_k, a_{-k} \ldots, a_{-1})$.

c. Since $\mathbf{A}$ is circulant, $\mathbf{A} = \mathbf{F}^* \text{diag}(\widehat{\mathbf{c}})\mathbf{F}$, where $\mathbf{F}\mathbf{x} = \texttt{fft}(\mathbf{x})$ in MATLAB, $\widehat{\mathbf{c}} = \mathbf{F}\mathbf{c}$, and $\mathbf{F}^*\mathbf{x} = \texttt{ifft}(\mathbf{x})$ [28]. For the PSF generated in `NonSymmetricPSF.m`, generate $\mathbf{A}$ using this spectral representation. Then plot the matrix using MATLAB's `mesh`.

d. Alternatively, you can generate $\mathbf{A}$ using the MATLAB command `A = gallery('circul',r)`, where $\mathbf{r}$ is the first row of $\mathbf{A}$. Do this and verify that you get the same matrix as in part c.

e. Modify `OnedDeblurBCs.m` so that a reconstruction is computed corresponding to the assumption that the values of x extend periodically outside of [0.15,0.85]. How does the reconstruction obtained with the periodic boundary condition compare to those obtained with the data driven and zero boundary conditions are already implemented?

3.3. *Toeplitz matrices.* These questions involve the matrix in (3.4) when a zero boundary condition is assumed. Assume that $k = n - 1$.

a. Write down the $n \times n$ linear system $\mathbf{b} = \mathbf{A}\mathbf{x}$ that results when a zero boundary condition is assumed, i.e., $(x_{-k+1}, \ldots, x_0) = \mathbf{0}$ and $(x_{n+1}, \ldots, x_{n+k}) = \mathbf{0}$. Observe that the matrix $\mathbf{A}$ is a Toeplitz matrix, i.e., is constant along all of its sub-diagonals.

b. If $\mathbf{a} = (a_{-k} \ldots, a_{-1}, a_0, a_1, \ldots, a_k)$ is the kernel, compute the first row and first column of $\mathbf{A}$. Call them $\mathbf{r}$ and $\mathbf{c}$.

c. An $n \times n$ Toeplitz matrix can be embedded within a $2n \times 2n$ circulant matrix. Find the $2n \times 2n$ circulant embedding of the matrix $\mathbf{A}$ computed in part a. Then use the spectral representation of $\widehat{\mathbf{A}}$ as in exercise 2.c to generate $\mathbf{A}$.

d. For the PSF generated in `NonSymmetricPSF.m`, generate $\mathbf{A}$ using MATLAB's `toeplitz` function, then verify that you get the same $\mathbf{A}$ as in part c.

3.4. *Toplitz-plus-Hankel matrices.* These questions involve the matrix vector equation (3.4) and the corresponding boundary conditions. Assume that n is odd and $k = (n-1)/2$.

a. Write down the $n \times n$ linear system $\mathbf{b} = \mathbf{A}\mathbf{x}$ that results when a Neumann boundary condition is assumed, i.e., $(x_{-k+1}, \ldots, x_0) = (x_k, \ldots, x_1)$ and $(x_{m+1}, \ldots, x_{m+k}) = (x_m, \ldots, x_{m-k+1})$. Observe that the matrix $\mathbf{A}$ can be written as a Toeplitz matrix plus a Hankel (constant skew diagonals) matrix, but only if the kernel $\mathbf{a} = (a_{-k} \ldots, a_{-1}, a_0, a_1, \ldots, a_k)$ is symmetric, i.e. $a_i = a_{-i}$ for $i = 1, \ldots, k$.

b. If $\mathbf{a} = (a_{-k} \ldots, a_{-1}, a_0, a_1, \ldots, a_k)$ is the kernel, compute the first column of $\mathbf{A}$. Call it $\mathbf{c}$.

c. Generate $\mathbf{A}$ using the MATLAB functions `toeplitz` and `hankel`.

d. Modify `OnedDeblurBCs.m` so that a reconstruction is computed corresponding to the assumption that the values of x extend reflectively (i.e. with a Neumann boundary condition) outside of [0.15,0.85]. How does the reconstruction obtained with the periodic boundary condition compare to those obtained with the data driven and zero boundary conditions are already implemented, and also the periodic boundary condition if you did exercise 2.c?

3.5. a. Verify that mid-point quadrature applied in the general 1D deconvolution case yields (3.3).

b. Verify that mid-point quadrature applied in the general 2D deconvolution case with a zero boundary condition yields (3.8).

c. Verify that (3.9) can be expressed in matrix-array form as (3.10).

3.6. a. Use the Kronecker product properties (3.12)-(3.14) to prove (3.15) and (3.16).

b. Suppose that $\mathbf{\Sigma}_1 = \text{diag}(\boldsymbol{\sigma}_1)$ and $\mathbf{\Sigma}_2 = \text{diag}(\boldsymbol{\sigma}_2)$. Prove that $\mathbf{\Sigma}_1 \otimes \mathbf{\Sigma}_2 = \text{diag}(\text{vec}(\boldsymbol{\sigma}_2 \boldsymbol{\sigma}_1^T))$.

c. Using (3.11), (3.16), and part a, prove (3.17).

3.7. a. Derive the formulas for GCV analogous to (3.18). Add lines of code to `Deblur2dSeparable.m` so that it implements GCV.

b. Derive the formula for the L-curve method analogous to (2.32), (2.33). Add lines of code to `Deblur2dSeparable.m` so that it implements the L-curve method.

Report the value(s) of α and plot the corresponding Tikhonov regularized solution(s).

3.8. a. Derive the formulas for UPRE analogous to (3.18). Add lines of code to `Deblur2dPeriodic.m` so that it implements UPRE.

b. Derive the formula for the DP method analogous to (2.30). Add lines of code to `Deblur2dPeriodic.m` so that it implements the DP regularization parameter selection method.

Report the value(s) of α and plot the corresponding Tikhonov regularized solution(s).

3.9. a. Modify `Deblur2dDataDriven.m` so that it implements DP regularization parameter selection.

b. Modify `Deblur2dDataDriven.m` so that it implements UPRE regularization parameter selection.

c. Modify `Deblur2dDataDriven.m` using (3.31) so that it implements L-curve regularization parameter selection.

Report the value(s) of α and plot the corresponding Tikhonov regularized solution(s).

3.10. Use the SVD to show that (3.31) and (2.33) are equivalent definitions for $s'(\alpha)$, which is needed for evaluating the curvature (2.32) for the L-curve regularization parameter selection method.

3.11. *An example in which the periodic boundary conditions assumption is poor.* Implement Tikhonov regularization with a periodic boundary condition and GCV stopping rule on the example given in `PeriodBCtest.m`. Note that the blurred image is the central 128×128 pixels of a 256×256 image and hence does not contain boundary artifacts from the periodic boundary conditions assumed for the forward model. Perform the deblurring on the 128×128 subimage. Plot a picture of the deblurred image.

3.12. Modify `Deblur2DataDriven.m` so that the truncated Landweber iteration, introduced in Chapter 2, is used for solving the deblurring problem. Use the DP stopping rule, i.e. choose the first k such that $\|\mathbf{A}\mathbf{x}_k - \mathbf{b}\|^2 \leq n^2\sigma^2$.

3.13. *Testing the accuracy of the randomized trace estimate.* Consider the 2D periodic boundary conditions case with a $n = 128$. Compute a single realization $\mathbf{v} \in \mathbb{R}^{n\times n}$ with independent entries equal to +1 or -1 with equal probability. Then create a grid for the Tikhonov regularization parameter α using `logspace` in MATLAB and plot the following two functions together on this grid: $\bar{t}(\alpha) = \mathbf{v}^T\mathbf{A}\mathbf{A}_\alpha\mathbf{v}$ and $t(\alpha) = \mathrm{tr}(\mathbf{A}\mathbf{A}_\alpha) = \sum_{i,j=1}^{n}[|\widehat{\mathbf{a}}_s|]_{ij}^2/([|\widehat{\mathbf{a}}_s|]_{ij}^2 + \alpha)$. Note that multiplication by $\mathbf{A}_\alpha$ is defined in equation (3.24). Is $\bar{t}(\alpha)$ a good approximation of $t(\alpha)$? Use your graph to support your answer.

