Chapter 1

Characteristics of Inverse Problems

In this chapter, we describe various characteristics of inverse problems, which we illustrate on several numerical examples. We also show that for inverse problems, the least squares estimator can be expected to have high variance, i.e., it is unstable with respect to measurement error.

1.1 Preliminaries

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

\[ b = Ax + \epsilon, \]  

(1.1)

where \( b \) is the \( m \times 1 \) model output, or observation, vector; \( x \) is the \( n \times 1 \) vector of unknown model parameters; \( A \) is the \( m \times n \) forward model matrix, which sometimes requires a separate set of observations; and \( \epsilon \) is an \( m \times 1 \) independent and identically distributed (iid) Gaussian random vector with variance \( \sigma^2 \) across all entries, which we denote \( \epsilon \sim N(0, \sigma^2 I) \).

Such models arise in introductory courses on linear algebra or statistics, when students study regression problems, which typically involve a statistical model of the form (1.1). A linear regression example follows.

**Example 1.1** Suppose that as a scientist you want to find a formula to estimate the weight \( b \) in kilograms of a lion by using its length \( \ell \) in meters. You have five samples: \((b, \ell) = (420, 2.4), (350, 2.0), (310, 2.1), (280, 1.8), \) and \((75, 1.3)\). A plot of this data in Figure 1.1 shows that it exhibits a linear trend, hence we assume a linear model of the form

\[ b_i = x_1 + x_2 \ell_i + \epsilon_i, \]  

(1.2)
where \( \epsilon_i \sim \mathcal{N}(0, \sigma^2) \). Gathering the data in matrix form \( \mathbf{b} = \mathbf{A} \mathbf{x} \), (1.2) yields
\[
\begin{bmatrix}
420 \\
350 \\
310 \\
280 \\
75
\end{bmatrix} =
\begin{bmatrix}
1 & 2.4 \\
1 & 2.0 \\
1 & 2.1 \\
1 & 1.8 \\
1 & 1.3
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}.
\] (1.3)

The system (1.3) does not have a solution since no line interpolates all of the data points, but we can compute the least squares solution, which is obtained by solving the normal equations \( \mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b} \). From this we obtain the best-fit line \( \mathbf{b} = -303.08 + 307.34 \ell \), also plotted with the data in Figure 1.1. We will discuss least squares in more detail in a moment.

Note the process that was involved in determining the model (1.2): first the data was plotted, a linear trend was noted, and then a linear model was chosen. Thus the data dictated the model. In this book, we focus on a different class of problems – called inverse problems – which also yield statistical models of the form (1.1), but are different from regression problems in that the measurements \( \mathbf{b} \) are not used to construction the model \( \mathbf{A} \mathbf{x} \). The model is instead obtained from the numerical discretization of a continuously defined physical model \( \mathbf{b} = \mathbf{A} \mathbf{x} \), where \( \mathbf{b} \) and \( \mathbf{x} \) are functions and \( \mathbf{A} \) is an operator. As a result the dimension \( n \) of \( \mathbf{x} \) is typically chosen to be relatively large so that the numerical discretization is accurate. The downside of this is that the model tends to be ‘over parameterized’, which leads to least squares solutions with high variance or, in other words, that are unstable with respect to the noise. This is a characteristic of inverse problems that will be discussed in detail it below. The next two examples are typical inverse problems.
1.1. Preliminaries

**Example 1.2** In many important applications, the underlying continuous physical model takes the form of a convolution:

\[ b(s) = \int_{-\infty}^{\infty} a(s - s')x(s')ds', \quad -\infty \leq s \leq \infty. \]

The problem of estimating \( x \) from \( b \) in this setting is a classical inverse problem [16].

In practice, the domain for \( b \) and \( a \) is defined by the measurement device. We'll assume that \( b \) is measured on the interval \([0,1]\), and that \( a \) is measured on the interval \((-1,1)\), a reasonable assumption. Given these assumptions, and assuming that \( a(s) = 0 \) for \(|s| \geq 1\), the convolution model above reduces to

\[ b(s) = \int_{-1}^{1} a(s - s')x(s')ds', \quad 0 \leq s \leq 1. \]  \hspace{1cm} (1.4)

We discretize (1.4) using midpoint quadrature. Let \( h = 1/n \) and \( s'_j = (j - 1/2)h \), for \( j = -n + 1, \ldots, 2n \). Then

\[ \int_{-1}^{1} f(s')ds' = h \sum_{j=-n+1}^{2n} f(s'_j) + E_n, \]  \hspace{1cm} (1.5)

where \( E_n \) denotes quadrature error for the midpoint rule, which is discussed in more detail in Exercise 2. Assuming the same grid for the \( s \) variable within \([0,1]\), i.e., \( s_j = s'_j \) for \( j = 1, \ldots, n \), we define \( b_i \equiv b(s_i) \), \( x_j \equiv x(s'_j) \), \( a_{i-j} \equiv a(s_i - s'_j) = a((i-j)h) \), and apply (1.5) to (1.4). Ignoring the quadrature error \( E_n \), and using the fact that \( a_{i-j} = 0 \) for \(|i-j| \geq n\), we obtain

\[ b_i = h \sum_{j=-n+1}^{i+n-1} a_{i-j}x_j = h \sum_{j=-n+1}^{i+n-1} a_jx_{i-j}, \quad i = 1, \ldots, n. \]  \hspace{1cm} (1.6)

In matrix-vector notation, (1.6) has the form

\[
\begin{bmatrix}
 b_1 \\
 b_2 \\
 \vdots \\
 b_n
\end{bmatrix}
= h
\begin{bmatrix}
 a_{n-1} & \cdots & a_0 & \cdots & a_{-n+1} \\
 & \ddots & \ddots & \ddots & \cdots \\
 & & a_{n-1} & \cdots & a_0 & \cdots & a_{-n+1}
\end{bmatrix}
\begin{bmatrix}
 x_{-n+2} \\
 \vdots \\
 x_1 \\
 \vdots \\
 x_n \\
 \vdots \\
 x_{2n-1}
\end{bmatrix}.
\]  \hspace{1cm} (1.7)

Note that this system has \( n \) equations and \( 3n - 2 \) unknowns, so that it is underdetermined.

It is evident from (1.7) that the values of \( b_i \) for \( i \) near 1 and \( n \) will depend upon the elements \( (x_{-n+2}, \ldots, x_0) \) and \( (x_{n+1}, \ldots, x_{2n-1}) \), respectively, all of which
Chapter 1. Characteristics of Inverse Problems

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

Correspond to values of $x(s)$ for $s \notin [0,1]$. In order to reduce (1.7) to an $n \times n$ system of equations, assumptions are often made – called boundary conditions in the imaging literature [29] – about the values of $x(s)$ for $s \notin [0,1]$. We begin in this chapter by assuming $x(s) = 0$ for $s \notin [0,1]$, which is known as the zero boundary condition. Then $(x_{n-1}, \ldots, x_0) = (x_{n+1}, \ldots, x_{2n-1}) = 0$, so that (1.6) can be written

$$b_i = h \sum_{j=1}^{n} a_{i-j} x_j, \quad i = 1, \ldots, n. \quad (1.8)$$

This allows us to express (1.7) as an $n \times n$ linear system $b = Ax$, where $A$ is a Toeplitz (constant diagonals) matrix [49]. We leave the derivation of (1.8) and $A$ to the reader in Exercise 3.

A common choice of kernel is a Gaussian probability density function

$$a(s) = \frac{1}{\sqrt{2\pi\gamma^2}} \exp \left( -\frac{s^2}{2\gamma^2} \right), \quad \gamma > 0, \quad (1.9)$$

which yields the coefficient matrix

$$[A]_{ij} = a((i-j)h) = \frac{h}{\sqrt{2\pi\gamma^2}} \exp \left( -\frac{(i-j)^2h^2}{2\gamma^2} \right), \quad 1 \leq i, j \leq n. \quad (1.10)$$

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

**Example 1.3** The problem of estimating a function from its indefinite integral is also an inverse problem [16]. In this case, the continuous model $b = Ax$ is given by

$$b(s) = \int_{-1}^{s} x(s') \, ds', \quad -1 \leq s \leq 1. \quad (1.11)$$
1.1. Preliminaries

The Second Fundamental Theorem of Calculus tells us that \( x \) can be obtained from \( b \), both defined on \([-1, 1]\), via \( x(s) = b'(s) \).

Now, let’s tie this to a specific application. Note that in the previous example, the kernel \( a \) was assumed to be known. In a typical image deblurring problem, 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 system. If a convolution model is assumed, and the step function has the form

\[
\chi(s) = \begin{cases} 
1 & s > 0, \\
0 & s \leq 0, 
\end{cases}
\]

then the output has the form

\[
b(s) = \int_{-\infty}^{\infty} a(s - s')\chi(s') \, ds'
= \int_{-\infty}^{\infty} \chi(s - s')a(s') \, ds'
= \int_{-\infty}^{\infty} a(s') \, ds'.
\]

Assuming \( a(s') = 0 \) for \( |s'| \geq 1 \), as we did in the previous example, and setting \( x(s') = a(s') \), we obtain (1.11).

For computations, we again use midpoint quadrature, but this time, motivated by the previous example, we seek estimates of the kernel \( x(s') \) at the grid points \( s'_j = jh \), where \( h = 1/n \) and \( j = -n + 1, \ldots, n - 1 \). Note that \( s'_j \) is the midpoint of the interval \([s_i, s_{i+1}]\), where \( s_i = (i - \frac{1}{2})h \). Since \( x(-1) = 0 \), we assume \( b(s_{-n+1}) = 0 \) and approximate (1.11) using midpoint quadrature as follows:

\[
b(s_i) = \int_{s_{i-1}}^{s_i} x(s') \, ds'
= \frac{1}{h} \sum_{j=-n+1}^{-1} x(s'_j), \quad i = -n + 2, \ldots, n.
\]  

(1.12)

Now, we assume that we have measurements \( b_i = b(s_i) \) for \( i = -n + 2, \ldots, n \), and that we want to estimate the kernel \( x_j = x(s'_j) \) for \( j = -n + 1, \ldots, n - 1 \). Letting \( b = [b_{-n+2}, \ldots, b_n]^T \) and \( x = [x_{-n+1}, \ldots, x_{n-1}]^T \), we obtain \( b = Ax \), with

\[
A = h \begin{bmatrix}
1 & 0 & \cdots & 0 \\
1 & 1 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
1 & 1 & \cdots & 1
\end{bmatrix}_{(2n-1) \times (2n-1)}.
\]  

(1.13)
We will leave it as an exercise to show that $A^{-1}$ is given by the backward difference matrix
\[
A^{-1} = h^{-1} \begin{bmatrix}
1 & -1 & 1 & \cdots & 1 \\
-1 & 1 & -1 & \cdots & 1 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
-1 & 1 & \cdots & \cdots & 1
\end{bmatrix}_{(2n-1) \times (2n-1)}.
\] (1.14)

Recall that $b(s_{n+1}) = 0$.

As a numerical test problem, we synthetically generate a $n \times 1$ kernel $x$ with $n = 159$, and then generate synthetic $m \times 1$ data $b$, with $m = 159$ and $\sigma^2$ chosen so that the SNR is 50, with the $m \times n$ matrix $A$ defined by (1.13). The result is plotted in Figure 1.2.

In the previous two examples, the vector of unknowns $x \in \mathbb{R}^n$ is determined by the numerical discretization of an integral equation. The value of $n$ is chosen in order that the discretizations are accurate, and hence is typically large relative to the data. The result is that the model $n$ must be relatively large, making the dimension of the unknown.

### 1.2 The Least Squares Estimator

We begin with some basic properties of Gaussian random variables/vectors. First, for a scalar Gaussian random variable $v \sim \mathcal{N}(\mu, \sigma^2)$, the probability density function for $v$ is given by
\[
p(v|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(v-\mu)^2\right).
\]
The mean $\mu$ of $v$ is given by $\mu = E(v) = \int_{\mathbb{R}} v p(v|\mu, \sigma^2) \, dx$, and the variance $\sigma^2$ of $v$, is given by $\sigma^2 = E((v-\mu)^2) = \int_{\mathbb{R}} (v-\mu)^2 p(v|\mu, \sigma^2) \, dx$. From these definitions, one can prove the following statement: let $c, d \in \mathbb{R}$, then
\[
v \sim \mathcal{N}(\mu, \sigma^2) \text{ and } w = c + dv \Rightarrow w \sim \mathcal{N}(c + d\mu, d^2\sigma^2).
\]

Next, an $n$-dimensional Gaussian random vector $v = [v_1, \ldots, v_n]^T \sim \mathcal{N}(\mu, \Sigma)$ has probability density function given by
\[
p(v|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left(-\frac{1}{2}(v-\mu)^T \Sigma^{-1}(v-\mu)\right).
\]
The mean $\mu = [\mu_1, \ldots, \mu_n]^T$ of $v$, is defined by $\mu = E(v) = [E(v_1), \ldots, E(v_n)]^T$, and the covariance matrix $\Sigma$ of $v$, is defined by $\Sigma = E((v-\mu)(v-\mu)^T)$; that is, $[\Sigma]_{ij} = E((v_i - \mu_i)(v_j - \mu_j))$ for $1 \leq i, j \leq n$. Using the linearity of the expected value function, one can prove the following statement (see Exercise 1.6): let $c \in \mathbb{R}^m$ and $B \in \mathbb{R}^{m \times n}$ with rank $m$, then
\[
v \sim \mathcal{N}(\mu, \Sigma) \text{ and } w = c + Bv \Rightarrow w \sim \mathcal{N}(c + B\mu, BB^T).
\] (1.15)
1.2. The Least Squares Estimator

A common approach for estimating $x$ given $b$ is to maximize the likelihood function, which is defined as

$$L(x|\sigma^2) = p(b|x,\sigma^2),$$

with respect to $x$. Equivalently, we can minimize the negative log-likelihood

$$-\ln L(x|\sigma^2),$$

whose minimizer $x_{LS}$ is the least squares solution, i.e.,

$$x_{LS} = \arg \min_x \frac{1}{2\sigma^2} \|Ax - b\|^2,$$

which, when the rank of $A$ is $n$, can be expressed as (see Exercise 1.8)

$$x_{LS} = (A^T A)^{-1} A^T b,$$

or, if $A$ is invertible, $x_{LS} = A^{-1} b$.

The reader with some experience studying inverse problems will not be surprised that the least squares solutions for Examples 1.2 and 1.3 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. Moreover, in both test cases, the magnitude of the measurement error perturbation, $\|\epsilon\| = \|b - Ax\|$, is amplified in the least squares solution. To see this, note that in the deblurring case $\|Ax - b\|/\|Ax\| = 0.024$ and $\|x - x_{LS}\|/\|x\| = 5.25 \times 10^{-3}$, while for the kernel reconstruction case $\|Ax - b\|/\|Ax\| = 0.022$ and $\|x - x_{LS}\|/\|x\| = 1.47$.

Note that in both cases,

$$\|x - x_{LS}\|/\|x\| \gg \|Ax - b\|/\|Ax\|.$$

![Figure 1.3. The least squares solution $x_{LS}$ for the one-dimensional deblurring example (on the left) and the kernel reconstruction example (on the right).](image)
which states that, relatively speaking, the magnitude of the perturbation in the measurements is significantly smaller than the magnitude of the perturbation in the corresponding least squares solution. This phenomena and the inequality (1.19) represent a fundamental characteristic of ill-posed inverse problems. However, note that both \( b \) and \( x_{LS} \) are random vectors, and hence, so are the expressions on either side of the inequality in (1.19). We will derive, via the use of expectations, a non-stochastic inequality corresponding to (1.19) below.

**1.2.1 The Singular Value Decomposition of \( A \) and \( x_{LS} \)**

In order to understand more deeply why the least squares solution is typically poor for inverse problems, we turn to the singular value decomposition (SVD) of a rank \( r \) matrix \( A \):

\[
A = U \Sigma V^T,
\]

where \( U \) is an \( m \times m \) orthogonal matrix with columns given by the orthonormal eigenvectors of \( AA^T \); \( V \) is an \( n \times n \) orthogonal matrix with columns given by the orthonormal eigenvectors of \( A^T A \); and

\[
\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) \in \mathbb{R}^{m \times n},
\]

where \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \) are known as the singular values of \( A \). The columns of \( U = [u_1, \ldots, u_m] \) and \( V = [v_1, \ldots, v_n] \) are called the left and right singular vectors of \( A \), respectively. It can be shown that (1.20) implies (see Exercise 1.5)

\[
A = \sum_{i=1}^{r} u_i \sigma_i v_i^T,
\]

which is known as the outer product form of the SVD. For more information on the SVD, see [22] and Exercise 5.

Next, we define the pseudo-inverse of \( A \), which can be defined in terms of the SVD by

\[
A^\dagger = V \Sigma^\dagger U^T,
\]

where

\[
\Sigma^\dagger = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) \in \mathbb{R}^{n \times m}.
\]

The outer product form for (1.22) is therefore given by

\[
A^\dagger = \sum_{i=1}^{r} v_i \sigma_i^{-1} u_i^T.
\]

And finally, the least squares solution can be expressed in terms of the pseudo-inverse and its SVD as \( x = A^\dagger b \) [22], so that

\[
x_{LS} = \sum_{i=1}^{r} \left( u_i^T b / \sigma_i \right) v_i
\]

\[
= \sum_{i=1}^{r} (v_i^T x) v_i + \sum_{i=1}^{r} \left( u_i^T \epsilon / \sigma_i \right) v_i.
\]
1.2. The Least Squares Estimator

On the right in (1.24), the first sum is the projection of the true solution $x$ onto the span of $\{v_i\}_{i=1}^{\infty}$, and it is what we are after. The second sum, on the other hand, represents the corruption of $x_{\text{LS}}$ that is due to the measurement error $\epsilon$.

For discretized integral equations, such as we have considered in Examples 1.2 and 1.3, the singular values and singular vectors have specific properties that result in the second term on the right-hand side in (1.24) dominating the first term. To illustrate, in Figures 1.4 and 1.5, we plot the singular values and several singular vectors of $A$ for the deblurring and kernel reconstruction examples, and note that as $i$ increases, $\sigma_i$ gets closer to zero and the $v_i$’s become more oscillatory (this is shown to hold more generally for inverse problems in [28]). As a result, the second term in (1.24) dominates the first and has significant high frequency content, which explains why least squares solutions in (1.3) are corrupted by high frequency content. Note, moreover, that the situation is not as bad in the kernel reconstruction case, which is due to the fact that the singular values do not decay to zero nearly as rapidly as they do in the deblurring case.

**Figure 1.4.** A plot of the log of the singular values (upper-left) and the right vectors $v_1$ (upper-right), $v_5$ (lower-left), and $v_{10}$ (lower-right) of $A$ for the one-dimensional deblurring example.
1.3 The statistical properties of $x_{LS}$ and ill-posedness

The least squares solution as defined by (1.24) is a random vector over the span of \{v_i\}_{i=1}^r. Specifically, note that $x_{LS} = \sum_{i=1}^r (v_i^T x_{LS}) v_i$, and hence from (1.24) and (1.15),

$$v_j^T x_{LS} = v_j^T x + \frac{u_j^T \epsilon}{\sigma_j} \sim N(v_j^T x, \sigma_j^2/\sigma_j^2),$$

(1.25)

for $j = 1, \ldots, r$, whereas $v_j^T x_{LS} = 0$ for $j = r+1, \ldots, n$. Equation (1.25) shows that the variance of $x_{LS}$ in the direction $v_j$ is $\sigma_j^2/\sigma_j^2$, which will be increasing in $j$ and quite large for large values of $j$. As a result, there is large uncertainty in the least squares estimator and, moreover, the uncertainty is largest in the directions of the highest frequency right singular vectors, which again, is supported by the plots of $x_{LS}$ in Figure 1.3.