3.14. Modify `Tomography.m` to that it implements Kaczmarz's Method (3.35). How many sweeps through all of the indices, i.e. implementations of (3.35), does it take to obtain a good reconstruction?

3.15. As was mentioned in Section 3.2.1, when a zero boundary condition is assumed and the kernel is non-separable, two dimensional convolution can be written as $\mathbf{b} = \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is block toeplitz with toeplitz blocks (BTTB). If $\mathbf{A}$ is $n^2 \times n^2$, it can be embedded within a $(2n)^2 \times (2n)^2$ BCCB matrix for fast matrix-vector multiplication; see [28] for details. In the directory `Chapter3/Dirichlet` are all of the pieces to compute the Tikhonov regularization solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$ when $\mathbf{A}$ is BTTB. The file `Deblur2dDirichlet.m` generates blurred noisy data, using the function `Amult_Dirichlet.m` for computing $\mathbf{A}\mathbf{x}$. To solve $(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I})\mathbf{x} = \mathbf{A}^T\mathbf{b}$, you will need to use `CG.m` and `Bmult_Dirichlet.m`, which performs multiplication by $\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}$. And finally, once you have CG up and running, you can implement either GCV or DP regularization parameter selection with the included codes. Provide enough output to show that you've got things working.

Chapter 4

Bayes' Law and Markov Random Field Priors

In Chapter 1, we motivated the need for regularization by showing that for inverse problems, the least squares solution has high variance in directions corresponding to the right singular vectors $\mathbf{v}_i$ with very small singular values. This model-based observation motivated Chapter 2, where we discussed the notion of 'filtering' the problematic directions ($\mathbf{v}_i$'s) contained within the model so that the new, regularized solutions have smaller variance, but with the negative side effect of bias in the regularized solution.

4.1 Bayes Law and Regularization

Another approach, which in many cases yields regularization methods that are equivalent to those obtained via the filtered SVD, is to instead model the unknown $\mathbf{x}$ as a random quantity with a probability density function $p(\mathbf{x}|\delta)$ known as the *prior probability density*, or simply the *prior*, with δ a positive scaling parameter. The prior should not be viewed in the same fashion as the probability density for the data model (1.1), which is given by

$$p(\mathbf{b}|\mathbf{x},\lambda) \propto \lambda^{n/2} \exp\left(-\frac{\lambda}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2\right), \tag{4.1}$$

where $\lambda = 1/\sigma^2$ is known as the noise *precision*. In contrast, $p(\mathbf{x}|\delta)$ encodes both prior knowledge about the properties of $\mathbf{x}$, as well as uncertainty about $\mathbf{x}$, making the choice of the prior a much more subjective enterprize.

Given a particular prior $p(\mathbf{x}|\delta)$, *Bayes' Law* can be written

$$p(\mathbf{x}|\mathbf{b},\lambda,\delta) \propto p(\mathbf{b}|\mathbf{x},\lambda)p(\mathbf{x}|\delta), \tag{4.2}$$

which defines the *posterior density function* $p(\mathbf{x}|\mathbf{b},\lambda,\delta)$. Maximizing the posterior density function $p(\mathbf{x}|\mathbf{b},\lambda,\delta)$ with respect to $\mathbf{x}$ yields the *maximum a posteriori* (MAP) estimator $\mathbf{x}_{\text{MAP}}$. Note that the computation of $\mathbf{x}_{\text{MAP}}$ requires estimates of the parameters λ and δ. In the next chapter, we will relax this requirement by

assuming that λ and δ are random variables, in which case Bayes' Law (4.2) must be modified.

For our first example of a Bayesian posterior density function, recall that for typical inverse problems, the least squares solution $\mathbf{x}_{\text{LS}}$ often has very large norm (see Figure 1.3 for the 1D test case in Chapter 1). We can incorporate our prior knowledge that the norm of $\mathbf{x}$ is not-too-large by assuming $\mathbf{x} \sim N(\mathbf{0}, \delta^{-1}\mathbf{I})$, for $\delta > 0$. This yields the prior

$$p(\mathbf{x}|\delta) \propto \exp\left(-\frac{\delta}{2}\|\mathbf{x}\|^2\right). \tag{4.3}$$

The MAP estimator is then the minimizer of $-\ln p(\mathbf{x}|\mathbf{b}, \lambda, \delta)$, i.e.,

$$\mathbf{x}_{\text{MAP}} = \arg\min_{\mathbf{x}} \left\{ \frac{\lambda}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \frac{\delta}{2}\|\mathbf{x}\|^2 \right\}, \tag{4.4}$$

which is equivalent to the minimization formulation of Tikhonov regularization defined by (2.5) with $\alpha = \delta/\lambda$. Thus an estimate of the α is required in order to compute $\mathbf{x}_{\text{MAP}}$. This can be obtained using one of the regularization parameter selection methods from Chapter 2.

Note that the prior (4.3) corresponds to the assumption that $x_i \overset{iid}{\sim} N(0, \delta^{-1})$, implying that the values x_i and x_j are unrelated for all i and j. However, we know that for typical images $\mathbf{x}$, the intensity values x_i and x_j at pixels i and j will typically be related if i and j are spatial 'neighbors'. This motivates the use of priors that take into account spatial structure in the image $\mathbf{x}$.

4.2 Gaussian Markov random field priors

In the pioneering work of Besag [6], an approach known as *conditional autoregression* was introduced for defining statistical models of a spatially distributed parameter $\mathbf{x}$. In this approach, probability distributions are assigned for the *full conditionals* $x_i|\mathbf{x}_{-i}$, where $\mathbf{x}_{-i}$ denotes all elements in $\mathbf{x}$ except x_i. We consider the special case in which the conditionals are of the form

$$x_i|\mathbf{x}_{-i} \sim \mathcal{N}\left(\sum_{j \neq i} \beta_{ij} x_j, \kappa_i^{-1}\right), \tag{4.5}$$

where $\beta_{ij}\kappa_i = \beta_{ji}\kappa_j$ for all i and j. Then, if we define

$$[\mathbf{Q}]_{ij} = \begin{cases} \kappa_i, & i = j, \\ -\kappa_i\beta_{ij}, & i \neq j, \end{cases} \tag{4.6}$$

for $i = 1, \ldots, n$, and make the additional assumption that $\mathbf{Q}$ is positive definite, by [25, Theorem 2.6] the random vector $\mathbf{x}$ defined by (4.5) is Gaussian with zero mean and *precision* (inverse-covariance) matrix $\mathbf{Q}$; that is

$$p(\mathbf{x}) = (2\pi)^{-n/2}|\mathbf{Q}|^{1/2} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}\right). \tag{4.7}$$

In this case, $\mathbf{x}$ is known as a *Gaussian Markov random field* (GMRF).

The sets $\partial_i = \{j \neq i \mid \beta_{ij} \neq 0\}$, for $i = 1, \ldots, n$, implicitly define the unique neighborhood system associated with (4.5)-(4.7). Specifically, note that for $j \neq i$, $[\mathbf{Q}]_{ij} \neq 0$ if and only if $j \in \partial_i$. Thus the sparsity structure of $\mathbf{Q}$ determines the neighborhood system, and vice versa. Moreover, if $\mathbf{x}_{\partial_i} = \{x_i \mid i \in \partial_i\}$, we have $p(x_i|\mathbf{x}_{-i}) = p(x_i|\mathbf{x}_{\partial_i})$ for all i, and hence it suffices to define the 'neighborhood conditionals' (see [25, Theorem 2.4])

$$x_i|\mathbf{x}_{\partial_i} \sim \mathcal{N}\left(\sum_{j\in\partial_i} \beta_{ij} x_j, \kappa_i^{-1}\right) \tag{4.8}$$

to obtain (4.6), (4.7).

As an example, suppose $\kappa_i = \delta n_i/h^2$, where $n_i = |\partial_i|$ and $\delta > 0$, and $\beta_{ij} = 1/n_i$ for $j \in \partial_i$ and 0 otherwise. Then

$$x_i|\mathbf{x}_{\partial_i} \sim \mathcal{N}\left(\bar{x}_{\partial_i}, (\delta n_i/h^2)^{-1}\right), \tag{4.9}$$

where $\bar{x}_{\partial_i} = \frac{1}{n_i}\sum_{i\in\partial_i} x_i$ and

$$[\mathbf{Q}]_{ij} = \frac{\delta}{h^2}\begin{cases} n_i & i = j, \\ -1 & j \in \partial_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

We now apply this GMRF in both one-dimensional (1D) and two-dimensional (2D) cases.

In 1D, we assume a uniform grid on $[0, 1]$ with n vertices $\{1, \ldots, n\}$ at locations $\{s_i = (i - 1/2)/n\}_{i=1}^n$, define $x_i = x(s_i)$, for $i = 1, \ldots, n$, and assume zero boundary conditions (BCs), so that $x_0 = x_{n+1} = 0$. We choose the simple first order neighborhood system $\partial_i = \{i - 1, i + 1\}$ for $i = 1, \ldots, n$. Thus $n_i = 2$ for all i and from (4.10), $\mathbf{Q} = \delta\mathbf{L}_{1\mathrm{D}}$ with

$$\mathbf{L}_{1\mathrm{D}} = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}_{n\times n}. \tag{4.11}$$

Note that $\mathbf{Q}$ is symmetric, positive definite (SPD), as required, and $\mathbf{L}_{1\mathrm{D}}$ is the standard discretization of the 1D second derivative with zero BCs; see, e.g., [18] or any of a number of standard texts on numerical analysis. On the left in Figure 4.1, we've plotted five realizations from this GMRF.

In 2D, we assume a uniform grid on $[0, 1] \times [0, 1]$ with n^2 vertices $\{(i, j)\}_{i,j=1}^n$ at locations $\{(s_i, t_j) = ((i - 1/2)/n, (j - 1/2)/n)\}_{i,j=1}^n$, define $x_{ij} = x(s_i, t_j)$, for $i, j = 1, \ldots, n$, and assume Dirichlet BCs so that $x_{0,j} = x_{n+1,j} = x_{i,0} = x_{i,n+1} = 0$. Again, we choose the simple first order neighborhood system

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

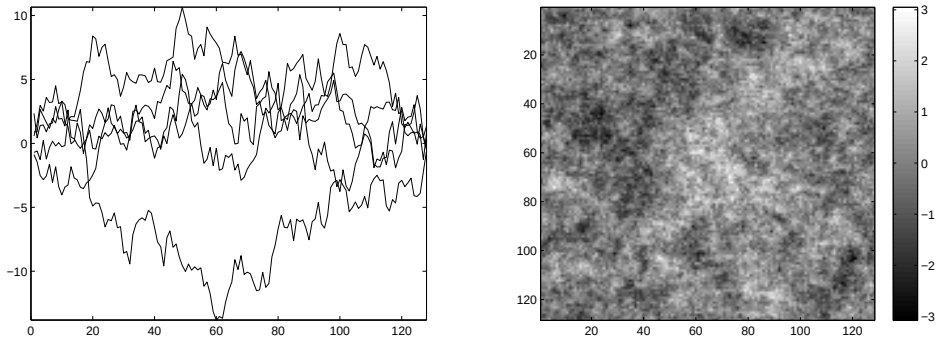

Figure 4.1.

Then $n_{ij} = |\partial_{ij}| = 4$ for $i, j = 1, \ldots, n$, and hence (4.9) takes the form

$$x_{ij}|\mathbf{x}_{\partial_{ij}} \sim \mathcal{N}\left(\bar{x}_{\partial_{ij}}, (4\delta/h^2)^{-1}\right), \tag{4.12}$$

where $\bar{x}_{\partial_i} = \frac{1}{4}(x_{i-1,j}+x_{i+1,j}+x_{i,j-1}+x_{i,j+1})$. After reordering the array $\{x_{ij}\}_{i,j=1}^{n}$ by stacking its columns, we obtain the precision matrix $\mathbf{Q} = \delta\mathbf{L}_{2\mathrm{D}}$, with

$$\mathbf{L}_{2\mathrm{D}} = \frac{1}{h^2}\begin{bmatrix} \widetilde{\mathbf{L}} & -\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ -\mathbf{I} & \widetilde{\mathbf{L}} & -\mathbf{I} & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \mathbf{0} \\ \vdots & \ddots & -\mathbf{I} & \widetilde{\mathbf{L}} & -\mathbf{I} \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \widetilde{\mathbf{L}} \end{bmatrix}_{n^2\times n^2}, \tag{4.13}$$

where $\mathbf{I}$ the $n \times n$ identity matrix, $\mathbf{0}$ the $n \times n$ zero matrix, and

$$\widetilde{\mathbf{L}} = \begin{bmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 4 & -1 \\ 0 & \cdots & 0 & -1 & 4 \end{bmatrix}_{n\times n}.$$

Note that $\mathbf{Q} = \delta\mathbf{L}_{2\mathrm{D}}$ is SPD and that $\mathbf{L}_{2\mathrm{D}}$ is the standard discretization of the 2D discrete negative-Laplacian with zero BCs [18]. On the right in Figure 4.1, we've plotted one realization from this GMRF.

It can be shown (see the exercises) that $\mathbf{L}_{2\mathrm{D}}$ can be alternatively expressed

$$\mathbf{L}_{2\mathrm{D}} = \mathbf{L}_{1\mathrm{D}} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{L}_{1\mathrm{D}}, \tag{4.14}$$

where $\mathbf{L}_{1\mathrm{D}}$ is defined in (4.11), which allows for an easy computation of the eigenvalues of $\mathbf{L}_{2\mathrm{D}}$ from those of $\mathbf{L}_{1\mathrm{D}}$.

In both the 1D and 2D cases, the prior (4.7) can be expressed

$$p(\mathbf{x}|\delta) \propto \delta^{N/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T\mathbf{L}\mathbf{x}\right), \tag{4.15}$$

where $N = n$ and $\mathbf{L} = \mathbf{L}_{1\mathrm{D}}$ in 1D, and $N = n^2$ and $\mathbf{L} = \mathbf{L}_{2\mathrm{D}}$ in 2D. This prior results in a quadratic regularization function with regularization parameter δ. Thus GMRFs provide a convenient way to relate pixel-wise distributional assumptions of the form (4.9) and (4.12) with regularization functions of the form $\frac{\delta}{2}\mathbf{x}^T\mathbf{L}\mathbf{x}$.

4.2.1 Intrinsic GMRFs

The assumption that $\mathbf{Q}$ is positive definite is often too restrictive. So-called *intrinsic GMRFs* (IGMRFs) allow for a symmetric positive *semi*-definite matrix $\mathbf{Q}$, so that $\mathbf{Q}$ is allowed to have zero eigenvalues [25]. We will focus on first-order IGMRFs, which have only one zero eigenvalue with eigenvector $\mathbf{1}$, i.e. $\mathbf{Q1} = \mathbf{0}$. Moreover, as above, the $N \times N$ precision matrix $\mathbf{Q} = \delta\mathbf{L}$ will be tied directly to pixel-wise conditional densities of the form (4.9) in 1D and (4.12) in 2D. In both cases, the prior takes the form

$$p(\mathbf{x}|\delta) \propto \delta^{(N-1)/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T\mathbf{L}\mathbf{x}\right). \tag{4.16}$$

We note that in (4.16), $\delta\mathbf{L}$ is not technically a precision matrix, since it has a zero eigenvalue, and hence, $p(\mathbf{x}|\delta)$ not technically a probability density function, however we will continue to call these the precision and prior, respectively, following the convention of [25].