Next, to obtain a numerical measure of the quality of the least squares solution, we compute the mean square error of $x_{LS}$, which is defined $\text{MSE}(x_{LS}) \equiv E(||x_{LS} -$
1.4. An illustrative example

We end the chapter with a simple two-variable ill-posed problem that nonetheless exhibits the characteristics, mentioned above, of ill-posed inverse problems in general. First we define the matrix $A$ via its SVD: let $v_1 = [1/\sqrt{2}, 1/\sqrt{2}]^T$, $v_2 = [-1/\sqrt{2}, 1/\sqrt{2}]^T$, and

$$A = v_1 v_1^T + 10^{-2} v_2 v_2^T = \begin{bmatrix} 0.505 & 0.495 \\ 0.495 & 0.505 \end{bmatrix}.$$ 

Note that we have used the outer product form of the SVD, as described in (1.21). Hence $v_1$ and $v_2$ are both the left and right singular vectors and the singular values $\sigma_1 = 1/\sqrt{2}$ and $\sigma_2 = 1/\sqrt{2}$, respectively.
12 Chapter 1. Characteristics of Inverse Problems

Figure 1.6. On the left are 1000 realizations from the data model \( \mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e} \), 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 \([1, 1]^T\).

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 \( \mathbf{v}_2 \).

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 \( \mathbf{e} = [0.026, 0.075]^T \) (a truncated realization of an iid Gaussian random vector with variance \( \sigma^2 = 0.01 \), then \( \mathbf{A}^{-1}\mathbf{b} = [-1.400, 3.501]^T \), which is far from \([1, 1]^T\); note that the left-hand side of (1.28) is 50.5, while the right-hand side is 1, implying that the problem is ill-posed.

Next, we plot 1000 realizations from the statistical model \( \mathbf{b} = \mathbf{A}\mathbf{x} + [\epsilon_1, \epsilon_2]^T \), with \( \epsilon_1, \epsilon_2 \sim \mathcal{N}(0, \sigma^2) \), on the left in Figure 1.6. Observe that, as expected, most – indeed, approximately 95% – of the samples lie within a distance \( 2\sigma = 0.2 \) of \( \mathbf{A}\mathbf{x} = [1, 1]^T \). Next, on the right in Figure 1.6, we plot the corresponding 1000 least squares solutions \( \mathbf{x}_{\text{LS}} = \mathbf{A}^{-1}\mathbf{b} \). Note that the mean of \( \mathbf{x}_{\text{LS}}, [1, 1]^T \), is contained within the data cloud, and that as expected from (1.25), most – again, approximately 95% – of the least squares solutions lie within a distance \( 2\sigma/\sigma_1 = 0.2 \) of \([1, 1]^T\) in the direction \( \mathbf{v}_1 \) and within a distance \( 2\sigma/\sigma_2 = 20 \) of \([1, 1]^T\) in the direction \( \mathbf{v}_2 \). That the samples of \( \mathbf{x}_{\text{LS}} \) are indeed distributed as (1.25) can be verified by computing normalized histograms of \( \mathbf{v}_1^T\mathbf{x}_{\text{LS}} \) and \( \mathbf{v}_2^T\mathbf{x}_{\text{LS}} \), for all \( \mathbf{x}_{\text{LS}} \), and comparing the resulting empirical densities with (1.25) (see Exercise 1.7).

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
Exercises

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 1.1.

(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) Assume the linear model \( b_i = x_1 + x_2 \ell_i + \epsilon_i \), where \( b_i \) denotes WHIP, \( \ell_i \) denotes AVG, and \( \epsilon_i \sim N(0, \sigma^2) \) for all \( i \). Write out by hand the first four rows of the design matrix \( A \). Construct \( b \) and \( A \) in MATLAB.

(c) Compute the least squares estimate. Note that \( x_{LS} \) is a 2 \( \times \) 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. **Midpoint quadrature.** The midpoint quadrature rule for a function \( f \) defined on \( [a, b] \) is defined by

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

where \( a \leq \eta \leq b \).

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

b. Determine an upper bound for the quadrature error in part (a) using (1.29) \((n \text{ times})\) and then verify that the inequality holds.

1.3. **Zero boundary conditions and Toeplitz blurring matrices.**

a. Show that when a zero boundary condition is assumed, (1.6) becomes (1.8), and (1.7) becomes an \( n \times n \) system of equations \( b = Ax \), where \( A \) is a Toeplitz matrix, i.e., \( A \) has constant diagonals. Finally, write down \( A \).
Chapter 1. Characteristics of Inverse Problems

b. Modify Deblurid.m so that the kernel

\[
a(s) = \begin{cases} 
100s + 10, & -\frac{1}{10} \leq s \leq 0, \\
-100s + 10, & 0 \leq s \leq \frac{1}{10}, \\
0, & \text{otherwise},
\end{cases}
\]

is used. Plot \(a\) on \([-1,1]\) and verify that the deblurring problem with this kernel is ill-posed using either (1.27) or (1.28).

c. Modify Deblurid.m so that the nonsymmetric kernel defined in KernelRecon.m is used, i.e.,

\[
a(s) = c \cdot \begin{cases} 
\exp \left(-s^2/(2\sigma_1^2)\right), & \sigma_1 = 0.1, \ -1 \leq s \leq 0, \\
\exp \left(-s^2/(2\sigma_2^2)\right), & \sigma_2 = 0.2, \ 0 < s \leq 1,
\end{cases}
\]

where \(c\) is chosen so that \(\int_{-1}^{1} a(s)ds = 1\). Note that in the previous examples the kernel was symmetric, and hence the toeplitz function in MATLAB was needed only the first row of \(A\). Since the kernel in this case is non-symmetric, the toeplitz function will require both the first row and column of \(A\). Plot \(a\) on \([-1,1]\) and verify that the deblurring problem with this kernel is ill-posed using either (1.27) or (1.28).

1.4. More on the kernel reconstruction inverse problem.

a. Verify that the quadrature described in (1.12) (and above) yields \(b = Ax\) with \(A\) defined in (1.13).

b. Show that \(A\) defined by (1.13) has inverse \(A^{-1}\) defined by (1.14). \textit{Hint:} You can use induction on \(n\).

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

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

1.5. The Singular Value Decomposition: Theory. Let \(A \in \mathbb{R}^{m \times n}\) have a singular value decomposition \(A = U \Sigma V^T\) as described in Section 1.2.1.

a. Using the orthonormality of the columns of \(U\) and \(V\), i.e., \(\{u_i\}_{i=1}^{m}\) and \(\{v_i\}_{i=1}^{n}\), show that

\[
Av_i = \begin{cases} 
\sigma_i u_i, & i = 1, \ldots, r \\
0, & i = r + 1, \ldots, n.
\end{cases}
\]

and

\[
A^T u_i = \begin{cases} 
\sigma_i v_i, & i = 1, \ldots, r \\
0, & i = r + 1, \ldots, m.
\end{cases}
\]
b. Using the columns of \( U \) and \( V \) as basis elements, write down bases for the four subspaces: the column space \( C(A) \), the row space \( C(A^T) \), the null space \( N(A) \), and the left null space \( N(A^T) \).

c. Show that (1.20) can be expressed (1.21) and hence that (1.22) can be expressed (1.23).

d. Prove that the least squares solution of \( Ax = b \) is given by \( x_{LS} = A^\dagger b \). *Hint:* substitute the SVD of \( A \) into \( \|Ax - b\| \).

e. Prove the following properties of the pseudo-inverse:
   
   i. \((A^\dagger)^T = (A^T)^\dagger\);
   
   ii. \(A^\dagger AA^\dagger = A^\dagger\);
   
   iii. \(A^\dagger = \lim_{\epsilon \to 0^+} (A^T A + \epsilon I)^{-1} A^T\).
   
   iv. \((A^T A)^\dagger = A^\dagger (A^\dagger)^T\).

1.6. **Statistical Properties of** \( x_{LS} \).

   a. Prove (1.15) using the linearity of the expectation function together with the fact that the mean of \( w \) is given by \( E(w) \), and the covariance of \( w \) is given by \( E((w - E(w))(w - E(w))^T) \).

   b. Use the result in part (a) to prove that the probability density for \( v_1^T x_{LS} \) is given by (1.25), where \( x_{LS} \) is given by (1.24).

   c. Verify the steps in the derivation of the mean square error for \( x_{LS} \) given by (1.26). *Hint:* you will need both part (a) and (b).

   d. Show that the ill-posedness inequality (1.28) implies the inequality (1.27).

1.7. Within **TwoVarTest.m** do the following: (i) use the \texttt{mean} and \texttt{cov} functions to verify that the empirical mean and covariance of the samples of \( x_{LS} \) are close to the theoretical values given in (1.25); (ii) compute the sample mean and variance of \( v_1^T x_{LS} \) and \( v_2^T x_{LS} \) and verify that they agree with the theoretical values given in (1.25); (iii) use \texttt{hist} to compute histograms of \( v_1^T x_{LS} \) and \( v_2^T x_{LS} \) and verify that they look Gaussian; and (iv) normalized histograms (i.e., so that they integrate to 1) and plot them together with the true normal densities defined by (1.25) to see that they agree.

1.8. **Deriving the least squares normal equations** \( A^T A x = A^T b \).

   a. Using Calculus: A function \( f : \mathbb{R}^n \to \mathbb{R} \) is convex if its Hessian \( \nabla^2 f(x) \) is positive semi-definite for all \( x \) and is strictly convex if \( \nabla^2 f(x) \) is positive definite for all \( x \). 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 \( x \) satisfying \( \nabla f(x) = 0 \), and this set has a single member (and hence \( f \) a unique minimizer) if \( f \) is strictly convex. Use these facts to show that if \( A \) has full column rank, \( \ell(x) = \frac{1}{2} \|Ax - b\|^2 \) has a unique minimizer given by (1.18).

   b. Using orthogonal projections: \( \|Ax - b\| \) is minimized when \( Ax \) is the element of the column space of \( A \), \( C(A) \), closest to \( b \), i.e., when \( Ax \) is
equal to the orthogonal projection of $b$ onto $C(A)$. Using this fact, and assuming that $A$ has full column rank, show that $x_{\text{LS}}$ is given by (1.18). 

*Hint:* use the formula for orthogonal projection onto the column space of a matrix.
Chapter 2

Spectral Methods and Regularization by Filtering

In this chapter, we present several methods that improve upon the least squares solution in inverse problems. The methods all perform some form of ‘filtering’ of the singular vectors along which the least squares solution has large variance. These techniques are known as regularization methods and we will interpret them as ‘filtered’ SVD, or spectral filtering, methods. The filters that we introduce are all dependent upon a single unknown parameter, known as the regularization parameter. Hence the practical usefulness of 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 1.4, 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 \( \epsilon_1, \epsilon_2 \sim \mathcal{N}(0, 0.01) \).

Recall our example: suppose \( \mathbf{x} = [1, 1]^T \), then in the noise-free case, \( \mathbf{b} = \mathbf{A}\mathbf{x} = [1, 1]^T \) and \( \mathbf{A}^{-1}\mathbf{b} = [1, 1]^T \), while if \( \mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{\epsilon} = [1.026, 1.075]^T \), then \( \mathbf{x}_{LS} = [-1.400, 3.501]^T \). Thus after adding a small amount of noise, the least squares solution is relatively far from the true solution \([1, 1]^T\). From our discussion in the previous chapter, this is due to high variance in \( \mathbf{x}_{LS} \) in the direction of the singular vector \( \mathbf{v}_2 = [-1/\sqrt{2}, 1/\sqrt{2}] \) with singular value \( \sigma_2 = 10^{-2} \).

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

\[
\mathbf{A}_{filt} = \sigma_1 \mathbf{v}_1 \mathbf{v}_1^T =
\begin{bmatrix}
    0.5 & 0.5 \\
    0.5 & 0.5
\end{bmatrix}.
\]
Chapter 2. Spectral Methods and Regularization by Filtering

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

\[
x_{\text{filt}} = \mathbf{A}_{\text{filt}}^+ \mathbf{b} = \sigma_1^{-1}(\mathbf{v}_1^T \mathbf{b}) \mathbf{v}_1 = \begin{bmatrix} 1.0505 \\ 1.0505 \end{bmatrix},
\]

where ‘+’ denotes pseudo-inverse [22], which is much nearer to the noise free solution \([1, 1]^T \) than is \( \mathbf{x}_{\text{LS}} \).

2.1.1 The Truncated Singular Value Decomposition

The approach of removing the singular values and vectors from \( \mathbf{A} \) 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

\[
\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_{i=1}^{r} \mathbf{u}_i \sigma_i \mathbf{v}_i^T,
\]

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 \( \Sigma \) in (2.1) by the \( m \times n \) diagonal matrix \( \Sigma_k \) with diagonal entries \( (\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0) \), yielding

\[
\mathbf{A}_k = \mathbf{U} \Sigma_k \mathbf{V}^T = \sum_{i=1}^{k} \mathbf{u}_i \sigma_i \mathbf{v}_i^T,
\]

which is known as a truncated singular value decomposition (TSVD) of \( \mathbf{A} \). The least squares solution of \( \mathbf{A}_k \mathbf{x} = \mathbf{b} \) yields the TSVD regularized solution

\[
\mathbf{x}_k = \mathbf{A}_k^+ \mathbf{b} = \mathbf{V} \Sigma_k^+ \mathbf{U}^T \mathbf{b} = \sum_{i=1}^{k} \left( \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \right) \mathbf{v}_i.
\]

Compare (2.3) with the analogous expression for the least squares solution (1.24).

The TSVD appears in many of the standard texts on inverse problems; see e.g., [27, 28, 29, 31, 49]. In the statistical literature, it is known as principal component regression [35] and is used to reduce the dimension of over-parameterized regression problems. Here, the truncation level \( k \) is 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.

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

\[
\mathbf{x}_\nu = \mathbf{V} \Phi_\nu \Sigma^\nu \mathbf{U}^T \mathbf{b} = \sum_{i=1}^{r} \phi_i(\nu) \left( \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \right) \mathbf{v}_i,
\]
2.1. Spectral Filtering Methods

Figure 2.1. A plot of the TSVD reconstruction $x_k$ for $k = 24$ in the deblurring case (left) and for $k = 20$ in the kernel reconstruction case (right) together with the true image.

where $\phi_i^{(v)}$, for $i = 1, \ldots, r$, are known as the filter factors, and

$$\Phi_v = \text{diag}(\phi_1^{(v)}, \ldots, \phi_r^{(v)}, 0, \ldots, 0) \in \mathbb{R}^{n \times n}. \quad (2.6)$$

The filter factors are parameterized by the variable $\nu$, satisfy $0 \leq \phi_i^{(v)} \leq 1$, and are chosen so that $\phi_i^{(v)} \approx 1$ for large singular values (small $i$) and $\phi_i^{(v)} \approx 0$ for the singular values that are near 0 (large $i$). For example, for the TSVD, if we let $\nu = k$ and

$$\phi_i^{(k)} = \begin{cases} 1 & i = 1, \ldots, k \\ 0 & i = k + 1, \ldots, r, \end{cases} \quad (2.7)$$

then it is straightforward to show (see Exercise 2.1) that (2.5) takes the form (2.3).

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, including [2, 7, 16, 27, 28, 29, 31, 49], as well as many others. The Tikhonov regularized solution is defined by

$$x_\alpha = \arg \min_x \left\{ \frac{1}{2} \|Ax - b\|^2 + \frac{\alpha}{2} \|x\|^2 \right\}, \quad (2.8)$$

which can be equivalently expressed (see Exercise 2.2) as the solution of the pseudonormal equations:

$$x_\alpha = (A^T A + \alpha I)^{-1} A^T b, \quad (2.9)$$

which reduced to the least squares when $\alpha = 0$. Statisticians call $x_\alpha$ the ridge estimator [34]. The Tikhonov solution $x_\alpha$ can also be written in the spectral forms (2.4) and (2.5), with $\nu = \alpha$, filter factors

$$\phi_i^{(\alpha)} = \frac{\sigma_i^2}{\sigma_i^2 + \alpha}, \quad i = 1, \ldots, r, \quad (2.10)$$
and $\Phi_\alpha$ defined as in (2.6); we leave the proof to Exercise 2.1. As was the case for the TSVD, for the we choose a value of the regularization parameter $\alpha$ that yields a good visual reconstruction of $x$. For $\alpha = 0.005$ in the deblurring case and $\alpha = 0.001$ in the kernel reconstruction case, the corresponding Tikhonov regularized solution $x_\alpha$ is plotted in Figure 2.2. Less subjective methods for choosing $\alpha$ are presented in Section 2.2.

2.1.3 Iterative Regularization

For inverse problems, iterative methods applied to the problem of minimizing $\|Ax-b\|^2$ often exhibit *semi-convergence*; that is, if $x_k$ is the $k^{th}$ iteration of the algorithm (not to be confused with $x_k$ defined by the TSVD), in early iterations $\|x_k-x\|^2$ decreases, but then eventually begins to increase as $x_k \to x_{LS} = A^{-1}b$. 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 [7, 16]:

$$x_k = x_{k-1} - \tau A^T(Ax_{k-1} - b).$$

(2.11)

We leave it to Exercise 4a for the reader to implement truncated Landweber iteration as a regularization method. If $x_0 = 0$, it can be shown (Exercise 4b) that (2.11) can be written in the spectral forms (2.4) and (2.5), with $\nu = k$, filter factors

$$\phi_i^{(k)} = 1 - (1 - \tau \sigma_i^2)^k, \quad i = 1, \ldots, r,$$

(2.12)

and $\Phi_k$ defined as in (2.6). Note that if $0 < \tau < 1/\sigma^2_1$, then $\phi_i^{(k)} \to 1$ and $x_k \to x_{LS}$ (Exercise 4c). We make this assumption and note that as the iteration proceeds, the filtering becomes less pronounced, eventually disappearing all together as the
iteration converges to \( \mathbf{x}_{\text{LS}} \). Moreover, the closer \( \tau \) is to \( 1/\sigma_i^2 \) the faster will be the convergence.

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 of these can be expressed in filtered SVD form (2.4), such as the steepest descent method (same as (2.11), 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 [37].

2.1.4 Statistical Properties of \( \mathbf{x}_\nu \)

Just as in the least squares case, we can analyze the statistical properties of the filtered solutions. First, substituting \( \mathbf{b} = \mathbf{A} \mathbf{x} + \mathbf{e} \) into (2.5), we obtain

\[
\mathbf{x}_\nu = \sum_{i=1}^{r} \phi_i^{(\nu)} (\mathbf{v}_i^T \mathbf{x}) \mathbf{v}_i + \sum_{i=r+1}^{n} \phi_i^{(\nu)} \left( \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \right) \mathbf{v}_i,
\]

(2.13)

where \( \nu = \alpha \) in the case of Tikhonov regularization and \( \nu = k \) in the case of TSVD and truncated Landweber iterations. From (2.13), we see that

\[
\mathbf{v}_i^T \mathbf{x}_\nu = \phi_i^{(\nu)} \left( \mathbf{v}_i^T \mathbf{x} + \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \right) \sim \mathcal{N} \left( \phi_i^{(\nu)} \mathbf{v}_i^T \mathbf{x}, \left( \phi_i^{(\nu)} \right)^2 \frac{\sigma^2}{\sigma_i^2} \right)
\]

(2.14)

for \( i = 1, \ldots, r, \) and \( \mathbf{v}_i^T \mathbf{x}_\nu = 0 \) for \( i = r+1, \ldots, n \). Comparing (2.14) to the analogous statement for \( \mathbf{x}_{\text{LS}} \) given by (1.25), we see that \( \mathbf{v}_i^T \mathbf{x}_\nu \overset{\text{dist}}{=} \phi_i^{(\nu)} \mathbf{v}_i^T \mathbf{x}_{\text{LS}} \), where ‘\( \overset{\text{dist}}{=} \)’ denotes ‘equal in distribution’. Thus for \( 0 < \phi_i^{(\nu)} < 1 \), in the direction \( \mathbf{v}_i \), the filtered solution \( \mathbf{x}_\nu \) has a smaller variance, by a factor of \( (\phi_i^{(\nu)})^2 \), than does the least squares solution \( \mathbf{x}_{\text{LS}} \), but bias is introduced, since the mean is no longer \( \mathbf{v}_i^T \mathbf{x} \).