Periodic boundary conditions

A standard example of an IGMRF prior results if, as in the one dimensional example above, we assume conditional distributions (4.9) (and hence precision (4.10)) but with periodic boundary conditions for $\mathbf{x}$. This amounts to assuming the same neighborhood system and computational grid, but that $x_{n+1} = x_1$ and $x_0 = x_n$. The precision matrix then has the form $\mathbf{Q} = \delta\mathbf{L}_{1\mathrm{D}}$ with

$$\mathbf{L}_{1\mathrm{D}} = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \cdots & -1 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ -1 & \cdots & 0 & -1 & 2 \end{bmatrix}_{n\times n}. \tag{4.17}$$

Note then that $\mathbf{Q1} = \mathbf{0}$. The corresponding prior is given by (4.16) with $N = n$.

In two dimensions, we proceed as in the two dimensional, zero boundary conditions case above, with the exception that $x_{0,j} = x_{n,j}$, $x_{n+1,j} = x_{1,j}$, $x_{i,0} = x_{i,n}$, and $x_{i,n+1} = x_{i,1}$ for $1 \le i, j \le n$. Then $\mathbf{Q} = \delta\mathbf{L}_{2D}$, where $\mathbf{L}_{2\mathrm{D}}$ can be expressed

in terms of the Kronecker product formula (4.14) with $\mathbf{L}_{1D}$ given by (4.17). The corresponding prior is given by (4.16) with $N = n^2$. Note also that in this case, $\mathbf{L}_{2D}$ has block circulant structure, and hence, can be diagonalized by the 2D-DFT, as discussed in Chapter 3.

The realizations from these GMRFs are very similar in appearance to those in Figure 4.1, except that they satisfy periodic (rather than zero) boundary conditions.

Independent and identically distributed increments

Another standard example of an IGMRF results when, given the uniform computational grid on [0,1] used in the above 1D examples, the *increments* $\Delta x_i = x_{i+1} - x_i$ are assumed to be of the form

$$\Delta x_i \overset{iid}{\sim} \mathcal{N}(0, \delta^{-1}), \quad i = 1, \ldots, n-1. \tag{4.18}$$

Then the density function for $\mathbf{x}$ has the form

$$\begin{aligned} p(\mathbf{x}|\delta) &\propto \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\sum_{i=1}^{n-1} \Delta x_i^2\right) \\ &\propto \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\sum_{i=1}^{n-1} (x_{i+1} - x_i)^2\right) \\ &= \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T(\mathbf{D}^T\mathbf{D})\mathbf{x}\right), \end{aligned} \tag{4.19}$$

where

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}_{(n-1)\times n}. \tag{4.20}$$

Note that (4.19) has the form (4.16) with $N = n$ and $\mathbf{L} = \mathbf{D}^T\mathbf{D}$.

It is easy to verify that $\mathbf{D}^T\mathbf{D}$ has the same form as $\mathbf{L}_{1D}$ in (4.11), but with $[\mathbf{L}_{1D}]_{11} = [\mathbf{L}_{1D}]_{nn} = 1$. Thus the increment model (4.18) corresponds to the autoregressive model of the form (4.9) with neighborhood system $\partial_1 = \{2\}$, $\partial_i = \{i-1, i+1\}$ for $i = 2, \ldots, n-1$, and $\partial_n = \{n-1\}$. It is left as an exercise that this is also the discrete second derivative matrix on the computational grid $\{i/(n+1)\}_{i=1}^n$ with Neumann (zero flux) BCs, $x'(0) = x'(1) = 0$.

In 2D, we define $\Delta_h x_{ij} = x_{i+1,j} - x_{ij}$ and $\Delta_v x_{ij} = x_{i,j+1} - x_{ij}$, and assume

$$\Delta_h x_{ij}, \Delta_v x_{ij} \overset{iid}{\sim} \mathcal{N}(0, \delta^{-1}), \quad i, j = 1, \ldots, n-1. \tag{4.21}$$

Then the density function for $\mathbf{x}$ has the form

$$\begin{aligned}
p(\mathbf{x}|\delta) &\propto \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\left(\sum_{j=1}^{n}\sum_{i=1}^{n-1}\Delta_h x_{ij}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n-1}\Delta_v x_{ij}^2\right)\right) \\
&\propto \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\left(\sum_{j=1}^{n}\sum_{i=1}^{n-1}(x_{i+1,j}-x_{ij})^2 + \sum_{i=1}^{n}\sum_{j=1}^{n-1}(x_{i,j+1}-x_{ij})^2\right)\right) \\
&= \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\left(\|(\mathbf{I}\otimes\mathbf{D})\mathbf{x}\|^2 + \|(\mathbf{D}\otimes\mathbf{I})\mathbf{x}\|^2\right)\right) \\
&= \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T\left(\mathbf{I}\otimes(\mathbf{D}^T\mathbf{D}) + (\mathbf{D}^T\mathbf{D})\otimes\mathbf{I}\right)\mathbf{x}\right) \qquad (4.22)
\end{aligned}$$

where $\mathbf{D}$ is defined in (4.20) and '$\otimes$' denotes Kronecker product. Note that (4.22) has the form (4.16) with $N = n^2$ and $\mathbf{L} = \mathbf{I}\otimes(\mathbf{D}^T\mathbf{D}) + (\mathbf{D}^T\mathbf{D})\otimes\mathbf{I}$. Note that this agrees with (4.14) for $\mathbf{L}_{1\mathrm{D}} = \mathbf{D}^T\mathbf{D}$.

Similar to the 1D case, the increment model (4.18) corresponds to the autoregressive model (4.12). In this case, the neighborhood system has the form

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

whereas if i or j is 1 or n, the vertices containing a 0 or $n+1$ are removed from ∂_{ij}; e.g. $\partial_{1j} = \{(2,j),(1,j-1),(1,j+1)\}$ and $\partial_{11} = \{(1,2),(2,1)\}$. Note then that $n_{ij} = |\partial_{ij}|$ is one of 2,3, or 4. Also as in 1D, the resulting matrix $\mathbf{L}$ is the discrete second derivative matrix on the computational grid $\{(i/(n+1), j/(n+1)\}_{i,j=1}^{n}$ with Neumann (zero flux) BCs, $\partial x(s,t)/\partial s|_{s=0,1} = 0$, and $\partial x(s,t)/\partial t|_{t=0,1} = 0$.

The realizations from these GMRFs are also very similar in appearance to those in Figure 4.1, except that they satisfy the Neumann boundary boundary condition.

Independent increments

In certain situations, it is advantageous to allow for the increment variance to be large in some areas than others. For example, if you know that your image $\mathbf{x}$ has a sharp intensity change at pixel i, you should allow for a large increment variance at that location, whereas in other areas of the image you might want to enforce a smaller increment variance.

In the 1D case, this motivates changing (4.18) to

$$\Delta x_i \overset{iid}{\sim} \mathcal{N}(0, (w_i\delta)^{-1}), \quad i = 1,\ldots,n-1, \qquad (4.23)$$

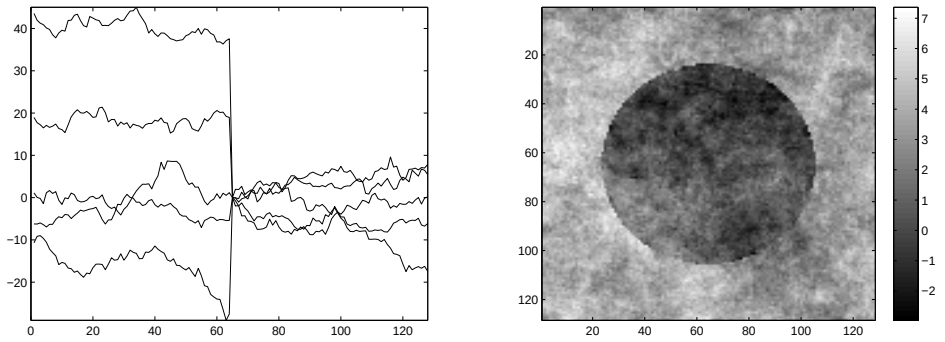

Figure 4.2.

where $\Delta x_i = x_{i+1} - x_i$. Then the density function for $\mathbf{x}$ has the form

$$\begin{aligned} p(\mathbf{x}|\delta) &\propto \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\sum_{i=1}^{n-1} w_i (x_{i+1}-x_i)^2\right) \\ &\propto \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\|\mathbf{\Lambda}^{1/2}\mathbf{D}\mathbf{x}\|\right) \\ &= \delta^{(n-1)/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T(\mathbf{D}^T\mathbf{\Lambda}\mathbf{D})\mathbf{x}\right), \end{aligned} \tag{4.24}$$

where $\mathbf{\Lambda} = \mathrm{diag}(w_1, \ldots, w_{n-1})$ and $\mathbf{D}$ is as in (4.20).

The prior (4.24) corresponds to the auto-regressive model (see [25, Chapter 3])

$$x_i|\mathbf{x}_{\partial_i} \sim \mathcal{N}(\overline{x}_{\partial_i}, (\delta n_i w_i)^{-1}), \tag{4.25}$$

where $\overline{x}_{\partial_i} = \frac{1}{n_i}\sum_{j\in\partial_i} x_j$, where the neighborhood system is the same as for the 1D iid increment case. Comparing with (4.9), we see that (4.25) allows for a spatially varying precision.

On the left in Figure 4.2, we plot five realization from such a GMRF, with $w_{n/2} = 0.05$ and $w_i = 1$ for all $i \neq n/2$. Note that by decreasing the precision value at pixel $n/2$, realizations with a large jump at pixel $n/2$ have relatively high probability.

Note also that (4.24) has the form (4.16) with $N = n$ and $\mathbf{L} = \mathbf{D}^T\mathbf{\Lambda}\mathbf{D}$, which is also a numerical discretization of the diffusion operator

$$\frac{d}{dt}\left(w(t)\frac{d}{dt}\right).$$

Here $0 \leq t \leq 1$, the computational grid is $\{i/(n+1)\}_{i=1}^n$, the derivative is approximated using forward differences, $w_i = w(i/(n+1))$, and Neumann boundary conditions, $x'(0) = x'(1) = 0$ are assumed.

In 2D case, we assume for $i, j = 1, \ldots, n-1$ that

$$\Delta_h x_{ij} \overset{iid}{\sim} \mathcal{N}(0, (w_{ij}^h \delta)^{-1}) \quad \text{and} \quad \Delta_v x_{ij} \overset{iid}{\sim} \mathcal{N}(0, (w_{ij}^v \delta)^{-1}). \tag{4.26}$$

Then the density function for $\mathbf{x}$ has the form

$$\begin{aligned} p(\mathbf{x}|\delta) &\propto \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\left(\sum_{j=1}^{n}\sum_{i=1}^{n-1} w_{ij}^h \Delta_h x_{ij}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n-1} w_{ij}^v \Delta_v x_{ij}^2\right)\right) \\ &= \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\left(\|\mathbf{\Lambda}_h^{1/2}(\mathbf{I}\otimes\mathbf{D})\mathbf{x}\|^2 + \|\mathbf{\Lambda}_v^{1/2}(\mathbf{D}\otimes\mathbf{I})\mathbf{x}\|^2\right)\right) \\ &= \delta^{(n^2-1)/2} \exp\left(-\frac{\delta}{2}\mathbf{x}^T(\mathbf{D}_h^T\mathbf{\Lambda}_h\mathbf{D}_h + \mathbf{D}_v^T\mathbf{\Lambda}_v\mathbf{D}_v)\mathbf{x}\right), \end{aligned} \tag{4.27}$$

where $\mathbf{D}_h = \mathbf{I}\otimes\mathbf{D}$ and $\mathbf{D}_v = \mathbf{D}\otimes\mathbf{I}$ are the discrete horizontal and vertical derivatives, respectively, $\mathbf{\Lambda}_h = \text{diag}(\text{vec}(\{w_{ij}^h\}_{ij=1}^n))$, and $\mathbf{\Lambda}_v = \text{diag}(\text{vec}(\{w_{ij}^v\}_{ij=1}^n))$.

The prior (4.27) corresponds to the the auto-regressive model (see [25, Chapter 3]),

$$x_{ij}|\mathbf{x}_{\partial_{ij}} \sim \mathcal{N}\left(\frac{w_{ij}^h \sum_{(r,s)\in\partial_{ij}^h} x_{rs} + w_{ij}^v \sum_{(r,s)\in\partial_{ij}^v} x_{rs}}{n_{ij}^h w_{ij}^h + n_{ij}^v w_{ij}^v}, \frac{1}{\delta(n_{ij}^h w_{ij}^h + n_{ij}^v w_{ij}^v)}\right),$$

where ∂_{ij}^h and ∂_{ij}^v denote the vertical and horizontal neighbors of pixel (i,j), respectively, and $n_{ij}^h = |\partial_{ij}^h|$ and $n_{ij}^v = |\partial_{ij}^v|$. This expression may seem complicated, but in the case that $w_{ij} = w_{ij}^h = w_{ij}^v$ it takes the form

$$x_{ij}|\mathbf{x}_{\partial_{ij}} \sim \mathcal{N}(\overline{x}_{\partial_{ij}}, (\delta n_{ij} w_{ij})^{-1}), \tag{4.28}$$

where $\overline{x}_{\partial_{ij}} = \frac{1}{n_{ij}}\sum_{(r,s)\in\partial_{ij}} x_{rs}$ and $n_{ij} = |\partial_{ij}|$. Comparing with (4.12), we see clearly that (4.28) allows for a spatially varying precision.

On the right in Figure 4.2, we plot a single realization from such a GMRF. This time we set $w_{ij} = 0.001$ along the boundary of a circle contained in $[0,1]\times[0,1]$ and otherwise let $w_{ij} = 1$. This realization shows that by uniformly decreasing the precision values along the boundary of the circle, we allow for different average values within and without the circle boundary. Otherwise the GMRF behaves similarly to the iid increment case.

Finally, note that (4.27) has the form (4.16) with $N = n^2$ and $\mathbf{L} = \mathbf{D}_h^T\mathbf{\Lambda}_h\mathbf{D}_h + \mathbf{D}_v^T\mathbf{\Lambda}_v\mathbf{D}_v$, which is also a numerical discretization of the diffusion operator

$$\frac{\partial}{\partial s}\left(w_h(s,t)\frac{\partial}{\partial s}\right) + \frac{\partial}{\partial t}\left(w_v(s,t)\frac{\partial}{\partial t}\right).$$

Here $0 \le s,t \le 1$, the computational grid is $\{(i/(n+1), j/(n+1)\}_{i=1}^n$, the derivative is approximated using forward differences, $w_{ij}^h = w_h((i/(n+1), j/(n+1))$, $w_{ij}^v = w_v((i/(n+1), j/(n+1))$, and Neumann boundary conditions are assumed: $\partial x(s,t)/\partial s|_{s=0,1} = 0$, and $\partial x(s,t)/\partial t|_{t=0,1} = 0$.

4.2.2 Maximum a Posteriori Estimation

With the GMRF priors introduced in the previous section, we can now return to the deblurring inverse problems from the previous chapters. In all instances, we use the probability density function (4.1) and either a GMRF prior of the form (4.15) or an IGMRF prior of the form (4.16). Then the *maximum a posteriori* (MAP) estimator is given by

$$\mathbf{x}_{\text{MAP}} = \arg\min_{\mathbf{x}} \left\{ \frac{\lambda}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \frac{\delta}{2}\mathbf{x}^T\mathbf{L}\mathbf{x} \right\}. \tag{4.29}$$

As in the case of Tikhonov regularization, $\mathbf{x}_{\text{MAP}}$ can also be expressed as the solution of the linear system of equations

$$(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{L})\mathbf{x} = \mathbf{A}^T\mathbf{b}, \tag{4.30}$$

where $\alpha = \delta/\lambda$. In this case, we will denote $\mathbf{x}_{\text{MAP}}$ by $\mathbf{x}_\alpha$.