Next, mimicking the derivation of the mean square error for \( \mathbf{x}_{\text{LS}} \), using (2.14) we compute the mean square error for \( \mathbf{x}_\nu \), defined \( \text{MSE}(\mathbf{x}_\nu) \overset{\text{def}}{=} E(\| \mathbf{x}_\nu - \mathbf{x} \|^2) \):

\[
\text{MSE}(\mathbf{x}_\nu) = E \left( \left\| \sum_{i=1}^{r} \phi_i^{(\nu)} \left( \mathbf{v}_i^T \mathbf{x} + \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \right) \mathbf{v}_i - \sum_{i=1}^{n} (\mathbf{v}_i^T \mathbf{x}) \mathbf{v}_i \right\|^2 \right)
\]

\[
= E \left( \sum_{i=1}^{r} \left[ (\phi_i^{(\nu)} - 1) \mathbf{v}_i^T \mathbf{x} + \phi_i^{(\nu)} \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \right]^2 + \sum_{i=r+1}^{n} (\mathbf{v}_i^T \mathbf{x})^2 \right)
\]

\[
= \sum_{i=1}^{r} (\phi_i^{(\nu)})^2 \frac{\sigma^2}{\sigma_i^2} + \sum_{i=r+1}^{n} (\phi_i^{(\nu)} - 1)^2 (\mathbf{v}_i^T \mathbf{x})^2 + \sum_{i=r+1}^{n} (\mathbf{v}_i^T \mathbf{x})^2. \quad (2.15)
\]

As in the least squares expression for the MSE, given by (1.25), the first sum on the right-hand side in (2.15) is the total variance of the estimator \( \mathbf{x}_\nu \), whereas the second two sums corresponds to the norm-squared of the bias, which is defined.
\( \mathbf{x}_{\text{bias}} = \mathbf{x} - E(\mathbf{x}_\nu) \). Note that if \( \phi^{(\nu)}_i = 1 \) for \( i = 1, \ldots, r \), i.e., there is no filtering, and the expression reduces to the mean square error for \( \mathbf{x}_{\text{LS}} \), given by (1.26). However, as the filter factors \( \phi^{(\nu)}_i > 0 \) decrease from 1 to 0, the total variance decreases, with the trade-off that the bias increases.

### 2.2 Regularization Parameter Selection Methods

The practical usefulness of the regularization methods presented above hinges on there being methods for choosing the regularization parameter \( \nu \). For a given regularization technique, a natural choice for \( \nu \) is the value that minimizes (2.15), or equivalently, the function

\[
\text{MSE}(\nu) = \sigma^2 \sum_{i=1}^{r} (\frac{\phi^{(\nu)}_i}{\sigma_i})^2 + \sum_{i=1}^{r} (\phi^{(\nu)}_i - 1)^2 (v_i^T \mathbf{x})^2.
\]  

(2.16)

Unfortunately, evaluating \( \text{MSE}(\nu) \) is infeasible in practice, since we do not know \( \mathbf{x} \). However, for the synthetic deblurring and kernel reconstruction test cases, since we know \( \mathbf{x} \), we can minimize expression for (2.16) for the TSVD, Tikhonov, and Landweber filters (see Exercise 2.6). Regularization methods do exist that minimize an estimator of the MSE, but we do not present any such methods here; the interested reader should see [3, 40, 45]. In the next several sub-sections, we present regularization parameter selection methods that do not require knowledge of \( \mathbf{x} \).

#### 2.2.1 The Unbiased Predictive Risk Estimator

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

\[
E \left( \| \mathbf{A} \mathbf{x}_\nu - \mathbf{A} \mathbf{x} \|^2 \right),
\]  

(2.17)

where \( \nu \) is the regularization parameter (\( \alpha \) or \( k \) in our examples). Since the unknown \( \mathbf{x} \) appears in (2.17), we can’t use it directly, but we can replace (2.17) by an unbiased estimator that does not contain \( \mathbf{x} \). The resulting method is appropriately known as the unbiased predictive risk estimator (UPRE) method; it was first introduced in [33], but can also be found in [49].

To obtain the estimator for (2.17), we first define \( \mathbf{A}_\nu \) to be the regularization matrix such that \( \mathbf{x}_\nu = \mathbf{A}_\nu \mathbf{b} \), which for the methods above has filtered SVD form (2.4), i.e., \( \mathbf{A}_\nu = \mathbf{V} \Phi^{(\nu)} \Sigma^{(\nu)} \mathbf{U}^T \). Then, \( \mathbf{A} \mathbf{A}_\nu = \mathbf{U} \Sigma \Phi^{(\nu)} \Sigma^{(\nu)} \mathbf{U}^T \), which can be shown to be symmetric, which we will use in calculating an expression for (2.17).

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

(2.18)
2.2. Regularization Parameter Selection Methods

where ‘tr’ denotes trace. The last equality follows from the fact that for a symmetric matrix \( B \) and white noise vector \( v \), \( E (v^T B v) = \text{tr}(B) \) [49, Lemma 7.2].

A similar calculation to that for deriving (2.18) gives us that

\[
E (\|Ax_\nu - b\|^2) = E (\|Ax_\nu - Ax\|^2) + \sigma^2 \text{tr}((AA_\nu)^2) - 2\sigma^2 \text{tr}(AA_\nu) + m\sigma^2,
\]

(2.19)

and hence, the predictive risk can be expressed

\[
U(\nu) = \|Ax_\nu - b\|^2 + 2\sigma^2 \text{tr}(AA_\nu) - m\sigma^2.
\]

(2.20)

Finally, estimating the expected value on the right-hand side of the previous equation by \( \|Ax_\nu - b\|^2 \), we obtain the unbiased predictive risk estimator

\[
U(k) = \sum_{i=k+1}^{r} (u_i^T b)^2 + 2\sigma^2 k
\]

(2.22)

and

\[
U(\alpha) = \sum_{i=1}^{r} \left[ \frac{\alpha^2 (u_i^T b)^2}{(\sigma_i^2 + \alpha)^2} + \frac{2\sigma^2 \sigma_i^2}{\sigma_i^2 + \alpha} \right].
\]

(2.23)

The expression for \( U \) for the Landweber iteration is left to Exercise 4.

We apply UPRE to the problem of choosing \( k \) for TSVD in the deblurring example and \( \alpha \) for Tikhonov regularization in the kernel reconstruction example. The reconstructions \( x_k \) and \( x_\alpha \) are plotted in Figure 2.3.

2.2.2 Generalized Cross Validation

One of the limitations of the UPRE selection method is that it requires knowledge of the noise variance \( \sigma^2 \).

A method similar to UPRE that does not require \( \sigma^2 \) is generalized cross validation (GCV). GCV was originally introduced in [51], is very commonly used in inverse problems (see, e.g., [2, 28, 49]), and is an approximation of leave-one-out cross validation (LOOCV) for large-scale problems.
The idea behind LOOCV is as follows. First, we define $A[i]x = b[i]$ to be the linear system that results after the $i$th element of $b$ and the $i$th row of $A$ are removed. We then define $x[i] = A[i]b[i]$, where $A[i]$ is the regularization matrix for $A[i]x = b[i]$, e.g., if Tikhonov regularization is used, $A[i] = ((A[i])^T A[i] + \alpha I)^{-1} (A[i])^T$. Ideally, $x[i]$ would accurately predict the missing data, i.e., $[Ax[i]]_i \approx b_i$. In the LOOCV approach, we choose the regularization parameter $\nu$ to minimize the predictive errors, $([Ax[i]]_i - b_i)^2$, for all $i$:

$$V(\nu) = \frac{1}{m} \sum_{i=1}^{m} (|Ax[i]|_i - b_i)^2.$$  

Evaluating the function $V$ at a single $\nu$ involves solving the $m$ large-scale inverse problems $A[i]x = b[i]$, and hence, LOOCV is infeasible – or at least highly inefficient – to implement for 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 [2] for details)

$$V(\nu) = \frac{1}{m} \sum_{i=1}^{m} \left( \frac{|Ax_\nu|_i - b_i}{1 - |AA_\nu|_ii} \right)^2.$$  

For many large-scale problems, the diagonal values of $AA_\nu$ are infeasible to compute. Thus the GCV approximation of $V(\nu)$ replaces $1 - |AA_\nu|_ii$ by $(m - tr(AA_\nu))/m$ for all $i$ to obtain

$$G(\nu) = \frac{m||Ax_\nu - b||^2}{|tr(I - AA_\nu)|^2}.$$  

(2.24)

The GCV choice of $\nu > 0$ is then the one that minimizes $G(\nu)$.
2.2. Regularization Parameter Selection Methods

Figure 2.4. On the left is a plot of the Tikhonov reconstruction \( x_\alpha \) with the GCV choice of \( \alpha = 0.0012 \) in the deblurring example. On the right is a plot of the TSVD reconstruction \( x_k \) with the GCV choice of \( k = 17 \) for the kernel reconstruction example.

Simplifying (2.24) using the SVD, as we did for UPRE above, and neglecting the factor \( m \), we obtain

\[
G(\nu) = \frac{\sum_{i=1}^{r}(\phi_i^{(\nu)} - 1)^2(u_i^Tb)^2 + \sum_{i=r+1}^{m}(u_i^Tb)^2}{(m - \sum_{i=1}^{r}\phi_i^{(\nu)})^2}, \tag{2.25}
\]

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

\[
G(k) = \frac{1}{(m-k)^2} \sum_{i=k+1}^{m}(u_i^Tb)^2, \tag{2.26}
\]

\[
G(\alpha) = \left( \sum_{i=1}^{r} \frac{\alpha^2(u_i^Tb)^2}{(\sigma_i^2 + \alpha)^2} + \sum_{i=r+1}^{m}(u_i^Tb)^2 \right) \left( m - \sum_{i=1}^{r} \frac{\sigma_i^2}{\sigma_i^2 + \alpha} \right)^2. \tag{2.27}
\]

The expression for the Landweber iteration it is left to Exercise 4.

We apply GCV to the problem of choosing \( k \) for TSVD in the deblurring example and \( \alpha \) for Tikhonov regularization in the kernel reconstruction example. The reconstructions \( x_k \) and \( x_\alpha \) are plotted in Figure 2.4. Note that the value of \( k \) is the same as was chosen using UPRE.

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., [2, 16, 28, 29, 31, 49]. DP advocates choosing \( \nu \) so that the sum of squared residuals \( \|Ax_\nu - b\|^2 \) is equal to the noise level \( E(\|\epsilon\|^2) = m\sigma^2 \), and it is equivalent to the method of moments in statistics on the second moment of the distribution. This amounts to solving to
Chapter 2. Spectral Methods and Regularization by Filtering

nonlinear equation

\[ 0 = D(\nu) \overset{\text{def}}{=} \|Ax_{\nu} - b\|^2 - m\sigma^2. \]  

(2.28)

The function \( D \) simplifies using the SVD to

\[ D(\nu) = \sum_{i=1}^{r} [(\phi_i^{(\nu)} - 1)u_i^Tb] + \sum_{i=r+1}^{m} (u_i^Tb)^2 - m\sigma^2, \]

(2.29)

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

\[ D(k) = \sum_{i=k+1}^{m} (u_i^Tb)^2 - m\sigma^2 \]

(2.30)

\[ D(\alpha) = \sum_{i=1}^{r} \frac{\alpha^2 (u_i^Tb)^2}{(\sigma_i^2 + \alpha)^2} + \sum_{i=r+1}^{m} (u_i^Tb)^2 - m\sigma^2. \]

(2.31)

The expression for \( D \) for the Landweber iteration it is left to Exercise 4d.

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 \( \alpha \). Alternatively, a minimization algorithm can be used to compute the minimizer of \( D(\nu)^2 \) – this is what we do – which will work even in cases where \( D(\nu) > 0 \) for all \( \nu > 0 \).

We apply DP to the problem of choosing \( k \) for TSVD in the deblurring example and \( \alpha \) for Tikhonov regularization in the kernel reconstruction example. The reconstructions \( x_k \) and \( x_\alpha \) are plotted in Figure 2.5. Note that the value of \( k \) is the same as was chosen using UPRE and GCV.

Figure 2.5. On the left is a plot of the Tikhonov reconstruction \( x_\alpha \) with the DP choice of \( \alpha = 0.0023 \) in the deblurring example. On the right is a plot of the TSVD reconstruction \( x_k \) with the DP choice of \( k = 17 \) for the kernel reconstruction example.
2.2. Regularization Parameter Selection Methods

2.2.4 The L-curve Criterion

The L-curve method is a heuristic developed in [26] 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 \( \alpha \). It can also be used for choosing the TSVD parameter \( k \), but it is a more involved procedure that we leave to [28] and the references therein. Finally, in order to simplify the formulas, we assume here that the rank of \( A \) is \( n \).

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

\[
(\xi(\alpha) \overset{\text{def}}{=} \log \|x_\alpha\|^2, \rho(\alpha) \overset{\text{def}}{=} \log \|Ax_\alpha - b\|^2),
\]

has a distinct L-shape. The corner of the curve corresponds, roughly, to the transition point between the values of \( \alpha \) near 0 for which \( \|x_\alpha\| \) is large and \( \|Ax_\alpha - b\| \) is small (too little regularization), and the large values of \( \alpha \) for which \( \|x_\alpha\| \) is small and \( \|Ax_\alpha - b\| \) is large (too much regularization). The L-curve method chooses \( \alpha \) corresponding to the corner, or point of maximum curvature, along the curve \((2.32)\), since this will often correspond to the value of \( \alpha \) at which this transition occurs.

We plot the L-curve for the 1D deblurring example on the left in Figure 2.6.

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

\[
C(\alpha) = \frac{\rho(\alpha)\xi''(\alpha) - \rho''(\alpha)\xi(\alpha)'}{(\rho'(\alpha)^2 + \xi'(\alpha))^3/2},
\]

For the parametric curve defined by \((2.32)\), \( C(\alpha) \) can be expressed entirely in terms of \( s(\alpha) \overset{\text{def}}{=} \|x_\alpha\|^2, r(\alpha) \overset{\text{def}}{=} \|Ax_\alpha - b\|^2 \) and \( s'(\alpha) \) [49]:

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

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

\[
s'(\alpha) = \sum_{i=1}^{r} -\frac{2\sigma_i^2(u_i^Tb)^2}{(\sigma_i^2 + \alpha)^3}.
\]

The L-curve choice for the regularization parameter \( \alpha \) is the value that maximizes the curvature \( C \) defined by \((2.33)\). 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 2.6. The resulting Tikhonov solution \( x_\alpha \) is plotted on the right in Figure 2.6.

2.2.5 Some Remarks on Parameter Selection Methods

A regularization method is said to be convergent if, roughly speaking, as the noise level (in our case represented by \( \sigma^2 \)) goes to 0, the methods choose values of the
 regularization parameter $\nu$ such that $x_\nu$ converges to $x$. An in-depth analysis of the convergence properties of the above methods can be found in [49]. 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 $\sigma^2$, while GCV does not. The L-curve method is shown to be non-convergent in [49], 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 deblurring example in this chapter.

Finally, we mention that since the data $b$ is a random vector, the value of $\nu$ given by the above methods will also be random. In all of the above test cases, the seed for the random number generator was fixed so that meaningful comparisons were possible. However, a Monte Carlo analysis, in which the each of the above methods is repeatedly run for different realizations of $b$, is useful for comparing performance. Such an analysis can be found for GCV, DP, and the L-curve in [28] and is also the topic of Exercise 13.

2.3 Periodic and Data Driven Boundary Conditions

In this section, we revisit the discretization of the convolution equation found in Chapter 1, Example 1.2. First, recall that a zero boundary condition was assumed, i.e., $x(s) = 0$ for $s \notin [0, 1]$, yielding an $n \times n$ linear system with a Toeplitz matrix $A$. In the next subsection, we instead assume a periodic boundary condition, i.e., that $x$ extends periodically with period 1 outside of $[0, 1]$, which yields a circulant matrix $A$ [49], which we will define below. We include this example (circular convolution) because of its ubiquitousness in image processing and its usefulness for obtaining
2.3. Periodic and Data Driven Boundary Conditions

computationally efficient algorithms, especially in two- and higher-dimensions. And finally, motivated by cases in which the zero and periodic boundary conditions yield poor results, we end the chapter with a description of what we call the data driven boundary condition. In this approach, the convolution equation is defined on an extended domain containing the domain of interest; a linear system is then derived via numerical integration; a regularized solution is obtained on the extended domain using one of the above boundary conditions; and finally, this solution is restricted to the domain of interest, removing any boundary artifacts. Both of the above techniques will be revisited in Chapter 3, where we focus on two-dimensional problems.

2.3.1 Periodic Boundary Conditions

In this subsection, we derive discrete cyclic convolution from continuous convolution given the assumption that the function \( x \) extends periodically, with period 1, outside of \([0, 1]\), i.e., \( x(s+1) = x(s) \) for all \( s \in \mathbb{R} \). We'll assume that \( b \) is measured on the interval \([0, 1]\), so that \( 0 \leq s \leq 1 \). For the numerical grid on \( s \), we take \( h = 1/n \), where \( n \) is even, \( s_i = (i - 1/2)h \), for \( i = 1, \ldots, n \), and define \( b_i = b(s_i) \). Next, we assume that \( a \) is supported on \([-1/2, 1/2)\), so that the full convolution reduces to

\[
\int_{-1/2}^{3/2} a(s - s')x(s')ds', \quad i = 1, \ldots, n.
\]

Next, we define the numerical grid on \([-1/2, 3/2]\):

\[
s_j' = (j - 1/2)h, \quad j = -n/2 + 1, \ldots, 3n/2, \quad \text{and} \quad x_j = x(s_j').
\]

Approximating the above integrals on this grid using mid point quadrature as in Example 1.1.2, and \( a_k = a(kh) \), yields

\[
b_i = h \sum_{j = i-n/2+1}^{i+n/2} a_{i-j}x_j, \quad i = 1, \ldots, n,
\]

where we have used the fact that \( a_{i-j} = a((i-j)h) \neq 0 \) only for \(-1/2 \leq (i-j)h < 1/2\).

Taking into account the periodicity of \( x \), which implies \( x_{1-j} = x_{n-j-1} \), \( x_{n+j} = x_j \) for \( j = 1, \ldots, n/2 \), the system of equations (2.35) defines an \( n \times n \) system of equations \( b = Ax \) with \( b = [b_1, \ldots, b_n]^T \), \( x = [x_1, \ldots, x_n]^T \), and \( A \) an \( n \times n \) circulant matrix [49]. Note that a circulant matrix is a Toeplitz (constant diagonals) matrix with the additional structure that each column can be obtained from the previous column by rotating it downward by one element. Thus circulant matrices are defined entirely by their first column. For the system defined by (2.35), the first column of \( A \) is given by

\[
a_s = h[a_0, a_1, \ldots, a_{n/2-1}, a_{-n/2}, \ldots, a_{-1}]^T.
\]

We leave the verification of this fact to Exercise 11.
If we define \( a_j^s = [a_s]_j \), then

\[
A = \sum_{j=1}^{n} a_j^s R^{j-1}, \quad \text{where} \quad R = \begin{bmatrix}
0 & 0 & \cdots & 0 & 1 \\
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & 1 & 0
\end{bmatrix}.
\] (2.37)

Note that circulant matrices are those that can be expressed as (2.37) for some \( a_s \). We can derive a diagonalization of \( A \) using this decomposition. First, if \( \omega = e^{-2\pi \sqrt{-1}/n} \), it can be shown (see, e.g., [49]) that

\[
R = F^* \text{diag}(1, \omega, \ldots, \omega^{n-1}) F,
\]

where \( F \) is the \( n \times n \) unitary discrete Fourier transform (DFT) matrix with components

\[
[F]_{ij} = \frac{1}{\sqrt{n}} \omega^{(i-1)(j-1)/n}, \quad 1 \leq i, j \leq n;
\] (2.38)

and \( F^* \) is the complex conjugate transpose of \( F \), which defines the unitary inverse discrete Fourier transform (IDFT). Note, moreover, that if we define

\[
\omega^{j-1} \equiv [1, \omega^{j-1}, \ldots, \omega^{(n-1)(j-1)}]^T,
\]

then we have from (2.37) that

\[
A = F^* \left( \sum_{j=1}^{n} a_j^s \text{diag}(\omega^{j-1}) \right) F
\]

\[
= F^* \text{diag} \left( \sum_{j=1}^{n} a_j^s \omega^{j-1} \right) F
\]

\[
= F^* \text{diag}(\sqrt{n} F a_s) F,
\] (2.39)

where \( a_s = h[a_0, a_1, \ldots, a_{n/2-1}, a_{-n/2}, \ldots, a_{-1}]^T \) is the first column of \( A \). The proof of the final equality (2.39) is left to Exercise 11.

In MATLAB, the vector \( a_s \) is obtained from the vector of kernel values \( a = h[a_{-n/2}, \ldots, a_{-1}, a_0, \ldots, a_{n/2-1}]^T \) via \( a_s = \text{fftshift}(a) \). Also, MATLAB uses the more typical, non-unitary definitions of the DFT and IDFT: \( \text{fft}(x) = \sqrt{n} F x \) and \( \text{idft}(x) = \frac{1}{\sqrt{n}} F^* x \), where \( \text{fft} \) and \( \text{idft} \) implement the fast DFT and IDFT, respectively. For optimal computational efficiency, \( n \) should be a power of 2, in which case implementation of \( \text{fft} \) and \( \text{idft} \) requires on the order of \( n \log_2 n \) operations, rather than \( n^2 \) operations for typical matrix-vector multiplication. This is a significant savings for large-scale problems, especially, as we will see in Chapter 3, in two-dimensions.

The spectral decomposition (2.39) can be used in precisely the same fashion as the SVD in the previous examples to compute regularized solutions, with the
exception that because the matrices are complex valued, matrix transpose is replaced by complex conjugate-transpose in all derivations. For example, if we define \( \hat{a}_s = \sqrt{n} F a_s \), the Tikhonov regularized solution can be expressed

\[
x_\alpha = (A^* A + \alpha I)^{-1} A^* b = F^* \text{diag} \left( \frac{\text{conj}(\hat{a}_s)}{|\hat{a}_s|^2 + \alpha I} \right) F b,
\]

where for the expression within \( \text{diag}(\cdot) \), \( \text{conj}(\cdot) \) denotes component-wise complex conjugate, \( | \cdot | \) denotes component-wise modulus, and division is component-wise.

It is straightforward to modify each of UPRE (2.23), GCV (2.27), DP (2.31), and L-curve (2.34) as well, to reflect the use of the decomposition (2.39). The DFT is used heavily in signal processing, and we will make use of it, and the periodicity assumption, again in the next chapter. However, because the results for the one-dimensional deblurring example are so similar for the two choices of boundary condition, we leave the implementation of (2.40) in this case to Exercise 11.

### 2.3.2 Data Driven Boundary Condition

The drawback to using either the zero or periodic boundary condition is that the assumptions lack physical motivation, and hence, when they are inaccurate, they result in non-physical artifacts in reconstructions. An alternative is to discretize the convolution equation on an extended domain, using one of the boundary conditions assumptions, compute a reconstruction \( x(s) \), and then restrict the result to the domain of interest, which in our case is \( s \in [0,1] \).

Here, we allow \( s \) to vary over the extended domain \([-1/2,3/2]\), and assume that \( a \) is measured on the interval \([-1,1]\), with \( a(s) = 0 \) for \( |s| \geq 1 \), so that the convolution model becomes

\[
b(s) = \int_{-3/2}^{5/2} a(s-s') x(s') ds', \quad -1/2 \leq s \leq 3/2.
\]

We then proceed precisely as in Example 1.2 – using midpoint quadrature with \( h = 1/n, \ s_j = s_j' = (j - \frac{1}{2}) h, \ b_i = b(s_i), \ x_j = x(s_j') \), and assuming a zero boundary condition, i.e., that \( x(s) = 0 \) for \( s \notin [-1/2,3/2] \) – to obtain the following system of equations:

\[
\begin{bmatrix}
 b_{-n/2+1} \\
 \vdots \\
 b_1 \\
 \vdots \\
 b_n \\
 \vdots \\
 b_{3n/2}
\end{bmatrix}
= h
\begin{bmatrix}
 a_0 & a_{-1} & \cdots & \cdots & a_{-2n+1} \\
 a_1 & a_0 & a_{-1} & \cdots & a_{-2n+2} \\
 \vdots & \vdots & \ddots & \ddots & \vdots \\
 a_{2n-2} & \cdots & a_1 & a_0 & a_{-1} \\
 a_{2n-1} & \cdots & \cdots & a_1 & a_0
\end{bmatrix}
\begin{bmatrix}
x_{-n/2+1} \\
\vdots \\
x_1 \\
\vdots \\
x_n \\
\vdots \\
x_{3n/2}
\end{bmatrix}.
\]

In (2.41), we define \( \hat{b} \) and \( x \) to be the \( 2n \times 1 \) vectors on the left and right, respectively, and \( \hat{A} \) to be the \( 2n \times 2n \) coefficient matrix.
Figure 2.7. 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 plot of the Tikhonov reconstructions using both the data driven and zero boundary conditions, together with the true image.

However, we have only observed $b = [b_1, \ldots, b_n]^T$, not all of $\hat{b}$, hence if $D$ is the indicator matrix on $[0, 1]$, given by rows $n/2 + 1$ to $3n/2$ of the $2n \times 2n$ identity matrix, we have $b = D\hat{b}$, and our linear model takes the form

$$b = D\hat{A}x + \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I). \quad (2.42)$$

Note that $b, \epsilon \in \mathbb{R}^n$, $x \in \mathbb{R}^{2n}$, and $A \overset{\text{def}}{=} D\hat{A} \in \mathbb{R}^{n \times 2n}$, thus estimating $x$ from $b$ is under-determined.

To reconstruct $x$ on $[0, 1]$, we compute a regularized solution of (2.42) and then restrict it to $[0, 1]$ via $Dx$. We call this the data driven boundary condition approach. In Figure 2.7, we plot the true image as well as data generated using (2.42) on the left, with $\sigma^2$ chosen so that the signal-to-noise ratio, $\|Ax\|/\sqrt{n\sigma^2}$, is 50. We reconstruct $x$ using Tikhonov regularization and both the data driven and zero boundary conditions, with $\alpha$ chosen in both cases using the discrepancy principle. These two solutions are plotted on the right in Figure 2.7, together with the true image. Due to the fact that the zero boundary condition ($x(s) = 0$ for $s \notin [0, 1]$) is a bad approximation of reality in this example, the corresponding reconstruction is poor near the boundaries, $t = 0, 1$.

An alternative to the data driven boundary condition for reducing boundary artifacts is to assume that $x$ satisfies a Neumann boundary condition, $x'(s) = 0$, for $s = 0, 1$. In this case, if the rather restrictive assumption that the kernel is symmetric holds, i.e., $a_i = a_{-i}$ for all $i$, then $A$ is a Toeplitz-plus-Hankel matrix and hence has a spectral representation similar to (2.39), but with $F$ and $F^*$ the discrete cosine and inverse cosine transforms, respectively; see [29] for details.
Exercises

2.1. The filtered SVD.
   a. Show that (2.5) follows from (2.4).
   b. Show that (2.5) with the TSVD filter factors (2.7) yields (2.3).
   c. Show that the Tikhonov solution (2.9) can be expressed in the form (2.5) with Tikhonov filter factors (2.10).

2.2. Tikhonov regularization: deriving the pseudo-normal equations. Define \( \ell(x) \) to be the penalized least squares function in (2.8). Show that \( \nabla \ell(x) = 0 \) yields (2.9) and hence that (2.8) and (2.9) are equivalent. Argue, moreover, that the \( x_\alpha \) is unique, which follows if the \( \nabla^2 \ell(x) \) is positive definite.

2.3. Plotting relative error curves.
   a. Create plots of the relative error function \( f(\alpha) = \| x_\alpha - x_{\text{true}} \| / \| x_{\text{true}} \| \) within both DeblurTikhonov.m and KernelReconTikhonov.m (use MATLAB's logspace function). Then compute the \( \alpha \) on your computational grid that minimizes the relative error and plot the corresponding regularization solution \( x_\alpha \).
   b. Repeat part (a) with DeblurTSVD.m and KernelReconTSVD.m.

2.4. Landweber iteration.
   a. Modify Deblur1d.m and KernelReconstruction.m from Chapter 1 so that truncated Landweber iteration is implemented as a regularization method. For what value of \( k \) is the relative error \( f(k) = \| x_{\text{true}} - x_k \| \) minimized in both cases? Plot the solution for that value of \( k \).
   b. Show that (2.11) can be written as (2.4) and (2.5), with \( \nu = k \), and filter factors (2.12). \textit{Hint:} First show that for \( x_0 = 0 \), (2.11) can be written \( x_k = \sum_{j=0}^{k-1} (I - \tau A^T A)^j (\tau A^T b) \). Then substitute the SVD of \( A \) into the expression.
   c. Prove that if \( 0 < \tau < 1/\sigma_1^2 \) the Landweber iteration (2.11) satisfies \( x_k \to x_{LS} \).
   d. Derive formulas for \( U(k) \), \( G(k) \), \( D(k) \), and MSE\((k) \) defined by (2.21), (2.25), (2.29), and (2.16), respectively, for the Landweber iteration.
   e. Modify your code from part (a) so that the values of \( f(k) \), \( U(k) \), \( G(k) \), \( D(k)^2 \), and MSE\((k) \) are monitored. Plot all of these curves in the same figure and report the value of \( k \) for which each of these functions are minimized. Plot at least one of the corresponding reconstructions.

2.5. Statistical properties of \( x_\nu \).
   a. Show that (2.5) implies (2.13).
   b. Show that (2.13) implies (2.14). \textit{Hint:} you will need (1.15).
   c. Fill in the details in the string of equalities used to derive the expression for the mean square error for \( x_\nu \) given by (2.15). Show, moreover, that \( \| x_{\text{bias}} \|^2 \) equals the second two sums in (2.15).
2.6. Mean square error.
   a. Find the expressions for the MSE defined by (2.16) for the TSVD and Tikhonov regularized solutions.
   b. Create plots of MSE(\(\alpha\)), using MATLAB’s \texttt{logspace} function, within both \texttt{DeblurTikhonov.m} and \texttt{KernelReconTikhonov.m}. Then compute the \(\alpha\) on your computational grid that minimizes the MSE and plot the corresponding regularized solution \(x_\alpha\).
   c. Repeat part (a) with \texttt{DeblurTSVD.m} and \texttt{KernelReconTSVD.m}.
   d. If you have done Exercise 2.3, how do the MSE and relative error curves compare? Plot the two curves together, for \texttt{DeblurTikhonov.m} and, separately, \texttt{KernelReconTikhonov.m}.

2.7. UPRE function derivation. Derive (2.19) and then show that, together with (2.18), it implies that the unbiased predictive risk can be written (2.20).

2.8. Plotting the UPRE, GCV, DP, and L-curve parameter selection curves.
   a. Modify \texttt{DeblurTikhonov.m} and \texttt{KernelReconTikhonov.m} so that the following curves are plotted together, (all on the same numerical grid created using \texttt{logspace}): 
      i. the UPRE curve \(U(\alpha)\) defined by (2.23),
      ii. the GCV curve \(G(\alpha)\) defined by (2.27),
      iii. the DP curve \(D(\alpha)^2\) defined by (2.31).
     Compute the \(\alpha\) that minimizes each curve on your computational grid and plot the corresponding regularization solution \(x_\alpha\).
   b. Repeat part a with \texttt{DeblurTSVD.m} and \texttt{KernelReconTSVD.m}.
   c. If you have done Exercises 3 and/or 6, plot the above parameter selection curves together with the relative error and/or the MSE curves. Create separate figures for the deblurring and kernel reconstruction test cases.

2.9. SVD analysis of the UPRE, GCV, and DP parameter selection methods
   a. Use the SVD of \(A\) to verify step (2.21) for UPRE.
   b. Use the SVD of \(A\) to verify that (2.24) implies (2.25) for GCV.
   c. Use the SVD of \(A\) to verify that (2.28) implies (2.29) for DP.

2.10. Alternate deblurring test cases. In Exercise 1.3 b and c, you were asked to modify \texttt{Deblur1d.m} so that different convolution kernels were used to define \(A\). Do the following in one or both of the resulting deblurring problems.
   a. Use Tikhonov regularization together with GCV and/or L-curve to reconstruct \(x\) from observations \(b\). What is the optimal regularization parameter \(\alpha\) in each case? Which gives the better reconstruction in your opinion?
   b. Use TSVD regularization together with UPRE and/or DP to reconstruct \(x\) from observations \(b\). 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 $x$ from observations $b$. What is the optimal stopping iteration $k$? Plot the regularized solution.

2.11. Deblurring with periodic boundary conditions and circulant matrices.

a. Derive the $n \times n$ coefficient matrix $A$ defined by (2.35) using the periodicity of $x$.

b. Verify (2.37) and the three equalities leading to equation (2.39).

c. Verify that when $A$ can be expressed (2.39), the Tikhonov regularized solution takes the form (2.40).

d. Modify `DeblurTikhonov.m` so that the spectral decomposition of $A$, given by (2.40), is used. Don’t construct $A$ explicitly, rather perform matrix-vector multiplication using

$$Ax = \text{real}(\text{ifft}(\text{ahat} \ast \text{fft}(x))), \quad \text{ahat} = \text{fft}(|\text{fftshift}(a)|),$$

where $a = [a_{n/2}, \ldots, a_{-1}, a_0, \ldots, a_{n/2-1}]^T$. Compute the Tikhonov regularized solution using (2.40), and make the necessary modifications to the expressions for UPRE (2.23), GCV (2.27), DP (2.31), and L-curve (2.34); these changes are straightforward and follow from setting $U = V = F^*$, $\Sigma = \text{diag}(\hat{a})$, and by replacing transpose by conjugate transpose wherever it appears. Report the value of $\alpha$ obtained using each of the parameter selection methods, and plot one of the reconstructions.

2.12. Data driven boundary conditions

a. Verify that the midpoint quadrature implementation, together with data driven boundary condition described in Section 2.3.2, yields the system of equations (2.41).

b. Modify `OnedDeblurBCs.m` so that it implements GCV and UPRE regularization parameter selection methods.

c. Modify `OnedDeblurBCs.m` so that, additionally, the reconstruction is computed when a periodic boundary condition is assumed on the extended interval $[0,1]$. To construct $A$, use one of the approaches in Exercise 11(b) and (e). Visually compare the results with those obtained using the data driven and zero boundary conditions?

2.13. PROJECT: Monte Carlo Analysis of regularization parameter selection methods. Modify `Deblur1dTikhonov.m` so that it repeatedly generates new data $b$ (to randomize the seed, use `rng('shuffle')`) and computes the following for each $b$:

i. $\alpha_{\text{UPRE}}, \alpha_{\text{GCV}}, \alpha_{\text{DP}},$ and $\alpha_{\text{opt}} = \arg\min_\alpha \|x_\alpha - x\|^2$.

ii. $R_{\text{UPRE}} = \alpha_{\text{UPRE}}/\alpha_{\text{opt}}, R_{\text{GCV}} = \alpha_{\text{GCV}}/\alpha_{\text{opt}},$ and $R_{\text{DP}} = \alpha_{\text{DP}}/\alpha_{\text{opt}}$.

Repeat this as many times as is feasible (say 500-1000) and then create histograms of the ratios computed in part (ii) above. What conclusion can you make about the performance of the regularization parameter selection
methods from the results? Does one of the methods perform better or worse than the others? Repeat the experiment for the kernel reconstruction problem by making the same modifications to `KernelReconTikhonov.m`. Do your conclusions about the performance of the regularization parameter selection methods change for this example?
Chapter 3

Two-Dimensional Test Cases

In this chapter, we present two classical two-dimensional imaging test cases. The first is two-dimensional image deblurring (deconvolution), which is very similar to the one-dimensional case, but which requires more sophisticated numerical techniques in order to keep computations efficient. In particular, the periodic boundary condition assumption that we ended with in Chapter 2 becomes one of the essential techniques for fast computations in two-dimensional image deblurring. The second example is computed tomography, which can be thought of as a two dimensional analogue of the kernel reconstruction problem introduced in Chapter 1, in the sense that an estimate of a function is sought from a collection of line integrals of the function.

3.1 Two-dimensional image deblurring

In two-dimensions, we consider the following convolution equation

\[ b(s, t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a(s - s', t - t') x(s', t') \, ds' \, dt', \quad 0 \leq s, t \leq 1. \tag{3.1} \]

We assume that \( a(s, t) = 0 \) for \( (s, t) \notin (-1, 1) \times (-1, 1) \), so that the model can be written instead

\[ b(s, t) = \int_{-1}^{1} \int_{-1}^{1} a(s - s', t - t') x(s', t') \, ds' \, dt', \quad 0 \leq s, t \leq 1. \tag{3.2} \]

As in one dimension, we use midpoint quadrature to discretize (3.2): let \( h = 1/n \) and define ordered pairs

\[ (s'_i, t'_j) = ((i - 1/2)h, (j - 1/2)h), \quad i, j = -n + 1, \ldots, 2n, \]

and \( (s_l, t_m) = (s'_i, t'_j) \), for all \( l, m = 1, \ldots, n \). Moreover, we define \( b_{lm} = b(s_l, t_m) \) and \( x_{ij} = x(s'_i, t'_j) \). The kernel is defined on the subinterval endpoints, since \( a(s_l - s_i, t_m - t_j) = a((l - i)h, (m - j)h) \). Moreover, \( a_{rs} = a(rh, sh) \) is nonzero, given our
Chapter 3. Two-Dimensional Test Cases

assumption, only for $-n + 1 \leq r, s \leq n - 1$. Then, ignoring quadrature error, midpoint quadrature applied to (3.2) yields the following system of linear equations, analogous to (1.6),

$$b_{lm} = h^2 \sum_{i=l-n+1}^{l+n-1} \sum_{j=m-n+1}^{m+n-1} a_{l-i, m-j}x_{ij}, \quad l, m = 1, \ldots, n.$$  \hspace{1cm} (3.3)

We now discuss different boundary conditions assumptions and kernel structures. We will not provide an exhaustive list, since this topic has been treated in detail in the books [29, 49]. The first case we consider is zero boundary conditions with a separable kernel, mainly because this case connects nicely to one-dimensional deblurring, but also because it is computationally efficient. We then present the periodic boundary condition case, because it is both very flexible and very computationally efficient. Moreover, these two assumptions allow for the use of the spectral methods of Chapter 2. And finally, we present the extension of the data driven boundary condition considered at the end of Chapter 2 to two-dimensions. The resulting method cannot be solved using spectral methods, and hence we present the conjugate gradient iterative method for its solution.

3.1.1 Zero Boundary Conditions with Separable Blur

The kernel $a$ in (3.2) is separable if it can be written $a(s, t) = a_1(s)a_2(t)$ [29, 38]. We consider this as our first two dimensional example because the resulting discretization can be expressed in terms of one-dimensional blurring matrices as defined in Chapter 1. An example of a separable kernel is a Gaussian with diagonal covariance, i.e.,

$$a(s, t) = \frac{1}{\sqrt{(2\pi)^2\gamma_1^2\gamma_2^2}} \exp\left(-\frac{1}{2}(s^2/\gamma_1^2 + t^2/\gamma_2^2)\right), \quad \gamma_1, \gamma_2 > 0.$$  \hspace{1cm} (3.4)

We will use this kernel in what follows. Note that $a(s, t) = a_1(s)a_2(t)$, with $a_1(s) = (2\pi\gamma_1^2)^{-1/2} \exp(-s^2/(2\gamma_1^2))$ 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 (1.9). It should not be a surprise, therefore, that the resulting inverse problem will be ill-posed.