One of the regularization parameter selection methods from Chapter 2 can be used to estimate the regularization parameter α: if $\lambda = 1/\sigma^2$ is known, UPRE or DP can be used, otherwise GCV or the L-curve method must be used. In the examples below, we will use GCV. Recall that the GCV function has the form (see (2.23))

$$G(\alpha) = \frac{n\|\mathbf{A}\mathbf{x}_\alpha - \mathbf{b}\|^2}{[\text{tr}(\mathbf{I} - \mathbf{A}\mathbf{A}_\alpha)]^2},$$

where $\mathbf{A}_\alpha = (\mathbf{A}^T\mathbf{A} + \alpha\mathbf{L})^{-1}\mathbf{A}^T$.

When parameters in the prior are estimated from the data, such as is the case when the regularization parameter α is estimated using the regularization parameter selection method, the approach is called *empirical Bayes* [9].

In the 1D deblurring case, minimizing G is computationally feasible regardless of any special structure in the problem. If we compute α using GCV for the GMRF prior defined by (4.15) with $N = n$ and $\mathbf{L}$ given by the 1D discrete negative-Laplacian with zero BCs (4.11), we obtain the reconstruction plotted on the upper-left in Figure 4.3.

In 2D, special structure in the problem must be exploited in order to make the computations feasible. In what follows, as in Chapter 3, $\mathbf{X}$ and $\mathbf{B}$ denote the $n \times n$ unknown image and data arrays, respectively, while $\mathbf{x}$ and $\mathbf{b}$ denote the corresponding $n^2 \times 1$ column-stacked vectors.

If periodic boundary conditions are assumed in the 2D array $\mathbf{X}$, the matrices $\mathbf{A}$ and $\mathbf{L}$ will both have block circulant structure and hence will be diagonalizable by the 2D-DFT. Thus $\mathbf{L} = \mathbf{L}_{2\text{D}}$ from Section 4.2.1. Recall that the eigenvalues of $\mathbf{A}$ are given by $n^2\text{DFT}(\mathbf{a}_s)$, where $\mathbf{a}_s$ is defined in (3.23). The eigenvalues of $\mathbf{L}$ can be similarly defined. In particular, if

$$\mathbf{l} = \begin{bmatrix} l_{1-n/2,1-n/2} & \cdots & l_{1-n/2,0} & \cdots & l_{1-n/2,n/2-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ l_{0,1-n/2} & \cdots & l_{0,0} & \cdots & l_{0,n/2-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ l_{n/2-1,1-n/2} & \cdots & l_{n/2-1,0} & \cdots & l_{n/2-1,n/2-1} \end{bmatrix},$$

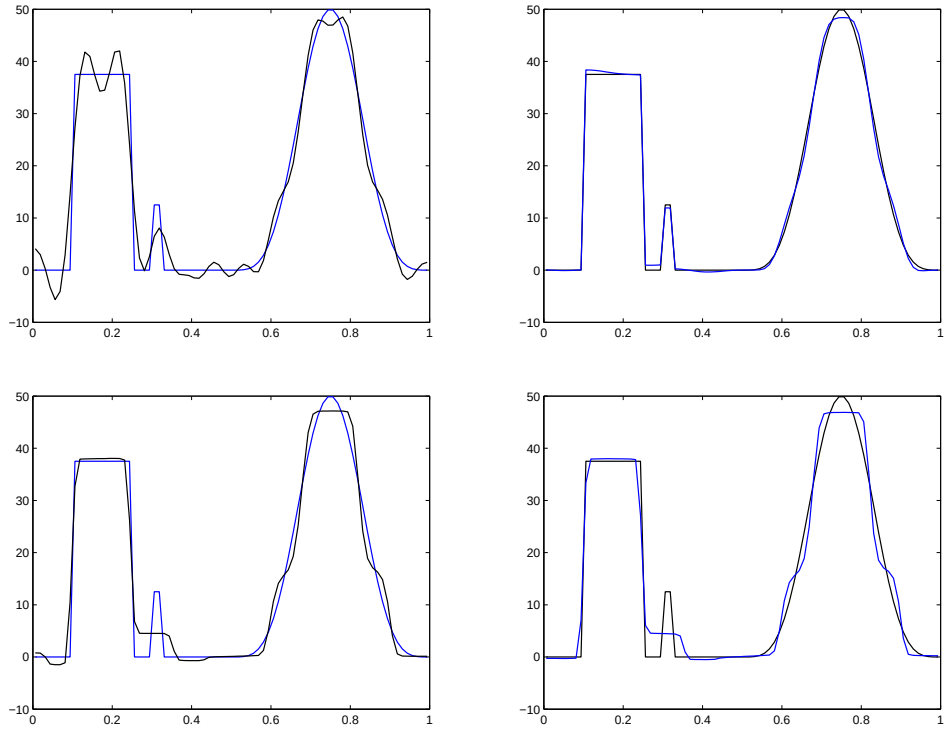

Figure 4.3. *One-dimensional test case. In the upper-left is the reconstruction obtained assuming a GMRF prior with* $\mathbf{L}$ *defined by (4.11). In the upper-right is the reconstruction obtained using* $\mathbf{L} = \mathbf{D}^T\mathbf{\Lambda}\mathbf{D}$*, where* $\mathbf{\Lambda}$ *is defined by (4.33) with* $\bar{\mathbf{x}} = \mathbf{x}_{\text{true}}$*. In the lower figures are the reconstructions obtained using Algorithm* **??** *after five (on the left) and ten (on the right) iterations.*

with $l_{0,0} = 4$, $l_{-1,0} = l_{1,0} = l_{0,1} = l_{0,-1} = -1$, and $l_{ij} = 0$ otherwise, then the eigenvalues of $\mathbf{L}$ have the form $\hat{\mathbf{l}}_s = n^2\text{DFT}(\mathbf{l}_s)$, where $\mathbf{l}_s$ is the circulant shift of $\mathbf{l}$ as defined in (3.23), and

$$\mathbf{L}\mathbf{x} = \text{vec}(\text{IDFT}(\hat{\mathbf{l}}_s \odot \text{DFT}(\mathbf{X}))).$$

Just as in the case of standard Tikhonov regularization with periodic boundary conditions (see (3.24)), the regularized solution can be expressed in the array form

$$\mathbf{X}_\alpha = \text{IDFT}\left(\left(\frac{\text{conj}(\hat{\mathbf{a}}_s)}{|\hat{\mathbf{a}}_s|^2 + \alpha\hat{\mathbf{l}}_s}\right) \odot \text{DFT}(\mathbf{B})\right). \tag{4.31}$$