Returning to (3.3), we note that in the separable case

$$a_{l-i, m-j} = a((l-i)h, (m-j)h) = a_1((l-i)h)a_2((m-j)h).$$

If we define $a_{l-i}^{(1)} = a_1((l-i)h)$ and $a_{m-j}^{(2)} = a_2((m-j)h)$. Then, taking note of the fact that $x_{ij} \neq 0$ only for $i, j = 1, \ldots, n$, equation (3.3) can be expressed

$$b_{lm} = h \sum_{j=1}^{n} a_{m-j}^{(2)} \left(h \sum_{i=1}^{n} a_{l-i}^{(1)}x_{ij}\right), \quad l, m = 1, \ldots, n.$$  \hspace{1cm} (3.5)

Next, we define $B$ and $X$ to be the $n \times n$ arrays with components $[B]_{lm} = b_{lm}$, for $l, m = 1, \ldots, n$ and $[X]_{ij} = x_{ij}$, for $i, j = 1, \ldots, n$, respectively, and $A_1$ and $A_2$
3.1. Two-dimensional image deblurring

To be the \( n \times n \) one dimensional Toeplitz blurring matrices corresponding to the kernels \( a_1 \) and \( a_2 \), respectively, derived as in Example 1.2 on the numerical grid defined above. Then, (3.5) can be equivalently expressed

\[
b_{lm} = h \sum_{j=1}^{n} a_{m-j}^{(2)} [A_1 X]_{lj} = h \sum_{j=1}^{n} a_{m-j}^{(2)} [(A_1 X)^T]_{jl} = [A_2 (A_1 X)^T]_{ml},
\]

and hence

\[
B = A_1 X A_2^T.
\]

As we will see below, doing computations in array form, as in equation (3.7), is handy, however in order to unify our presentation with that of the previous chapters, and also to simplify our later discussions, we will express (3.7) in the matrix-vector form \( \mathbf{b} = \mathbf{A} \mathbf{x} \). First, 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, from left to right, with the first column on top and last column on bottom. Then it can be shown that

\[
\mathbf{A} \mathbf{x} \overset{\text{def}}{=} \text{vec}(\mathbf{A}_1 \mathbf{X} \mathbf{A}_2^T) = (\mathbf{A}_2 \otimes \mathbf{A}_1) \mathbf{x},
\]

where ‘\( \otimes \)’ denotes Kronecker product and is defined, for matrices \( \mathbf{C} \in \mathbb{R}^{m \times n} \) and \( \mathbf{D} \in \mathbb{R}^{r \times s} \), by

\[
\mathbf{C} \otimes \mathbf{D} = \begin{bmatrix}
  c_{11} \mathbf{D} & c_{12} \mathbf{D} & \cdots & c_{1n} \mathbf{D} \\
  c_{21} \mathbf{D} & c_{22} \mathbf{D} & \cdots & c_{2n} \mathbf{D} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{m1} \mathbf{D} & c_{m2} \mathbf{D} & \cdots & c_{mn} \mathbf{D}
\end{bmatrix}_{mr \times ns},
\]

with \( c_{ij} = [\mathbf{C}]_{ij} \). We leave a derivation of the details of (3.8) to Exercise 2.

In what follows, we will need the following additional properties of the Kronecker product: for matrices \( \mathbf{C} \in \mathbb{R}^{m \times n}, \mathbf{D} \in \mathbb{R}^{r \times s}, \mathbf{E} \in \mathbb{R}^{n \times p}, \) and \( \mathbf{F} \in \mathbb{R}^{s \times t}, \) the following hold (see [29, 48]):

\[
(\mathbf{C} \otimes \mathbf{D})^T = \mathbf{C}^T \otimes \mathbf{D}^T
\]

\[
(\mathbf{C} \otimes \mathbf{D})^\dagger = \mathbf{C}^\dagger \otimes \mathbf{D}^\dagger
\]

\[
(\mathbf{C} \otimes \mathbf{D})(\mathbf{E} \otimes \mathbf{F}) = (\mathbf{CE}) \otimes (\mathbf{DF}),
\]

where “\( \dagger \)” denotes pseudo-inverse, as defined in Chapter 1. Then if \( \mathbf{A}_1 = \mathbf{U}_1 \Sigma_1 \mathbf{V}_1^T \) and \( \mathbf{A}_2 = \mathbf{U}_2 \Sigma_2 \mathbf{V}_2^T \) are the SVDs of \( \mathbf{A}_1 \) and \( \mathbf{A}_2 \), respectively, we obtain

\[
\mathbf{A} = (\mathbf{U}_2 \otimes \mathbf{U}_1)(\mathbf{\Sigma}_2 \otimes \mathbf{\Sigma}_1)(\mathbf{V}_2^T \otimes \mathbf{V}_1^T),
\]

which is, essentially, the SVD of \( \mathbf{A} \), except that the diagonal elements of \( \mathbf{\Sigma}_2 \otimes \mathbf{\Sigma}_1 \) are no longer in descending order. Moreover, the least squares solution can be expressed

\[
\mathbf{x}_{\text{LS}} = \mathbf{A}^\dagger \mathbf{b} = (\mathbf{V}_2 \otimes \mathbf{V}_1)(\mathbf{\Sigma}_2^\dagger \otimes \mathbf{\Sigma}_1^\dagger)(\mathbf{U}_2^T \otimes \mathbf{U}_1^T)\mathbf{b},
\]
Figure 3.1. Two dimensional deblurring example. The true image $X$ is given on the upper-left; the blurred and noisy data $B$ is given on the upper-right; the least squares solution $X_{LS}$ is given on the bottom; and the Tikhonov regularized solution $X_\alpha$ with $\alpha = 0.00087$ chosen UPRE is given on the lower-right.

or equivalently, in array form,

$$X_{LS} = A_1^T B (A_2^T)^\dagger = V_1 (\Sigma_1^T U_1^T B U_2 ) (\Sigma_2^T)^T V_2^T. \tag{3.14}$$

We leave the details of the derivations of (3.12-3.14) to Exercise 2.

Adding iid Gaussian noise then yields a linear statistical model of the form $b = Ax + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I_{n^2})$, or equivalently, $B = A_1 X A_2^T + E$, where $\text{vec}(E) \sim \mathcal{N}(0, \sigma^2 I_{n^2})$. We choose $\sigma^2$ so that the signal-to-noise ratio (SNR), $\|Ax\|/\sqrt{n^2 \sigma^2}$, is 50. A synthetically generated satellite image $X$, blurred noisy data $B$, and the least squares solution $X_{LS}$ are plotted in Figure 3.1. Note that as for the deblurring problem in Chapter 1, the problem is severely ill-posed, and hence the least squares solution is worthless.

Next, we discuss implementation of Tikhonov regularization in this setting, and leave the implementation of TSVD to the Exercise 7. First, the following result
3.1. Two-dimensional image deblurring

will be useful: let \( a \in \mathbb{R}^m \) and \( b \in \mathbb{R}^n \), then

\[
A = \text{diag}(a), \quad B = \text{diag}(b) \quad \implies \quad A \otimes B = \text{diag(vec}(ba^T))).
\] (3.15)

This will allow us to perform efficient multiplication by \( A \) using (3.12):

\[
Ax = \text{vec}(U_1((\sigma_1\sigma_2^T) \odot (V_1^T XV_2))U_2^T).
\] (3.16)

Moreover, analogous arguments allow us to express the Tikhonov solution using only the SVDs of \( A_1 \) and \( A_2 \), making the computation very efficient:

\[
X_\alpha = V_1 \left( \left( \frac{\sigma_1\sigma_2^T}{(\sigma_1\sigma_2^T)^2 + \alpha 1} \right) \odot (U_1^T BU_2^T) \right) V_2^T.
\] (3.17)

where the division and squaring of arrays is component-wise, ‘\( \odot \)’ denotes component-wise multiplication, and \( 1 \) is an \( n \times n \) array of 1s. We leave the details of the derivation of (3.16) and (3.17) to Exercise 2.

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 (2.23), and the DP function \( D(\alpha) \) defined by (2.31):

\[
U(\alpha) = \sum_{|\sigma_1\sigma_2^T|_{ij} > 0} \left[ \frac{\alpha^2[U_1^T BU_2^T]_{ij}^2}{[(\sigma_1\sigma_2^T)_{ij}^2 + \alpha]^2} + \frac{2\sigma_1\sigma_2^T_{ij}^2}{(\sigma_1\sigma_2^T)_{ij}^2 + \alpha} \right],
\] (3.18)

\[
D(\alpha) = \sum_{|\sigma_1\sigma_2^T|_{ij} > 0} \frac{\alpha^2[U_1^T BU_2^T]_{ij}^2}{[(\sigma_1\sigma_2^T)_{ij}^2 + \alpha]^2} + \sum_{|\sigma_1\sigma_2^T|_{ij} = 0} [U_1^T BU_2^T]_{ij}^2 - n^2\sigma^2.
\] (3.19)

Note that the singular values of \( A \) are the positive values of the \( n \times n \) array \( \sigma_1\sigma_2^T \) and they are not ordered, hence the sum over \( |\sigma_1\sigma_2^T|_{ij} > 0 \). Analogous expressions for the GCV and L-curve functions can also be derived, which we leave to Exercise 3. Minimizing \( U(\alpha) \) using MATLAB’s \texttt{fminbnd} function for the example shown in Figure 3.1, with kernel \( a \) of the form (3.4), we obtain regularization parameter \( \alpha = 0.00087 \) (for details see the accompanying code \texttt{Deblur2dSeparable.m}). We plot the corresponding Tikhonov regularized solutions in Figure 3.1.

Finally, we note that the above equations will also hold for \( x \) defined on a rectangular domain, leading to \( X \in \mathbb{R}^{n \times k} \), \( A_1 \in \mathbb{R}^{n \times n} \) and \( A_2 \in \mathbb{R}^{k \times k} \) Toeplitz matrices, and \( B \in \mathbb{R}^{n \times k} \).

3.1.2 Periodic boundary conditions

The assumption of a separable kernel will rarely be accurate in practice. Returning to the discrete convolution (3.3), if the kernel is non-separable and a zero boundary condition is assumed, the resulting linear system \( b = Ax \) has a matrix \( A \) that is block Toeplitz with Toeplitz blocks (BTTB) [49]. BTTB matrices are not efficiently diagonalizable, however fast algorithms exist for computing regularized solutions of BTTB systems. They involve embedding the BTTB matrix \( A \) into a larger matrix that is block circulant with circulant blocks (BCCB); the interested reader
should see [49] and Exercise 14 for details. BCCB matrices arise in the numerical discretization of (3.1) when the unknown image $x$ extends periodically outside of the domain $[0, 1] \times [0, 1]$, i.e., $x(s + 1, t + 1) = x(s, t)$ for all $(s, t) \in \mathbb{R}^2$. We treat this case now.

First, we define the same numerical grid for $(s, t) \in [0, 1] \times [0, 1]$ as above: $(s_l, t_m) = ((l - 1/2)h, (m - 1/2)h)$ with $h = 1/n$, where $n$ is even, and $l, m = 1, \ldots, n$. Moreover, similar to as in the one-dimensional case, we assume that $a$ is nonzero on $[-1/2, 1/2] \times [-1/2, 1/2]$ and define $b_{lm} = b(s_l, t_m)$ so that (3.1) reduces to

$$b_{lm} = \int_{-1/2}^{1/2} \int_{-1/2}^{1/2} a(s_l - s', t_m - t') x(s', t') ds' dt', \quad 1 \leq l, m \leq n. \quad (3.20)$$

To discretize this integral, we use the following numerical grid for $(s', t') \in [-1/2, 3/2] \times [-1/2, 3/2]$: $(s'_i, t'_j) \overset{\text{def}}{=} ((i - 1/2)h, (j - 1/2)h), \quad i, j = -n/2 + 1, \ldots, 3n/2.$

In addition, we define $x_{ij} = x(s'_i, t'_j)$ and store the kernel values $a_{r,s} = a(rh, sh)$, for $-n/2 \leq r, s \leq n/2 - 1$, in the array

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

Moreover, if we express $a$ in the block form

$$a = h^2 \begin{bmatrix}
    \overbrace{a_{--}}^{a_-} & \overbrace{a_{+-}}^{a_+} \\
    \overbrace{a_{++}}^{a_+}
\end{bmatrix},$$

the circular shift of $a$ is defined

$$a_s \overset{\text{def}}{=} h^2 \begin{bmatrix}
    \overbrace{a_{++}}^{a_+} & \overbrace{a_{+-}}^{a_+} \\
    \overbrace{a_{--}}^{a_-}
\end{bmatrix}, \quad (3.21)$$

which will be needed below. Returning to (3.20), we apply midpoint quadrature to obtain

$$b_{lm} = h^2 \sum_{i=-n/2+1}^{l+n/2} \sum_{j=m-n/2+1}^{m+n/2} a_{l-i, m-j} x_{ij}, \quad l, m = 1, \ldots, n,$

$$= \sum_{j=m-n/2+1}^{m+n/2} \left( h^2 \sum_{i=-n/2+1}^{l+n/2} a_{l-i, m-j} x_{ij} \right), \quad l, m = 1, \ldots, n, \quad (3.22)$$

where we have used the fact that $a_{ij} \neq 0$ only for $i, j \in \{-n/2, \ldots, n/2 - 1\}$. 


3.1. Two-dimensional image deblurring

Taking into account the periodicity of $x_{ij}$ with respect to $i$, the inner-sum in (3.22) defines a discrete circular convolution of the form (2.35). Specifically, if we define $x_j$ to the $j$th column of $X$ and $\hat{A}_j$ to be the circulant matrix with first column $[a_0, m-j, \ldots, a_{n/2-1}, m-j, a_{-n/2}, m-j, \ldots, a_{-1}, m-j]^T$, then (3.22) can be equivalently expressed

$$b_{lm} = \sum_{j=m-n/2+1}^{m+n/2} [\hat{A}_{m-j} x_j]_l, \quad l, m = 1, \ldots, n.$$  \hspace{1cm} (3.23)

Next, we take into account the periodicity of $x_{ij}$ with respect to $j$, define $b_m$ to be the $m$th column of $B$, and define $A_{j-1}$ to be the circulant matrix with first column given by the $j$th row of $a_s$, defined in (3.21). Then (3.23) can be written

$$b_m = \sum_{j=1}^{m} A_{m-j} x_j + \sum_{j=m+1}^{n} A_{n+m-j} x_j,$$  \hspace{1cm} (3.24)

and hence,

$$\begin{bmatrix} b_1 \\
 b_2 \\
 \vdots \\
 b_n \end{bmatrix} = \begin{bmatrix} A_0 & A_{n-1} & \cdots & A_2 & A_1 \\
 A_1 & A_0 & A_{n-1} & \cdots & A_2 \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 A_{n-2} & \cdots & A_1 & A_0 & A_{n-1} \\
 A_{n-1} & A_{n-2} & \cdots & A_1 & A_0 \end{bmatrix} \begin{bmatrix} x_1 \\
 x_2 \\
 \vdots \\
 x_n \end{bmatrix}.$$ \hspace{1cm} (3.25)

Note that both the block structure and each individual block of the coefficient matrix in (3.25), which we will denote $A$, is circulant. Hence we say that $A$ is block circulant with circulant blocks (BCCB).

We can now diagonalize $A$ in (3.25) by extending the arguments for the diagonalization of circulant matrices found in Section 2.3.1. First, note that

$$A = \sum_{j=1}^{n} R_j^{-1} \otimes A_{j-1},$$ \hspace{1cm} (3.26)

where $R$ is defined in (2.37). Both $R$ and $A_j$ are circulant matrices, and hence, can be diagonalized by the DFT matrix $F$ defined in (2.38). First, recall from Section 2.3.1 that $R_j^{-1} = F^* \text{diag}(\omega_j^{-1}) F$, where $\omega_j^{-1} \triangleq [1, \omega_j^{-1}, \ldots, \omega_j^{(n-1)(j-1)}]^T$ with $\omega = e^{-2\pi \sqrt{-1}/n}$. Moreover, from (2.39), we have

$$A_{j-1} = F^* \text{diag}(\sqrt{n}F a_j^*) F,$$

where $a_j^* \in \mathbb{R}^n$ is defined to be the $j$th row of $a_s$. Hence, using the properties of
the Kronecker product (3.9-3.11), we obtain from (3.26)

\[
A = \sum_{j=1}^{n} \left( F^* \text{diag}(\omega_j^{j-1}) F \right) \otimes \left( F^* \text{diag} \left( \sqrt{n} F a_j \right) F \right)
\]

\[
= \left( F^* \otimes F^* \right) \left( \sum_{j=1}^{n} \text{diag}(\omega_j^{j-1}) \otimes \text{diag} \left( \sqrt{n} F a_j \right) \right) \left( F \otimes F \right)
\]

\[
= \left( F^* \otimes F^* \right) \text{diag} \left( \sqrt{n} \text{vec} \left( \sum_{j=1}^{n} F a_j (\omega_j^{j-1})^T \right) \right) \left( F \otimes F \right)
\]

\[
= \left( F^* \otimes F^* \right) \text{diag} \left( n \text{vec}(F a F^T) \right) \left( F \otimes F \right).
\]

(3.27)

Note that the third equality follows from (3.15) and \( F^T = F \). The two-dimensional discrete Fourier transform (DFT) and inverse DFT are defined by \( F_{2D} = F \otimes F \) and \( F^*_{2D} = F^* \otimes F^* \), and hence (3.27) implies

\[
A = F_{2D}^* \text{diag}(n F_{2D} \text{vec}(a)) F_{2D}.
\]

(3.28)

In practice, computations are done on the arrays \( X \) and \( B \), such that \( b = \text{vec}(B) \) and \( x = \text{vec}(X) \). Specifically, if we define the functions \( \text{DFT}(X) = FXF \) and \( \text{IDFT}(X) = F^*XF^* \), by (3.8) and (3.28), we have

\[
Ax = \text{vec}(\text{IDFT}(\tilde{a} \odot \text{DFT}(X))),
\]

(3.29)

where \( \tilde{a} = n \text{DFT}(a) \) and \( \odot \) denotes component-wise multiplication. The noise-free \( n \times n \) data array \( B \) can therefore be computed by

\[
B = \text{IDFT}(\tilde{a} \odot \text{DFT}(X)).
\]

(3.30)

In MATLAB, the more standard, non-unitary DFT and IDFT are implemented, this time by the functions \( \text{fft} \) and \( \text{ifft} \), respectively, which satisfy \( \text{fft}(V) = n \text{DFT}(V) \) and \( \text{ifft}(V) = \frac{1}{n} \text{IDFT}(V) \). Just as in one dimension, the implementation uses the fast Fourier transform (FFT) algorithm, which is most efficient if \( n \) is a power of 2, in which case \( F_{2D}v \) is computed in order \( n^2 \log_2 n \) operations, as opposed to \( n^4 \) operations for standard matrix-vector multiplication. Hence a huge computational savings is gained, especially for large \( n \), in addition to the fact that only the \( n^2 \) eigenvalues in \( \tilde{a} \), rather than all \( n^4 \) entries of \( A \), need to be stored.

We generate noise-free data \( B \) using (3.30) with the same satellite image \( X \) and kernel \( a \) as in the separable blur case. We then add iid Gaussian noise to each \( [B]_{ij} \) with noise variance \( \sigma^2 \) chosen so that the SNR is, once again, 50. As one would expect, the resulting noisy data \( B \) is visually indistinguishable from what appears in Figure 3.1.

Moreover, note that (3.28) 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
3.1. Two-dimensional image deblurring

directly applied. First, we note that the Tikhonov regularized solution takes the form \( x_\alpha = (A^*A + \alpha I)^{-1}A^*b \), or in array form

\[
X_\alpha = \text{IDFT}\left(\frac{\text{conj}(\hat{a}_s)}{|\hat{a}_s|^2 + \alpha} \otimes \text{DFT}(B)\right),
\]

where ‘\text{conj}’ denotes component-wise complex conjugate; \(|\hat{a}_s|^2\) is the \( n \times n \) array whose components are the squared magnitudes of the complex elements of \( \hat{a}_s \); \( I \) is an \( n \times n \) array of 1’s; and division is component-wise.