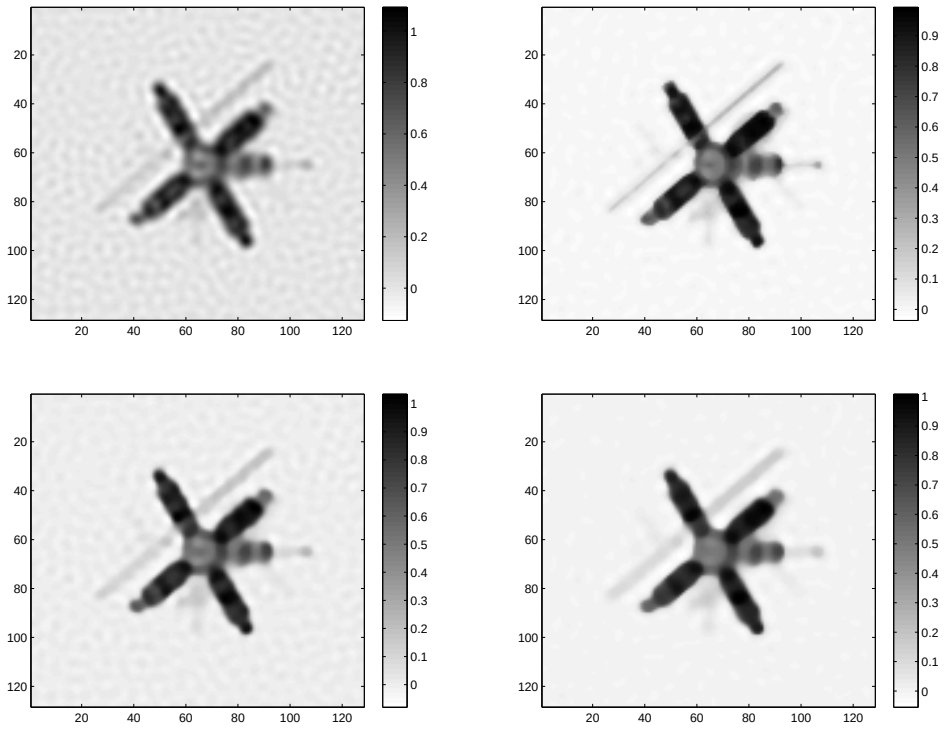

Figure 4.4. *Two-dimensional test case. In the upper-left is the blurred, noisy data. In the upper-right is the reconstruction using the iid increment IGMRF with periodic boundary conditions, described in Section 4.2.1. In the lower figures are the reconstructions obtained using Algorithm* **??** *after five and ten iterations.*

Moreover, the GCV function can be written

$$G(\alpha) = \left(\alpha^2 \sum_{i,j=1}^{n} \frac{|\hat{\mathbf{l}}_s|_{ij}^2 \hat{\mathbf{B}}_{ij}^2}{(|\hat{\mathbf{a}}_s|_{ij}^2 + \alpha[\hat{\mathbf{l}}_s]_{ij})^2} \right) \Bigg/ \left(n^2 - \sum_{i,j=1}^{n} \frac{|\hat{\mathbf{a}}_s|_{ij}^2}{|\hat{\mathbf{a}}_s|_{ij}^2 + \alpha[\hat{\mathbf{l}}_s]_{ij}} \right)^2 . \quad (4.32)$$

where $\hat{\mathbf{B}} = n^2 \mathrm{DFT}(\mathbf{B})$, and if $\mathbf{v} \in \mathbb{C}^n$, $|\mathbf{v}|$ denotes the vector whose components are the magnitudes of the elements of $\mathbf{v}$. Equation (4.32) can be derived from (2.24) and should be compared with (3.25), where instead of $\hat{\mathbf{l}}_s$ the array of ones $\mathbf{1}$ is used instead. The blurred noisy data considered in Chapter 3, together with the reconstruction obtained using (4.31) with α the nonnegative minimizer of (4.32), are presented in Figure 4.4

Edge-Preserving GMRF Priors

Because typical images contain sharp intensity changes, it is desirable to develop GMRF priors that allow for the formation of edges in MAP estimates. However, this requires knowledge of the location of edges in the unknown, which we do not in general know. In this sub-section, we propose a method for learning the location of edges via the repeated computation of MAP estimates.

To motivate, we note that the edges in the unknown $\mathbf{x}$ correspond to pixels i at which the magnitude of the discrete gradient – $\sqrt{[\mathbf{D}\mathbf{x}]_i^2}$ in 1D and $\sqrt{[\mathbf{D}_h\mathbf{x}]_i^2 + [\mathbf{D}_v\mathbf{x}]_i^2}$ in 2D – is large and where we wish to allow for larger variance in the increments.

Indeed, we can scale the increment precision value directly by the magnitude of the gradient. In 1D, this is encoded into the diagonal matrix $\mathbf{\Lambda}$ in (4.24) by choosing

$$\mathbf{\Lambda}(\overline{\mathbf{x}}) = \text{diag}\left(\frac{\mathbf{1}}{\sqrt{(\mathbf{D}\overline{\mathbf{x}})^2 + \beta\mathbf{1}}}\right), \tag{4.33}$$

where $\overline{\mathbf{x}}$ is an estimator for the random vector $\mathbf{x}$ and $\beta > 0$ (we use $\beta = 0.001$ in the examples below); whereas in 2D, $\mathbf{\Lambda}_h$ and $\mathbf{\Lambda}_v$ appear in (4.27) and we choose

$$\mathbf{\Lambda}_h(\overline{\mathbf{x}}) = \text{diag}\left(\frac{\mathbf{1}}{\sqrt{(\mathbf{D}_h\overline{\mathbf{x}})^2 + \beta\mathbf{1}}}\right), \quad \mathbf{\Lambda}_v(\overline{\mathbf{x}}) = \text{diag}\left(\frac{\mathbf{1}}{\sqrt{(\mathbf{D}_v\overline{\mathbf{x}})^2 + \beta\mathbf{1}}}\right). \tag{4.34}$$

again where $\overline{\mathbf{x}}$ is an estimator for the random vector $\mathbf{x}$.

As a proof of concept, we take $\overline{\mathbf{x}}$ to be the image used to generate the data, in both the 1D and 2D test cases. Choosing α using GCV, the resulting reconstructions are given in the upper-left in Figures 4.3 and 4.4. Of course, this is an unrealistic approach since we are assuming that $\mathbf{x}$ is a random vector and will never have the 'true image' in hand. However, the result motivates to following sequential MAP method for edge preserving image reconstruction.

Algorithm 4.1. Edge-Preserving Reconstruction Using GMRF Priors.

0. Set $\mathbf{\Lambda} = \mathbf{I}$ in 1D and $\mathbf{\Lambda}_h = \mathbf{\Lambda}_v = \mathbf{I}$ in 2D.
1. In 1D, define $\mathbf{L} = \mathbf{D}^T\mathbf{\Lambda}\mathbf{D}$. In 2D, define $\mathbf{L} = \mathbf{D}_h^T\mathbf{\Lambda}_h\mathbf{D}_h + \mathbf{D}_v^T\mathbf{\Lambda}_v\mathbf{D}_v$.
2. Compute the solution $\mathbf{x}_\alpha$ of (4.30) with α obtained using GCV.
3. In 1D, set $\mathbf{\Lambda} = \mathbf{\Lambda}(\mathbf{x}_\alpha)$ defined by (4.33). In 2D, set $\mathbf{\Lambda}_h = \mathbf{\Lambda}_h(\mathbf{x}_\alpha)$ and $\mathbf{\Lambda}_v = \mathbf{\Lambda}_v(\mathbf{x}_\alpha)$ defined by (4.34). Return to Step 1.

Note in the first iteration of Algorithm 4.2.2, $\mathbf{\Lambda} = \mathbf{I}$ which corresponds to the iid increment test case, and the corresponding reconstruction is given on the upper-left in Figure 4.3 in 1D and in Figure 4.4 in 2D, whereas in the lower-left and lower-right in Figures 4.3 and 4.4 after a specified number of iterations of Algorithm 4.2.2. Note that as the iterations proceed, the reconstructions become

increasingly piece-wise constant, which results in a cartoon-like appearance in the 2D reconstructions in Figure 4.4.

In 2D special structure in the problems must be exploited in order to make the computations feasible. First, recall that $\mathbf{D}_h = \mathbf{I} \otimes \mathbf{D}$ and $\mathbf{D}_v = \mathbf{D} \otimes \mathbf{I}$. We assume periodic boundary conditions on $\mathbf{X}$ so that

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & -1 & 1 \\ 1 & 0 & \cdots & 0 & -1 \end{bmatrix}_{n \times n} . \tag{4.35}$$

Nonetheless, the presence of $\mathbf{\Lambda}$ in (4.24) leads to a matrix $\mathbf{L}$ that no longer has periodic structure. Thus as in the case of deblurring with Dirichlet BCs in Section **??**, we must use Algoirthm 3.2.3 (PCG) to compute the solution $\mathbf{x}_\alpha$ of (4.30). Indeed, our implementation parallels that of PCG in Section **??**, except that $\mathbf{B} = \mathbf{A}^T\mathbf{A} + \alpha\mathbf{L}$, where $\mathbf{L} = \mathbf{D}_h^T\mathbf{\Lambda}\mathbf{D}_h + \mathbf{D}_v^T\mathbf{\Lambda}\mathbf{D}_v$, and the preconditioner is defined by

$$\mathbf{M}^{-1}\mathbf{r} = \text{vec}\left(\text{IDFT}\left(\frac{\mathbf{1}}{|\hat{\mathbf{a}}_s|^2 + \alpha\hat{\mathbf{l}}_s} \odot \text{DFT}(\mathbf{R})\right)\right),$$

where $\mathbf{r} = \text{vec}(\mathbf{R})$, and $\hat{\mathbf{l}}_s$ is the $n \times n$ eigenvalue array of the 2D discrete negative-Laplacian with periodic BCs as defined above. Moreover, GCV must be implemented in the same fashion as in Section **??**.