The GCV function \( G(\alpha) \) defined by (2.27) takes the form

\[
G(\alpha) = \sum_{|i|, |j| > 0} \left( \frac{\alpha^2 |\hat{B}_{ij}|^2}{(|\hat{a}_s|_{ij}^2 + \alpha)^2} + \sum_{|i|, |j| = 0} |\hat{B}_{ij}|^2 \right) \left( n^2 - \sum_{|i|, |j| > 0} |\hat{a}_s|_{ij}^2 / (|\hat{a}_s|_{ij}^2 + \alpha)^2 \right),
\]

where \( \hat{B} = \text{DFT}(B) = \frac{1}{n} \text{fft}2(B) \). The expression for the L-curve function \( C(\alpha) \) does not change (see (2.33)), however \( s'(\alpha) \) given by (2.34) becomes

\[
s'(\alpha) = \sum_{|i|, |j| > 0} \frac{-2 |\hat{a}_s|_{ij}^2 |\hat{B}_{ij}|^2}{(|\hat{a}_s|_{ij}^2 + \alpha)^3}.
\]

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 \text{fminbnd} function for the example shown in Figure 3.1, we obtain values of \( \alpha = 0.00083 \) and \( \alpha = 0.0034 \), respectively. Details of this implementation can be found in \text{Deblur2dPeriodic.m}. The corresponding Tikhonov regularized solutions are plotted in Figure 3.2.

3.1.3 Data Driven Boundary Conditions

Assuming zero or periodic boundary conditions for the unknown \( x \) can be problematic, particularly if the true \( x \) is spatially varying and nonzero near the boundaries of \([0,1] \times [0,1]\). As an example, consider the deblurring problem presented by the example in Figure 3.3. The data was generated by first convolving a \( 256 \times 256 \) true image with the same Gaussian kernel used in the previous example, then by extracting a \( 128 \times 128 \) subimage, and finally, by adding white Gaussian noise with a variance \( \sigma^2 \) chosen so that the signal-to-noise is, once again, 50. We leave it to the reader in Exercise 6 to verify that if a periodic boundary condition is assumed on the sub-image, the reconstruction is very poor.

We propose dealing with the boundary artifact issue by instead using the data driven boundary condition idea presented for one dimensional deblurring in Section 2.3.2. This approach was first used for two dimensional imaging deblurring in [8]. Analogous to the one dimensional case, we extend the domain for \((s,t)\) to \([-1/2,3/2] \times [-1/2,3/2]\), but here we assume a periodic (rather than zero) boundary
Chapter 3. Two-Dimensional Test Cases

Figure 3.2. The Tikhonov regularized solutions for two-dimensional de-blurring examples. On the left is the reconstruction in the periodic boundary conditions case with GCV choice of $\alpha = 0.00083$. On the right is the reconstruction in the periodic boundary condition case with the L-curve choice of $\alpha = 0.0034$.

Figure 3.3. 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.

condition, so that $a$ is assumed to be measured on $[-1, 1] \times [-1, 1]$, with $a = 0$ otherwise. Our convolution model then becomes

$$b(s, t) = \int_{-3/2}^{5/2} \int_{-3/2}^{5/2} a(s - s', t - t') x(s', t') \, ds' \, dt', \quad -1/2 \leq s, t \leq 3/2. \quad (3.34)$$

Now we proceed precisely as in the previous subsection, discretizing the integral on the analogous computational grid on $[-3/2, 5/2]$, and imposing a periodic boundary condition on the extended domain $[-3/2, 5/2]$, to obtain a linear system $\tilde{b} = \tilde{A}x$, where $b$ and $x$ are $(2n)^2 \times 1$ vectors, and $A$ is an $(2n)^2 \times (2n)^2$ BCCB matrix that is diagonalizable by the 2D-DFT. If we define $\tilde{B}$ and $X$ to be the $2n \times 2n$ arrays...
satisfying $\mathbf{b} = \text{vec}(\mathbf{B})$ and $\mathbf{x} = \text{vec}(\mathbf{X})$, then this linear system can be equivalently expressed
\[
\hat{\mathbf{B}} = \text{IDFT}(\mathbf{\tilde{a}}_s \odot \text{DFT}(\mathbf{X})),
\]
where $\mathbf{\tilde{a}}_s = 2n\text{DFT}(\text{fftshift}(\mathbf{a}))$, with $\mathbf{a}$ the $2n \times 2n$ array containing the kernel measurements, ordered analogous to (3.21).

To relate the model (3.35) to the actual observations $\mathbf{B}$, on $[0, 1] \times [0, 1]$, we define $\mathbf{D}$ to be the $n^2 \times (2n)^2$ diagonal matrix satisfying $\mathbf{D} \mathbf{b} = \mathbf{b}$. Then our model, in matrix-vector form, and including observation noise, is given by
\[
\mathbf{b} = \mathbf{A} \mathbf{x} + \mathbf{\epsilon}, \quad \mathbf{A} \overset{\text{def}}{=} \mathbf{D} \mathbf{A}.
\]
where as before $\mathbf{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$. Due to the properties of the matrix $\mathbf{\tilde{A}}$, the problem (3.36) will be ill-posed, and hence, regularization is needed. However, in this case the decomposition of the matrix $\mathbf{A}$ is infeasible, and hence we will use an iterative method to solve the Tikhonov pseudo-normal equations $(\mathbf{A}^T \mathbf{A} + \alpha \mathbf{I}) \mathbf{x} = \mathbf{A}^T \mathbf{b}$.

Before continuing, we note that an alternative way of dealing with the boundary artifact problem is to assume a Neumann boundary condition. The Neumann boundary condition corresponds to the assumption that the unknown $\mathbf{x}$ is extended outside of $[0, 1] \times [0, 1]$ via a reflection about the 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 two dimensional DFT; see [29] for details on this approach.

### 3.2 Two-dimensional computed tomography

In this section, we consider the test case 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 2, in the sense that the forward mathematical model involves the computation of a collection of line integrals of an object with unknown density $\mathbf{x}$.

In CT, the collection of integrals is along all lines in the plane intersecting $\mathbf{x}$. 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 $as + bt = c$, which can be equivalently expressed
\[
\left\{(s, t) \in \mathbb{R}^2 \left\mid \frac{a}{\sqrt{a^2 + b^2}} s + \frac{b}{\sqrt{a^2 + b^2}} t = \frac{c}{\sqrt{a^2 + b^2}} \right\} \right.
\]
Next, noting that there exists $\theta$ such that $(\cos \theta, \sin \theta) = \left( \frac{a}{\sqrt{a^2 + b^2}}, \frac{b}{\sqrt{a^2 + b^2}} \right)$, if $z = c/\sqrt{a^2 + b^2}$ and $\mathbf{\omega}_\theta \overset{\text{def}}{=} [\cos \theta, \sin \theta]^T$, then the collection of all lines in the plane can be expressed
\[
\ell_{\theta, z} \overset{\text{def}}{=} \left\{(s, t) \in \mathbb{R}^2 \left\mid \mathbf{\omega}_\theta^T \begin{bmatrix} s \\ t \end{bmatrix} = z \right\} \right.,
\]
(3.37)
where $z \in \mathbb{R}$ and $\theta \in [-\pi/2, \pi/2]$, since $\omega_{\theta + \pi} = -\omega_\theta$. Note that $\omega_\theta$ 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 $\omega_\theta$) vector $\omega_\theta^\perp = (-\sin \theta, \cos \theta)$. Then
\[
\ell_{\theta, z} = \{z\omega_\theta + \xi \omega_\theta^\perp | \xi \in \mathbb{R}\}.
\]

We can now define the Radon transform of $x$. For an X-ray traveling along $\ell_{\theta, z}$ in the positive direction with source intensity $I_{\theta, z}(\xi_0)$, we assume that X-ray scatter is negligible compared to absorption. Moreover, we assume that the decrease in the intensity $I_{\theta, z}(\xi)$, due to absorption, along a line segment $d\xi$ of $\ell_{\theta, z}$ is modeled by
\[
dI_{\theta, z}(\xi) = -x(z\omega_\theta + \xi \omega_\theta^\perp)I_{\theta, z}(\xi)d\xi.
\]
Then if $I_{\theta, z}(\xi_{\text{end}})$ is the intensity at the receiver after the X-ray has passed through the object, we can solve the above ordinary differential equation, to obtain
\[
-\ln \left( \frac{I_{\theta, z}(\xi_{\text{end}})}{I_{\theta, z}(\xi_0)} \right) = -\int_{\xi_0}^{\xi_{\text{end}}} \frac{dI_{\theta, z}(\xi)}{I_{\theta, z}(\xi)}d\xi = \int_{\xi_0}^{\xi_{\text{end}}} x(z\omega_\theta + \xi \omega_\theta^\perp)d\xi.
\]
Finally, defining $b(\theta, z) = -\ln \left( \frac{I_{\theta, z}(\xi_{\text{end}})}{I_{\theta, z}(\xi_0)} \right)$, and assuming that the source and receiver are located outside of $[-1,1] \times [-1,1]$, where $x(s,t) = 0$, we obtain the formula for the Radon transform of $x$:
\[
b(\theta, z) = \int_{-\infty}^{\infty} x(z\omega_\theta + \xi \omega_\theta^\perp)d\xi. \tag{3.38}
\]

The inverse problem of computed tomography is to reconstruct $x$ from its Radon transform $b$ [15].

To discretize (3.38), we assume that the support of $x$ is contained 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
\[
(s_i, t_j) = ((i-1)/n, (j-1)/n), \quad i, j = -n+1, \ldots, n.
\]

We denote $x_{ij} = x(s_i, t_j)$ and assume that $x$ is piecewise constant within each pixel; specifically
\[
x(s,t) = x_{ij}, \quad \text{for } (s,t) \in P_{ij},
\]
where
\[
P_{ij} = \{(s,t) \in \mathbb{R}^2 | (i-1)/n \leq s \leq i/n, \text{ and } (j-1)/n \leq t \leq j/n\}
\]
and $-n+1 \leq i, j \leq n$.

Next, we discretize the $z$ and $\theta$ variables, also uniformly. First, $\theta \in [-\pi/2, \pi/2)$, and hence,
\[
\theta_k = -\pi/2 + \frac{(k-1)}{m_2}\pi/2, \quad l = 1, \ldots, m_1,
\]
and since $x$ is contained in the unit circle, we assume $-1 \leq z \leq \text{nd}l$, and hence,
\[
z_l = -1 + 2\left(\frac{l-1}{m_2}\right), \quad l = 1, \ldots, m_2.
\]
3.2. Two-dimensional computed tomography

Figure 3.4. The line $\ell_{z_k,\theta_l}$ is $z_k$ units from $(0,0)$ in the direction $\omega_{\theta_l} = (\cos \theta_l, \sin \theta_l)$. Let $\Delta \ell^i_{kl}$ be the intersection length of $\ell_{z_k,\theta_l}$ with $P_{ij}$ and $x_{ij}$ be the value of $x$ at the midpoint of $P_{ij}$. Then if $X$ is the $n \times n$ array with $ij$th entry $x_{ij}$, the discrete Radon transform of $X$ is defined by $b_{kl} = \sum_{ij=1}^{n} \Delta \ell^i_{kl} x_{ij}$.

Then finally, if we define $b_{kl} = b(\theta_k, z_l)$, for $k = 1, \ldots, m_1$ and $l = 1, \ldots, m_2$, from (3.38) we obtain

$$b_{kl} = \sum_{i,j=1}^{n} \Delta \ell^i_{kl} x_{ij}, \quad k = 1, \ldots, m_1, \quad l = 1, \ldots, m_2, \quad (3.39)$$

where $\Delta \ell^i_{kl}$ is the intersection length of line $\ell_{\theta_k, z_l}$ with pixel $P_{ij}$. This is illustrated in Figure 3.4.

The system of equations defined by (3.39) characterizes the discrete Radon transform. Letting $B$ be the $m_1 \times m_2$ array with $kl$th element $b_{kl}$, $b = \text{vec}(B)$, and $x = \text{vec}(X)$, we obtain the $m_1 m_2 \times n^2$ system of equations $b = Ax$ from (3.39), where $A$ contains the intersection lengths $\Delta \ell^i_{kl}$ in the appropriate locations. Note that the $k$th row of $B$ corresponds to the collection of line integrals for all lines making an angle $\theta_k$ with the $s$ axis; thus this arrangement is known as parallel beam X-ray tomography. For numerical tests, we generate a $100 \times 100$ synthetic true object $X$, called the Shepp-Logan phantom, and plot it on the upper-left in Figure 3.5. We construct $A$ using $\text{Xraymat.m}$ with $m_1 = m_2 = 100$, and then generate the blurred noisy data $b$ (called a sinogram in CT problems) plotted on the upper-right in Figure 3.5. Independent and identically distributed white noise was added to $B$ so that the SNR, $\|Ax\|/\sqrt{m_1 m_2 \sigma^2}$, is 50.

Due to the properties of the matrix $A$, the CT inverse problem is ill-posed, and hence, regularization is needed to obtain stable solutions of the inverse problem. However, as in the data drive boundary condition case, we cannot solve the Tikhonov pseudo-normal equations, $(A^T A + \alpha I)x = A^T b$, directly.
3.3 The Preconditioned Conjugate Gradient Iteration

In the last two cases, the Tikhonov pseudo-normal equations, \((A^T A + \alpha I) x = A^T b\), could not be solved directly. In this section, we present PCG, which is a very efficient method for solving linear systems of the form

\[
Bx = c,
\]

where \(B \in \mathbb{R}^{N \times N}\) is symmetric positive definite (SPD). Note that the Tikhonov pseudo-normal equations are of this form with \(B = A^T A + \alpha I\) and \(c = A^T b\).

Conjugate gradients (CG) is a standard iterative method that appears in most undergraduate numerical analysis text books. For a thorough introduction to the CG algorithm and its theory, see [39]. Some highlights about CG convergence are that in exact arithmetic it converges to the solution \(x = B^{-1} c\) in at most \(r\) iterations, where \(r\) is the number of distinct eigenvalues of \(B\). Moreover, it can be shown that if the eigenvalues of \(B\) have \(m\) distinct clusters, then CG yields a good approximation of the solution in \(m\) iterations.

PCG iterations apply CG to a transformed system with better eigenvalue clustering than the original system \(Bx = c\), yielding faster convergence. Specifically, if we let \(\tilde{x} = C x\), where \(M = C^T C\) is called the preconditioner, then \(Bx = c\) can be equivalently expressed as

\[
C^{-T} B C^{-1} \tilde{x} = C^{-T} c.
\]

Applying CG to this transformed system and then changing variables back again via \(x = C^{-1} \tilde{x}\) yields the PCG algorithm, which is presented next.

**Algorithm 3.1.** PCG method for solving \(Bx = c\), where \(B\) is SPD.

Given \(x_0, B, c,\) and SPD preconditioner \(M\), let \(g_0 = Bx_0 - c, z_0 = M^{-1} g_0, p_0 = -z_0, \delta_0 = g_0^T z_0\) and \(k = 1\). Specify some stopping tolerance \(\epsilon\).

1. \(h_{k-1} = B p_{k-1}\);
3.3. The Preconditioned Conjugate Gradient Iteration

2. \( \tau_{k-1} = \delta_{k-1} / p_{k-1}^T h_{k-1} \);
3. \( x_k = x_{k-1} + \tau_{k-1} p_{k-1} \);
4. \( g_k = g_{k-1} + \tau_{k-1} h_{k-1} \);
5. \( z_k = M^{-1} g_k \);
6. \( \delta_k = g_k^T z_k \);
7. \( \beta_k = \delta_k / \delta_{k-1} \);
8. \( p_k = -z_k + \beta_k p_{k-1} \);
9. Quit if \( ||g_k|| < \epsilon \), else set \( k = k + 1 \) and go to step 1.

PCG is a powerful and broadly useful algorithm for solving large-scale SPD systems, and hence it has been studied extensively in the literature; see [22] and the references therein. Note that \( C \) does not appear in the PCG iteration, only the preconditioner \( M \). Moreover, if \( M = I \), PCG reduces to CG. The preconditioner \( M \), in general, should be an approximation of \( B \) that can be efficiently inverted, if one exists. For example, in the data driven boundary condition case, where \( B = A^T D^T D A + \alpha I \), we use as a preconditioner \( M = A^T \tilde{A} + \alpha I \), which is diagonalizable, and hence easily invertible, by the DFT.

3.3.1 Regularization Parameter Selection

Up to this point, the regularization parameter selection methods that we’ve presented have all depended up either an SVD or eigenvalue decomposition of the matrix \( A \). In the case when such a decomposition is not available, and we use, instead, an iterative method such as PCG to solve \( (A^T A + \alpha I)x = A^T b \), the regularization parameter selection methods presented in Chapter 2 must be implemented differently.

For UPRE, recall that we chose \( \alpha \) that minimizes

\[
U(\alpha) = ||Ax_\alpha - b||^2 + 2\sigma^2 \text{tr}(AA_\alpha) - m\sigma^2,
\]

where \( x_\alpha = (A^T A + \alpha I)^{-1} A^T b \). To evaluation \( U(\alpha) \), we first use PCG to compute \( x_\alpha \), and for \( \text{tr}(AA_\alpha) \), we use randomized trace estimation, which is motivated from the fact that if \( v \) is a white noise random vector and \( C \) is a symmetric matrix, then \( E(v^T Cv) = \text{tr}(C) \) [49]. Thus if we choose \( v \) to be a realization of a white noise vector,

\[
\text{tr}(AA_\alpha) \approx v^T AA_\alpha v. \tag{3.40}
\]

The choice of \( v \) that minimizes the variance in this estimator is the one for which the components of \( v \) are independent and take on the values of 1 and -1 with equal probability (see [49] and the references therein). Note that to compute \( A_\alpha v = (A^* A + \alpha I)^{-1} A^* v \) requires another implementation of PCG. Thus the evaluation of \( U(\alpha) \) requires two PCG runs.
The implementation of GCV in this setting is very similar to that of UPRE, since evaluating the GCV function

\[ G(\alpha) = \frac{\|Ax_\alpha - b\|^2}{(m - \text{tr}(AA_\alpha))^2} \]

requires a PCG run for both the computation of \(x_\alpha\), as well as for the approximation of \(\text{tr}(AA_\alpha)\). The implementation of DP, on the other hand, only requires one PCG run per evaluation of

\[ D(\alpha) = \|Ax_\alpha - b\|^2 - m\sigma^2 \]

for the computation of \(x_\alpha\).

For the L-curve method, two PCG runs are also required; in particular, in place of an SVD-type definition for \(s'(\alpha)\) like (2.34), one can use

\[ s'(\alpha) = \frac{2}{\alpha}x_\alpha^Tz_\alpha, \quad (A^*A + \alpha I)z_\alpha = A^*(Ax_\alpha - b), \quad (3.41) \]

where \(z_\alpha\) is computed using PCG. The proof that (3.41) and (2.34) are equivalent is left as an exercise.

### 3.3.2 Computing \(x_\alpha\) for the data driven boundary conditions and computed tomography test cases

First, we implement PCG for solving \((A^TA + \alpha I)x = A^Tb\) in the data driven boundary condition case. We consider the example described in the first paragraph of Section 3.1.3, with true image and blurred noisy data shown in Figure 3.3. We chose the regularization parameter using the implementation of GCV mentioned in the previous subsection, yielding \(\alpha = 4.03 \times 10^{-5}\), and plot the corresponding Tikhonov reconstruction on the upper-left 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 (\(M = I\)) and PCG with \(M = A^TA + \alpha I\) on the upper-right in Figure 3.6, and note that PCG converges reduces the residual norm by six order of magnitude in roughly half as many iterations at CG. For details of the implementation, see `Deblur2dDataDriven.m`.