Finally, we note that a similar edge-preserving quadratic regularization technique was presented in [**?**], but there the regularization parameter was only estimated in the first iteration, which resulted in a more complicated update scheme for $\mathbf{\Lambda}$.

Exercises

4.1. a. Let $h = 1/n$ and $s_i = (i - 1/2)h$ for $i = 1, \ldots, n$. Expand both $x(s_i + h)$ and $x(s_i - h)$ in a four term Taylor series about s_i and show that

$$x''(s_i) = \frac{x(s_{i-1}) - 2x(s_i) + x(s_{i+1})}{h^2} + E_h, \quad i = 1, \ldots, n,$$

where E_h has order h^2. Assume that x is four times continuously differentiable.

b. Let $x_i = x(s_i)$ and define $\mathbf{x} = (x_1, \ldots, x_n)$. Assuming a zero boundary condition, i.e. $x_0 = x_{n+1} = 0$, show that the expression in part (a) yields $\mathbf{L}_{1\text{D}}\mathbf{x}$, where $\mathbf{L}_{1\text{D}}$ is defined in (4.11).

4.2. Show that (4.13) can be equivalently expressed (4.14).

4.3. *Sampling from Gaussian probability densities.*

a. Let $\mathbf{B}$ be an $m \times n$ matrix, $\mathbf{v} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, and $\mathbf{w} = \mathbf{B}\mathbf{v}$, then $\mathbf{w} \sim \mathcal{N}(\mathbf{B}\boldsymbol{\mu}, \mathbf{B}\mathbf{C}\mathbf{B}^T)$. Use this to show that if $\mathbf{R}$ is a square root matrix for $\mathbf{C}$, i.e. $\mathbf{C} = \mathbf{R}^T\mathbf{R}$, and if $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ then $\mathbf{w} = \boldsymbol{\mu} + \mathbf{R}^T\mathbf{v}$ has distribution $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$.

b. Use MATLAB and part (b) to compute and plot 1000 samples from the random vector $\mathbf{w} \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\right)$.

c. If $\mathbf{v} = (v_1, \ldots, v_k)$ is a random vector such that $v_i \sim \mathcal{N}(0, 1)$ for $i = 1, \ldots, k$, then $Q = \sum_{i=1}^{k} w_i^2$ is a $\chi^2(k)$ random variable. Verify that, approximatively, a correct amount of the sampled points from part (b) are located inside the confidence regions given by the 95% and 99% limits of the $\chi^2(2)$ distribution, computed using the `chi2inv` function. Note that you need to 'normalize' the samples, i.e. look at $\mathbf{C}^{-1/2}(\boldsymbol{\mu} - \mathbf{w}_i)$.

4.4. *GMRF with periodic boundary conditions.*

a. Modify `GMRFDirichlet.m` so that it computes samples from (4.16) where $\mathbf{L}_{1\mathrm{D}}$ is given by (4.17) in one dimension and (4.14) in two dimensions. Note that these precision matrices have a zero eigenvalue, so you cannot use the Cholesky factorization. Compute the square root of $\mathbf{L}$ and $\mathbf{L}^\dagger$ using an eigenvalue decomposition instead.

b. Note that in the periodic boundary conditions case, multiplication by $\mathbf{L}_{2\mathrm{D}}$ is equivalent to discrete convolution (3.8) with the $n \times n$ kernel $\boldsymbol{\ell}$ defined by

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

Use this and the results from Chapter 3 to compute samples from (4.16) using the 2D-DFT.

Bibliography

[1] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter Estimation and Inverse Problems*, Elsevier, 2005.

[2] G. Backus and F. Gilbert, *The resolving power of gross earth data*, Geophysical Journal of the Royal Astronomical Society, **266** (1968), pp. 169-205.

[3] J. M. Bardsley, *An MCMC Method for Estimation and Uncertainty Quantification in Linear Inverse Problems*, submitted, Math Sciences, University of Montana, Tech. Report #28, 2010.

[4] M. Bertero and P. Boccacci, *Inverse Problems in Imaging*, Institute of Physics, London, 1998.

[5] M. Bertero and P. Boccacci, *A simple method for the reduction of boundary effects in the Richardson-Lucy approach to image deconvolution*, Astronomy and Astrophysics, **437** (2005), pp. 369-374.

[6] J. Besag, *Spatial Interaction and the Statistical Analysis of Lattice Systems*, Journal of the Royal Statistical Society, Series B, **36(2)** (1974), pp. 192-236.

[7] C. L. Epstein, *An Introduction to the Mathematics of Medical Imagine*, SIAM 2008.

[8] H. Engl, M. Hanke, A. Neubauer, *Regularization for Inverse Problems*, Springer 1996.

[9] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis, Second Edition*, Chapman & Hall/CRC, Texts in Statistical Science, 2004.

[10] G. Golub and C. Van Loan, *Matrix Computations, 3rd Edition*, Johns Hopkins University Press, 1996.

[11] J. Hadamard, *Sur les problmes aux drives partielles et leur signification physique*, Princeton University Bulletin, (1902), pp. 4952

[12] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Review, **34** (1992), pp. 561-580.

[13] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM, Philadelphia, 1996.

[14] P. C. Hansen, *Discrete Inverse Problems: Insight and Algorithms*, SIAM, Philadelphia, 2010.

[15] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, 2006.

[16] Dave Higdon, *A primer on space-time modelling from a Bayesian perspective*, Los Alamos Nation Laboratory, Statistical Sciences Group, Technical Report, LA-UR-05-3097.

[17] J. Kaipio and E. Somersalo, *Statistical and Computational Methods for Inverse Problems*, Springer, 2005.

[18] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, Brooks/Cole, 2002.

[19] C. L. Mallows, *Some comments on C_P*, Technometrics, **15(4)** (1973), pp. 661-675.

[20] D. W. Marquardt, *Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation*, Technometrics, **12** (1970), pp. 591-612.

[21] R. H. Meyers, *Classical and Modern Regression with Applications*, 2nd ed., PWS Kent, Boston, 1990.

[22] J. Nagy and K. Palmer, *Steepst Descent, CG and Iterative Regularization of Ill-Posed Problems*, BIT, **43** (2003), pp. 1003-1017.

[23] J. Kamm and J. G. Nagy, *Kronecker Product and SVD Approximations in Image Restoration*, Linear Algebra and its Applications, **284** (1998), pg. 177–192.

[24] F. O'Sullivan, *A statistical perspective on ill-posed inverse problems*, Statistical Science, **1(4)** (1986), pp. 104-127.

[25] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, Chapman and Hall/CRC, 2005.

[26] L. Tenorio, *Statistical Regularization of Inverse Problems*, SIAM Review, **43** (2001), pp. 347-366.

[27] C. Van Loan, *The ubiquitous Kronecker product*, Journal of Computational and Applied Mathematics, **123** (2000), pp. 85-100.

[28] C. R. Vogel, *Computational Methods for Inverse Problems*, SIAM, Philadelphia, 2002.

[29] C. R. Vogel, *Non-convergence of the L-curve regularization parameter selection method*, Inverse Problems, **12** (1996), pp. 535-547.

[30] G. Wahba, *Practical approximate solutions to linear operator equations when the data are noisy*, SIAM Journal on Numerical Analysis, **14** (1977), pp. 651-667.