Next, we implement CG for solving \((A^TA + \alpha I)x = A^Tb\) in the computed tomography case, noting that even without preconditioning, CG is fast to converge. We again use GCV to compute the regularization parameter, yielding \(\alpha = 4.53 \times 10^{-4}\). The corresponding Tikhonov solution is plotted on the lower-left in Figure 3.6. We also plot the convergence history of CG on the lower-right in Figure 3.6 and note that even without preconditioning, it reduces the residual norm by six orders of magnitude more rapidly than PCG did in the data drive boundary condition case. For details on the implementation, see `TomographyCG.m`.

### Exercises

3.1. *Midpoint quadrature for two-dimensional convolution.*
a. Verify that mid-point quadrature applied in the 2D deconvolution case presented at the beginning of the chapter yields (3.3).

b. Show that when a zero boundary condition is assumed, (3.3) takes the form

\[ b_{lm} = h^2 \sum_{i=1}^{n} \sum_{j=1}^{n} a_{l-i, m-j} x_{ij}, \quad l, m = 1, \ldots, n. \]

c. Verify that in the case of zero boundary conditions and a separable kernel, with the numerical grid defined in Section 3.1.1, (3.3) implies (3.5). Then fill in the details of the arguments showing that (3.5) implies (3.7).
d. Verify that in the case periodic boundary conditions, with the numerical grid defined in Section 3.1.2, mid-point quadrature yields (3.22). Then fill in the details of the arguments showing that (3.22) implies (3.24).

3.2. Separable blur and the Kronecker product
a. Prove (3.8), i.e., that vec($A_1X A_2^T$) = ($A_2 \otimes A_1)x$.

b. Use $A = A_2 \otimes A_1$ and the Kronecker product properties (3.9)-(3.11) to prove (3.12) and (3.13).

c. Show that (3.13) and (3.14) are equivalent.

d. Prove (3.15), then use it to prove (3.16) and (3.17).

3.3. Regularization parameter selection methods for the separable blur case.
a. Verify that (3.18) follows from (2.23) and that (3.19) follows from (2.31).

b. Derive the formula for GCV for the separable blur case from (2.27). Add lines of code to Deblur2dSeparable.m so that it implements GCV. Report the value(s) of $\alpha$ and plot the corresponding Tikhonov regularized solution(s).

c. Derive the formula for the L-curve method analogous to (2.33), (2.34). Add lines of code to Deblur2dSeparable.m so that it implements the L-curve method. Report the value(s) of $\alpha$ and plot the corresponding Tikhonov regularized solution(s).

3.4. Periodic boundary conditions and DFT
a. Verify that $A$ defined by (3.25) can be expressed (3.26), and then verify the string of equalities found in (3.27).

b. Verify that (3.27) implies (3.28) and (3.29).

c. Verify that the Tikhonov solution $x_\alpha = (A^* A + \alpha I)^{-1} A^* b$ is given in array form by (3.31).

3.5. Regularization parameter selection methods for the periodic boundary condition case.

a. Verify that (3.32) follows from (2.27) and that (3.33) follows from (2.34).

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

c. Derive the formula for the DP method analogous to (2.31). Add lines of code to Deblur2dPeriodic.m so that it implements the DP regularization parameter selection method.

Report the value(s) of $\alpha$ and plot the corresponding Tikhonov regularized solution(s).

3.6. 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 a 128×128 pixel subimage 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.

3.7. *The TSVD for two-dimensional deblurring.* Note that the singular values aren’t ordered in either of the below test cases, so you will not be able to directly apply the TSVD from Chapter 2.

a. Modify `Deblur2dSeparable.m` so that it implements TSVD regularization.

b. Modify `Deblur2dPeriodic.m` so that it implements TSVD regularization.

3.8. *Ill-posedness for two-dimensional deblurring.*

a. Verify that the inequality (1.28), which we used to define ill-posedness in Chapter 1, holds for the example in Section 3.1.1, which can be found in `Deblur2dSeparable.m`.

b. Verify that the inequality (1.28) holds for the example in Section 3.1.1, which can be found in `Deblur2dPeriodic.m`.

3.9. *Testing the accuracy of the randomized trace estimate.*

a. Show that in the periodic boundary condition case

\[
\text{tr}(AA_\alpha) = \sum_{i,j=1}^n |\hat{\mathbf{a}}_i|_2^2/((|\hat{\mathbf{a}}_i|_2^2 + \alpha),
\]

where \( A_\alpha = (A^*A + \alpha I)^{-1}A^* \). Note that the trace of a matrix is the sum of its eigenvalues.

b. Let \( n = 128 \) and compute a single realization \( \mathbf{v} \in \mathbb{R}^{n \times n} \) with independent entries equal to +1 or -1 with equal probability. This can be done in MATLAB using the command \( \mathbf{v} = 2*(\text{rand}(n,n)>0.5)-1 \). Next, create a grid for the Tikhonov regularization parameter \( \alpha \) using `logspace` and plot the trace estimate \( \bar{t}(\alpha) = \mathbf{v}^TAA_\alpha\mathbf{v} \) together with the true trace \( t(\alpha) = \text{tr}(AA_\alpha) \), defined in part a. Note that multiplication by \( A_\alpha \) is defined in equation (3.31). Is \( \bar{t}(\alpha) \) a good approximation of \( t(\alpha) \)? Use your graph to support your answer.


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. Use the SVD to show that (3.41) and (2.34) are equivalent definitions for \( s'(\alpha) \), which is needed for evaluating the curvature (2.33) for the L-curve regularization parameter selection method. Then, using (3.41) modify `Deblur2dDataDriven.m` so that L-curve regularization parameter selection is implemented.
Chapter 3. Two-Dimensional Test Cases

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

3.11. Regularization parameter selection methods for the computed tomography case.

a. Modify TomographyCG.m so that it implements DP regularization parameter selection.

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

c. Use the SVD to show that (3.41) and (2.34) are equivalent definitions for \(s'(\alpha)\), which is needed for evaluating the curvature (2.33) for the L-curve regularization parameter selection method. Then, using (3.41) modify Deblur2dDataDriven.m so that L-curve regularization parameter selection is implemented.

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

3.12. Landweber iteration

a. Modify Deblur2DataDriven.m so that the truncated Landweber iteration, introduced in Chapter 2, is used for minimizing \(\|Ax - b\|^2\). Use the DP stopping rule, i.e., stop iterations at the first \(k\) for which \(\|Ax_k - b\|^2/(n^2\sigma^2)\leq 1\).

b. Modify TomographyCG.m so that the truncated Landweber iteration is used for minimizing \(\|Ax - b\|^2\). Use the DP stopping rule, i.e., stop iterations at the first \(k\) for which \(\|Ax_k - b\|^2/(n^2\sigma^2)\leq 1\).


a. Express \(Ax = b\) in the modified form

\[
a_i^T x = b_i, \quad i = 1, \ldots, m_1 m_2,
\]

where \(a_i\) is the \(i^{th}\) row of \(A\). The idea behind a single sweep of standard Kaczmarz’s method is to start with an approximate solution \(x_0\) and project it onto hyper-plane \(a_i^T x = b_i\) to obtain \(x_1\), then project \(x_1\) onto hyper-plane \(a_2^T x = b_2\) to obtain \(x_2\), etc., until \(x_{m_1 m_2}\) is computed. Use projections to show that this implies the following iteration: choose \(x_0\), then

\[
x_i = x_{i-1} + \frac{b_i - a_i^T x_{i-1}}{\|a_i\|^2} a_i, \quad \text{for } i = 1, \ldots, m_1 m_2. \tag{3.42}
\]

b. Once \(i = m_1 m_2\) in (3.42), the iteration is then continued by setting \(x_0 = x_{m_1 m_2}\) and repeating (3.42). It can be shown that the iterates converge to the minimum norm least squares solution of \(Ax = b\). If \(x^k\) is the reconstruction obtained after \(k\) full sweeps of Kaczmarz’s iterations, one can implement the DP stopping rule by stopping iterations once \(\|Ax^k - b\|^2/(n^2\sigma^2)\leq 1\). Modify TomographyCG.m so that it implements Kaczmarz’s method with DP stopping rule.
c. Derive Kaczmarz’s method for the block form of $Ax = b$ given by

$$
\begin{bmatrix}
A_1^T \\
A_2^T \\
\vdots \\
A_k^T \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_k \\
\end{bmatrix} =
\begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_k \\
\end{bmatrix},
$$

where $A_i$ contains at least one row of $A$, and $b_i$ contains at least one element of $b$, for $i = 1, \ldots, k$. Note that if $k = m_1 m_2$, we return to standard Kaczmarz. Modify your code from part b so that block Kaczmarz’s method is implemented, where $A_i$ contains 10, and then 100, rows for each $i$. Does blocking reduce the CPU time needed to satisfy the DP stopping rule?

d. Kaczmarz method (3.42) can be expressed $x_i = P_{a_i} x_{i-1}$, where $P_{a_i} x = x + (b_i - a_i a_i^T x)/\|a_i\|^2$, for $i = 1, \ldots, m_1 m_2$. Use the splitting $x = \frac{1}{2} x + (1 - \frac{1}{2}) x$ to derive the relaxed version of Kaczmarz method, i.e., begin with $\frac{1}{2} x_i + (1 - \frac{1}{2}) x_{i-1} = P_{a_i} x_{i-1}$ and arrive at an iteration for $x$ that contains the parameter $\omega$. Modify KaczmarzMethod.m so that it implements the relaxed version of Kaczmarz’s method, and determine if relaxation reduces the CPU time needed to satisfy the DP stopping rule. In [31], the values $\omega \in [0, 2]$ are suggested.

This problem requires the codes in the directory Chapter3/Dirichlet, the book [49], and is quite involved, so it would make for a good project.

a. When a zero boundary condition is assumed and the kernel is non-separable, two dimensional convolution can be written as $b = Ax$, where $A$ is block toeplitz with toeplitz blocks (BTTB). In [49], this system is derived. Read and understand the derivation and write a brief summary.

b. The matrix $A$ is $n^2 \times n^2$ BTTB, and hence, can be embedded within a $(2n)^2 \times (2n)^2$ BCCB matrix for fast matrix-vector multiplication. Provide a brief description of the embedding (again see [49]) and verify that the function Amult_Dirichlet.m performs multiplication by $A$ using this embedding.

c. The file Deblur2dDirichlet.m generates blurred noisy data using the convolution model with zero boundary conditions. Implement CG for solving $(A^T A + \alpha I)x = A^T b$ using CG.m. Note that Bmult_Dirichlet.m performs multiplication by $A^T A + \alpha I$. Enter $\alpha$ manually, and provide enough output to show that you’ve got things working, including a de-blurred image for a ‘good’ value of $\alpha$.

d. Implement GCV for choosing $\alpha$. Note that you will need to use randomized trace estimation. Alternatively, you could implement UPRE, which also requires randomized trace estimation, DP, or L-curve. Report the value of $\alpha$ and plot the corresponding reconstruction.
Chapter 3. Two-Dimensional Test Cases

e. Implement preconditioning for the CG iterations. Use \( \hat{A}^2 \hat{A} + \alpha I \), where \( \hat{A} \) is the \( n^2 \times n^2 \) BCCB matrix that would arise if, instead, a periodic boundary condition was assumed on \( x \). Plot the convergence history for CG versus PCG as in (3.6). You should notice that PCG has much better convergence properties.
Chapter 4
Bayes’ Law, Markov
Random Field Priors, and
MAP Estimation

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 $v_i$ of $A$ with very small singular values. This model-based observation motivated Chapter 2, where we discussed the notion of ‘filtering’ out the problematic singular vectors from the model. The resulting regularized solutions have smaller variance, but with the negative side effect of bias in the regularized solution.

In this chapter, we provide an alternative motivation for regularization via Bayes’ Law. Bayes’ Law requires the assumption of a probability model for the unknown $x$, whereas in Chapters 1 and 2, $x$ was assumed to be a fixed vector. The probability model for $x$ is known as the prior probability density function and should be constructed from prior knowledge about $x$. An effective and intuitive way to build the prior for the problems we consider is using Markov random fields. We focus much of our attention in this chapter on the development of various Markov random field priors.

Finally, with the prior in hand, the maximum a posteriori (MAP) estimator can be computed by maximizing the posterior density function, which is the product of the likelihood (which was defined in Chapter 1) and the prior. This involves the solution of either a linear system of equations or of an optimization problem.

4.1 Bayes Law and Regularization

In the Bayesian approach to regularization, we model the unknown $x$ as a random vector with a probability density function $p(x|\delta)$ known as the prior probability density, or simply the prior, with $\delta$ a positive scaling parameter. The prior encodes both prior knowledge about the properties of $x$, as well as uncertainty about $x$, making the choice of $p(x|\delta)$ a subjective enterprise. Assuming the data model

$$b = Ax + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \Lambda^{-1}I_M)$$

(4.1)
where \( A \in \mathbb{R}^{M \times N} \), the probability density function for \( b \) is given by
\[
p(b|\mathbf{x}, \lambda) = \left( \frac{\lambda}{2\pi} \right)^{M/2} \exp \left( -\frac{\lambda}{2} \| \mathbf{Ax} - b \|^2 \right).
\]

(4.2)

Once a prior is chosen, Bayes’ law defines the posterior density function
\[
p(\mathbf{x}|b, \lambda, \delta) \propto p(b|\mathbf{x}, \lambda)p(\mathbf{x}|\delta),
\]
where ‘\( \propto \)’ denote proportionality. The posterior density function defines a probability density for the unknown \( \mathbf{x} \), and hence we can compute point estimates of this random vector, such as the mean, and also quantify uncertainty by computing measures of variability of \( \mathbf{x} \), such as the variance. Perhaps the most common point estimator for the posterior density function is its maximizer, called the maximum a posteriori (MAP) estimator, and which we denote by \( \mathbf{x}_{\text{MAP}} \).

For our first example of a prior, recall that in the one-dimensional image deblurring test case, the least squares solution \( \mathbf{x}_{\text{LS}} \) had very large magnitude; see Figure 1.3. We can incorporate our prior knowledge that the magnitude of \( \mathbf{x} \) is not-too-large by assuming \( \mathbf{x} \sim N(0, \delta^{-1}I) \), for \( \delta > 0 \). This yields the prior
\[
p(\mathbf{x}|\delta) = \left( \frac{\delta}{2\pi} \right)^{N/2} \exp \left( -\frac{\delta}{2} \| \mathbf{x} \|^2 \right).
\]

(4.4)

The MAP estimator is then the minimizer of \( -\ln p(\mathbf{x}|b, \lambda, \delta) \), i.e.,
\[
\mathbf{x}_{\text{MAP}} = \arg \min_{\mathbf{x}} \left\{ \frac{\lambda}{2} \| \mathbf{Ax} - b \|^2 + \frac{\delta}{2} \| \mathbf{x} \|^2 \right\},
\]
which is the Tikhonov regularized solution, defined by (2.8), with \( \alpha = \delta/\lambda \). Thus an estimate of \( \alpha \) 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 (4.4) corresponds to the assumption that \( x_i \overset{iid}{\sim} N(0, \delta^{-1}) \), where \( iid \) denotes independent and identically distributed, 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 often be near one another 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 Choosing \( p(\mathbf{x}|\delta) \): Gaussian Markov random fields

Our discussion in this section follows that in [6], which in turn follows [42]. In the pioneering work of Besag [9], an approach known as conditional autoregression was introduced for defining statistical models of a spatially distributed parameter \( \mathbf{x} \in \mathbb{R}^N \). 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 first consider the general assumption
\[
x_i|\mathbf{x}_{-i} \sim N \left( \sum_{j \neq i} \beta_{ij} x_j, \kappa_i^{-1} \right),
\]

(4.6)
4.2. Choosing \( p(x|\delta) \): Gaussian Markov random fields

where it is assumed that \( \beta_{ij}\kappa_i = \beta_{ji}\kappa_j \) for all \( i \) and \( j \). Then, if we define

\[
[Q]_{ij} = \begin{cases} 
\kappa_i, & i = j, \\
-\kappa_i\beta_{ij}, & i \neq j,
\end{cases}
\]  \hspace{1cm} (4.7)

for \( i,j = 1,\ldots,N \), and we make the additional assumption that \( Q \) is positive definite, by [42, Theorem 2.6] the random vector \( x \) defined by (4.6) is Gaussian with zero mean and precision (inverse-covariance) matrix \( Q \), i.e.,

\[
p(x) = \sqrt{|Q|/(2\pi)^N} \exp \left( -\frac{1}{2}x^T Qx \right),
\]  \hspace{1cm} (4.8)

where \( |\cdot| \) denotes determinant. Due to the Gaussianity assumption in (4.6), \( x \) is known as a Gaussian Markov random field (GMRF). We will present some concrete, and more easily understandable, GMRFs in a moment, but first we establish more notation.

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 (4.6-4.8); note that \( [Q]_{ij} \neq 0 \) if and only if \( j \in \partial_i \). Thus the sparsity structure of \( Q \) determines the neighborhood system, and vice versa. Moreover, if \( x_{\partial_i} = \{ x_j \mid j \in \partial_i \} \), we have \( p(x_i|x_{-i}) = p(x_i|x_{\partial_i}) \) for all \( i \), and hence it suffices to define the ‘neighborhood conditionals’ (see [42, Theorem 2.4])

\[
x_i|x_{\partial_i} \sim N \left( \sum_{j \in \partial_i} \beta_{ij} x_j, \kappa_i^{-1} \right)
\]  \hspace{1cm} (4.9)

to obtain (4.7-4.8).

As an example, assume \( x \) is a discretization of a function \( x \) either on \([0,1]\) or \([0,1] \times [0,1]\). Moreover, we assume a zero boundary condition on \( x \), so that in the one-dimensional case \( x = (x_1,\ldots,x_n) \) with \( x_i = x((i-1/2)h) \), for \( h = 1/n \) and \( i = 1,\ldots,n \). Whereas in two-dimensions, we define \( x = \text{vec}(X) \), where \( X \) is an \( n \times n \) array, with \( [X]_{ij} = x_{ij} = x((i-1/2)h,(j-1/2)h) \), for \( i,j = 1,\ldots,n \). Now, we assume, intuitively, that \( \kappa_i^{-1} \) (the variance of \( x_i|x_{\partial_i} \)) is inversely proportional to \( n_i = |\partial_i| \), so that \( \kappa_i = \delta n_i \), where \( \delta > 0 \). Moreover, we assume that \( \beta_{ij} = 1/n_i \) for \( j \in \partial_i \) and 0 otherwise. Then (4.9) reduces to

\[
x_i|x_{\partial_i} \sim N \left( \bar{x}_{\partial_i}, (\delta n_i)^{-1} \right), \quad i = 1,\ldots,n,
\]  \hspace{1cm} (4.10)

where \( \bar{x}_{\partial_i} = \frac{1}{n_i} \sum_{i \in \partial_i} x_i \), and from the discussion above, we see that we obtain a Gaussian prior of the form (4.8) with precision matrix

\[
[Q]_{ij} = \delta \begin{cases} 
n_i, & i = j, \\
-1, & j \in \partial_i, \\
0, & \text{otherwise},
\end{cases}
\]  \hspace{1cm} (4.11)
4.2.1 Smoothness priors as Gaussian Markov Random Fields

In one-dimension, we will use the simple first order neighborhood system \( \partial_i = \{i-1, i+1\} \), so that \( n_i = 2 \), for \( i = 1, \ldots, n \), and (4.10) takes the form

\[
x_i | x_{\partial_i} \sim \mathcal{N}\left(\frac{x_{i-1} + x_{i+1}}{2}, \delta^{-1}\right),
\]

for \( i = 1, \ldots, n \). Note that the neighborhoods \( \partial_1 \) and \( \partial_n \) require that we define the values of \( x_0 \) and \( x_{n+1} \), which will be done through the boundary conditions.

In two-dimensions, we will use the first order neighborhood system

\[
\partial_{ij} = \{(i-1,j), (i+1,j), (i,j-1), (i,j+1)\},
\]

so that \( n_{ij} = 4 \), for \( i,j = 1, \ldots, n \), and (4.10) can be written

\[
x_{ij} | x_{\partial_{ij}} \sim \mathcal{N}\left(\frac{x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1}}{4}, \delta^{-1}\right),
\]

for \( i,j = 1, \ldots, n \). Note that the neighborhoods \( \partial_{1,j} \), \( \partial_{n,j} \), \( \partial_{i,1} \), and \( \partial_{i,n} \) require that we define the values of \( x_{0,j} \), \( x_{n+1,j} \), \( x_{i,0} \) and \( x_{i,n+1} \), which will be done through the boundary conditions. In practice, as in Chapter 3, we will turn the \( n \times n \) array \( X \) into an \( n^2 \times 1 \) vector via \( x = \text{vec}(X) \), which will allow us to express (4.13) precisely as in (4.10).

Zero boundary conditions

In the one-dimensional case, a zero boundary condition implies \( x_0 = x_{n+1} = 0 \), and from (4.11) we obtain \( Q = \delta L_{1D} \) with

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

Note that \( Q = \delta L_{1D} \) is symmetric positive definite (SPD), as required, and moreover, that \( (1/h^2)L_{1D} \) is a finite difference discretization of (see Exercise 4.2):

\[
-\frac{d^2}{ds^2} x(s), \quad 0 \leq s \leq 1, \quad x(0) = x(1) = 0.
\]

On the left in Figure 4.1, inspired by [11, 13], we’ve plotted five samples from the GMRF \( x \sim \mathcal{N}(0, L_{1D}^{-1}) \). We will discuss the problem of sampling from a Gaussian in Chapter 5.

In the two-dimensional case, the zero boundary conditions implies \( x_{0,j} = x_{n+1,j} = x_{i,0} = x_{i,n+1} = 0 \), and from (4.11), we obtain \( Q = \delta L_{2D} \), where \( L_{2D} \) can be expressed (see Exercise 4.3)

\[
L_{2D} = I \otimes L_{1D} + L_{1D} \otimes I,
\]

for \( i \in \mathbb{Z} \).
4.2. Choosing $p(x|\delta)$: Gaussian Markov random fields

Figure 4.1. Realizations from standard one and two dimensional GMRFs. On the left are five sample from the one-dimensional GMRF $x \sim \mathcal{N}(0, L_{1D})$, with $L_{1D}$ defined by (4.14). On the right is one sample from the two-dimensional GMRF $x \sim \mathcal{N}(0, L_{2D})$, with $L_{2D}$ defined by (4.15), (4.14).

where $L_{1D}$ is defined in (4.14) and ‘⊗’ is the Kronecker product, as defined in Chapter 3. Note that $Q = \delta L_{2D}$ is SPD, as required, and that $(1/h^2)L_{2D}$ is a discretization of the negative Laplacian operator (see Exercise 4.3):

$$ \left( -\frac{\partial^2}{\partial s^2} - \frac{\partial^2}{\partial t^2} \right) x(s,t), \quad 0 \leq s, t \leq 1, \quad x(0, t) = x(1, t) = x(s, 0) = x(s, 1) = 0. $$

On the right in Figure 4.1, we’ve plotted one sample from $x \sim \mathcal{N}(0, L_{-1}^{-1})$.

In both of the one- and two-dimensional cases above, the prior (4.8) has the form

$$ p(x|\delta) \propto \delta^{N/2} \exp \left( -\frac{\delta}{2} x^T L x \right), \quad (4.16) $$

where $N = n$ and $L = L_{1D}$ in one-dimension; and $N = n^2$ and $L = L_{2D}$ in two-dimensions.

Periodic boundary conditions

In one-dimension, a periodic boundary condition implies $x_0 = x_n$ and $x_{n+1} = x_1$, so that $\partial_1 = \{n, 2\}$, and $\partial_n = \{n - 1, 1\}$. The precision matrix given by (4.11) then has the form $Q = \delta L_{1D}$, with

$$ L_{1D} = \begin{bmatrix}
2 & -1 & 0 & \cdots & -1 \\
-1 & 2 & -1 & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & -1 & 2 & -1 \\
-1 & \cdots & 0 & -1 & 2 \\
\end{bmatrix}_{n \times n}. \quad (4.17) $$
Note that \((1/h^2)L_{1D}\) is a standard finite difference discretization of the second derivative operator (see Exercise 4.2):

\[-\frac{d^2}{ds^2}x(s), \quad 0 \leq s \leq 1, \quad x(0) = x(1).\]

Moreover, since \(Q1 = 0\), the corresponding prior (4.8) is improper. Such GMRFs are considered in [42] and are called *intrinsic GMRFs* (IGMRFs). Moreover, when \(Q\) has only one zero eigenvalue with eigenvector 1, as is the case here, the resulting IGMRF is said to be first-order [42]. For such IGMRFs, the corresponding prior has the form

\[p(x|\delta) \propto \delta^{(N-1)/2} \exp \left( -\frac{1}{2} x^T Q x \right),\]

(4.18)

where \(x \in \mathbb{R}^N\) and \(N - 1\) is the rank of \(Q\), with \(N = n\) in the one-dimensional case.

In two-dimensions, the periodic boundary condition implies \(x_{0,j} = x_{n,j}, x_{n+1,j} = x_{1,j}, x_{i,0} = x_{i,n}, x_{i,n+1} = x_{i,1}\). Then \(Q = \delta L_{2D}\), where \(L_{2D}\) has the Kronecker product form (4.15) with \(L_{1D}\) given by (4.17). Using this Kronecker formula, it is straightforward to show that \(L_{2D}\) is a block circulant with circulant blocks (BCCB) matrix. Hence, it can be diagonalized by the 2D-DFT with \(n \times n\) eigenvalue array \(\hat{l}_s\) given in Exercise 5, so that

\[L_{2D} x = \text{vec}(\text{IDFT}(\hat{l}_s \odot \text{DFT}(X))).\]

(4.19)

The corresponding prior has the form (4.18) with \(N = n^2\). And finally, we note that in this case \((1/h^2)L_{2D}\) is the standard finite difference discretization of the negative Laplacian operator (see Exercise 4.3):

\[-\frac{\partial^2}{\partial s^2} - \frac{\partial^2}{\partial t^2} x(s,t), \quad 0 \leq s, t \leq 1, \quad x(0, t) = x(1, t), \quad x(s, 0) = x(s, 1).\]

The samples from both of these GMRFs are very similar in appearance to those in Figure 4.1, except that they satisfy periodic (rather than zero) boundary conditions. We will discuss the problem of sampling from Gaussian random vectors in Chapter 5.

**Independent and identically distributed increments**

Another standard example of an IGMRF results when the *increments* \(\Delta x_i = x_{i+1} - x_i\) are assumed to be of the form

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

(4.20)
4.2. Choosing \( p(x|\delta) \): Gaussian Markov random fields

Then the probability density function for \( x \) has the form

\[
p(x|\delta) \propto \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} \sum_{i=1}^{n-1} \Delta x_i^2 \right)
\]

\[
\propto \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 \right)
\]

\[
= \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} x^T (D^T D) x \right), \quad (4.21)
\]

where

\[
D = \begin{bmatrix}
-1 & 1 & 0 & \cdots & 0 \\
0 & -1 & 1 & \ddots & \\
\vdots & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & -1 & 1
\end{bmatrix}_{(n-1) \times n}.
\]

Note that (4.21) has the form (4.18) with \( N = n \) and \( Q = \delta D^T D \). It is easy to verify that

\[
D^T D = \begin{bmatrix}
1 & -1 & 0 & \cdots & 0 \\
-1 & 2 & -1 & \ddots & \\
\vdots & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & -1 & 2 & -1 \\
0 & \cdots & 0 & -1 & 1
\end{bmatrix}_{n \times n}.
\]

(4.23)

Finally, \( (1/h^2)D^T D \) is the standard finite difference discretization of the negative second derivative operator (see Exercise 4.3):

\[
-\frac{d^2}{ds^2} x(s), \quad 0 \leq s \leq 1, \quad x'(0) = x'(1) = 0.
\]

Zero or periodic boundary conditions can also be imposed by modifying the derivative matrix \( D \) accordingly (see Exercise 1). Finally, note that the increment model (4.20) corresponds to the autoregressive model (4.12) for \( i = 2, \ldots, n-1, \ x_1|x_{\delta_1} \sim \mathcal{N}(x_2, \delta^{-1}), \) and \( x_n|x_{\delta_n} \sim \mathcal{N}(x_{n-1}, \delta^{-1}) \).

In two-dimensions, we define the horizontal and vertical increments \( \Delta_x x_{ij} = x_{i+1,j} - x_{ij} \) and \( \Delta_t x_{ij} = x_{i,j+1} - x_{ij} \), respectively, and assume

\[
\Delta_x x_{ij} \overset{iid}{\sim} \mathcal{N}(0, \delta^{-1}), \quad i = 1, \ldots, n-1, \ j = 1, \ldots, n,
\]

\[
\Delta_t x_{ij} \overset{iid}{\sim} \mathcal{N}(0, \delta^{-1}), \quad i = 1, \ldots, n, \ j = 1, \ldots, n-1.
\]

(4.24)
Then the density function for \( x \) has the form

\[
p(x|\delta) \propto \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} \left( \sum_{i=1}^{n} \sum_{j=1}^{n-1} \Delta x_{ij}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n-1} \Delta e_{ij}^2 \right) \right)
\]

\[
\propto \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} \left( \sum_{i=1}^{n} \sum_{j=1}^{n-1} (x_{i+1,j} - x_{ij})^2 + \sum_{i=1}^{n} \sum_{j=1}^{n-1} (x_{i,j+1} - x_{ij})^2 \right) \right)
\]

\[
= \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} \left( \| (I \otimes D)x \|^2 + \| (D \otimes I)x \|^2 \right) \right)
\]

\[
= \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} x^T (D_s^T D_s + D_t^T D_t) x \right)
\]

(4.25)

where \( D_s = I \otimes D \) and \( D_t = D \otimes I \), with \( D \) given by (4.22), are the discrete derivatives with respect to \( s \) and \( t \), respectively. Note that (4.25) has the form (4.18) with \( N = n^2 \) and \( Q = \delta(D_s^T D_s + D_t^T D_t) \), which can be shown to have the Kronecker product form (4.15) for \( L_{1D} = D^T D \). Moreover, \((1/h^2)(D_s^T D_s + D_t^T D_t)\) is the standard finite difference discretization of the negative Laplacian operator

\[
\left( -\frac{\partial^2}{\partial s^2} - \frac{\partial^2}{\partial t^2} \right) x(s,t), \quad 0 \leq s, t \leq 1, \quad x(s,0) = x(s,1) = 0, \quad x(t,s) = x(t,1).
\]

Finally, the above shows that the increment model (4.41) corresponds to the autoregressive model (4.13) for \( i, j = 2, \ldots, n-1 \). For \( i = 1 \), the autoregressive model has the form \( x_{1j} | x_{s_{ij}} \sim N \left( (x_{1,j-1} + x_{1,j+1} + x_{2,j})/3, (3\delta)^{-1} \right), j = 2, \ldots, n; \)

\( x_{11} | x_{s_{11}} \sim N \left( (x_{1,2} + x_{2,1})/2, (2\delta)^{-1} \right) \); and \( x_{1n} | x_{s_{1n}} \sim N \left( (x_{1,n-1} + x_{2n})/2, (2\delta)^{-1} \right) \). The autoregressive models for \( i = n \) and \( j = 1 \), are left to the reader to derive. The samples from these two IGMRFs are also very similar in appearance to those in Figure 4.1, except that they satisfy the Neumann boundary condition. The problem of sampling from such Gaussian random vectors will be addressed in Chapter 5.

**Independent increments**

In many instances, it is desirable to allow for the increment variance to be larger in some locations than others. For example, if you know that your image \( x \) has a sharp intensity change at pixel \( i \), i.e., \( \Delta x_i \) is relatively large, you would want to allow for that in your increment model. In the one-dimensional case, this motivates changing (4.20) to

\[
\Delta x_i \sim N(0, (w_i \delta)^{-1}), \quad i = 1, \ldots, n-1.
\]

(4.26)
Then the density function for $x$ has the form
\[
p(x|\delta) \propto \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} \sum_{i=1}^{n-1} w_i(x_{i+1} - x_i)^2 \right) \\
\propto \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} \|A^{1/2}Dx\|^2 \right) \\
= \delta^{(n-1)/2} \exp \left( -\frac{\delta}{2} x^T(D^TAD)x \right),
\]
(4.27)

where $A = \text{diag}(w_1, \ldots, w_{n-1})$ and $D$ is as in (4.22). Note that this yields an implicit Neumann boundary condition. Zero or periodic boundary conditions can also be imposed by modifying the derivative matrix $D$ accordingly (see Exercise 1). Note also that (4.27) has the form (4.18) with $N = n$ and $L = D^TAD$, and that $(1/h^2)L$ is a finite difference discretization of the negative second derivative operator (see Exercise 4.2):

\[ -\frac{d}{ds} \left( w(s) \frac{d}{ds} x(s) \right), \quad 0 \leq s \leq 1, \quad x'(0) = x'(1) = 0. \]

Here $w_i = w((i-1/2)h)$. By changing $D$, as mentioned above, the boundary condition can be modified.

The prior (4.27) corresponds to the auto-regressive model (see [42, Chapter 3])
\[
x_i|x_{\theta_i} \sim \mathcal{N}\left( (x_{i-1} + x_{i+1})/2, (2\delta w_i)^{-1} \right),
\]
(4.28)

for $i = 2, \ldots, n - 1$; $x_1|x_{\theta_1} \sim \mathcal{N}(x_2, (\delta w_1)^{-1})$; and $x_n|x_{\theta_n} \sim \mathcal{N}(x_{n-1}, (\delta w_n)^{-1})$. We see that (4.28) allows for a spatially varying precision. On the left in Figure 4.2, again inspired by [12, 13], we plot five samples from such an IGMRF, 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 $n/2$ allows for jumps to form at pixel $n/2$, with relatively high probability.

In the two-dimensional case, we assume that
\[
\begin{align*}
\Delta_{x_{ij}} &\sim \mathcal{N}(0, w_{ij}^x\delta^{-1}), \quad i = 1, \ldots, n - 1, \quad j = 1, \ldots, n, \\
\Delta_{t_{ij}} &\sim \mathcal{N}(0, w_{ij}^t\delta^{-1}), \quad i = 1, \ldots, n, \quad j = 1, \ldots, n - 1.
\end{align*}
\]
(4.29)

Then the density function for $x$ has the form
\[
p(x|\delta) \propto \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} \left( \sum_{j=1}^{n-1} \sum_{i=1}^{n-1} w_{ij}^x \Delta_{x_{ij}}x_{ij}^2 + \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} w_{ij}^t \Delta_{t_{ij}}x_{ij}^2 \right) \right) \\
= \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} \left( \|A_{x}^{1/2}(I \otimes D)x\|^2 + \|A_{t}^{1/2}(D \otimes I)x\|^2 \right) \right) \\
= \delta^{(n^2-1)/2} \exp \left( -\frac{\delta}{2} x^T(D_{x}^T A_{x} A_{x} D_{x} + D_{t}^T A_{t} D_{t})x \right),
\]
(4.30)

where $A_{x} = \text{diag}(\text{vec}(\{w_{ij}^x\}_{i,j=1}^{n-1,n}))$, and $A_{t} = \text{diag}(\text{vec}(\{w_{ij}^t\}_{i,j=1}^{n,n-1}))$. As mentioned above, $D$ has an implicit Neumann boundary condition that can be changed.
Figure 4.2. Realizations from independent increment IGMRFs. On the left are five samples from the IGMRF (4.28) with $w_{n/2} = 0.0025$ and $w_i = 1$ for all $i \neq n/2$, $n = 128$. On the right is one sample from the IGMRF (4.31) with $w_{ij} = 0.0025$ along the boundary of a circle contained in $[0,1] \times [0,1]$ and $w_{ij} = 1$ otherwise, where $1 \leq i, j \leq 128$.

to reflect zero or periodic boundary conditions instead. Note also that (4.30) has the form (4.18) with $N = n^2$ and $Q = \delta(D_s^2 A_s + D_t^2 A_t)$, and that $(1/h^2)(D_s^2 A_s + D_t^2 A_t)$ is a finite difference discretization of the diffusion operator (see Exercise 4.3)

$$-\frac{\partial}{\partial s} \left( w_s(s,t) \frac{\partial}{\partial s} \right) - \frac{\partial}{\partial t} \left( w_t(s,t) \frac{\partial}{\partial t} \right) x(s,t), \quad 0 \leq s, t \leq 1,$$

with Neumann boundary conditions $\partial x/\partial s(0, t) = \partial x/\partial s(1, t) = 0$, and $\partial x/\partial t(s, 0) = \partial x/\partial t(s, 1)$.

In order to simplify our autoregressive model, we assume that $w_{ij} = w_s^{ij} = w_t^{ij}$, i.e., the horizontal and vertical weights are the same, then

$$x_{ij}|x_{\neq ij} \sim \mathcal{N}\left((x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1})/4, (4\delta w_{ij})^{-1}\right),$$

(4.31)

for $i, j = 2, \ldots, n-1$; and for $i, j = 1, n$, the autoregressive models, whose derivation we leave to the reader, are defined similarly to as in the two-dimensional independent increment case. On the right in Figure 4.2, we plot a single sample 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 sample 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. For problems devoted to sampling from such a prior, see Chapter 5.

Much of the above discussion is summarized in Table 4.1; specifically, assuming that $w_s = w_t = w$ in the 2D case, the connection between the above Gaussian conditional autoregressive models, finite difference discretizations of diffusion operator, and regularization matrices is summarized.
4.2. Choosing \( p(\mathbf{x}|\delta) \): Gaussian Markov random fields

<table>
<thead>
<tr>
<th>Conditional Autoregressive Model</th>
<th>Associated Diffusion Operator</th>
<th>Regularization or Precision Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_i</td>
<td>\mathbf{x}<em>0 \sim \mathcal{N}(\pi</em>{\mathbf{x}_i},(n_i w_i)^{-1}) )</td>
<td>( - \frac{\partial}{\partial s} \left( w(s) \frac{\partial}{\partial s} \right) )</td>
</tr>
<tr>
<td>( x_{ij}</td>
<td>\mathbf{x}<em>{0,ij} \sim \mathcal{N}(\pi</em>{\mathbf{x}<em>{ij}},(n</em>{ij} w_{ij})^{-1}) )</td>
<td>( - \frac{\partial}{\partial s} \left( w(s,t) \frac{\partial}{\partial s} \right) - \frac{\partial}{\partial t} \left( w(s,t) \frac{\partial}{\partial t} \right) )</td>
</tr>
</tbody>
</table>

Table 4.1. Conditional autoregressive models and differential operators yielding the same regularization/precision matrix after discretization. The table does not contain mention of the boundary conditions or of the parameters \( \delta \) and \( 1/h^2 \). Note that \( w \equiv 1 \) yields the negative-Laplacian.

4.2.2 Maximum a posteriori estimation for GMRF priors

We now turn to the problem of computing solutions to inverse problems formulated using Bayes’ law together with one of the GMRF (or IGMRF) priors defined above. Each of these priors can be written in the form \( p(\mathbf{x}|\delta) \propto \exp \left( -\frac{\delta}{2} \mathbf{x}^T L \mathbf{x} \right) \), and we use maximum a posteriori (MAP) estimation to obtain an estimate of \( \mathbf{x} \). The MAP estimator is the maximizer of the posterior density function \( p(\mathbf{x}|\mathbf{b},\lambda,\delta) \), defined in (4.3), or equivalently, the minimizer of \(-\ln p(\mathbf{x}|\mathbf{b},\lambda,\delta)\), i.e.,

\[
\mathbf{x}_{\text{MAP}} = \arg \min_{\mathbf{x}} \left\{ \ell(\mathbf{x}) \overset{\text{def}}{=} \frac{\lambda}{2} \| \mathbf{A} \mathbf{x} - \mathbf{b} \|^2 + \frac{\delta}{2} \mathbf{x}^T \mathbf{L} \mathbf{x} \right\}. \tag{4.32}
\]

It can be shown (see Exercise 4.4) that if the null spaces of \( \mathbf{A} \) and \( \mathbf{L} \) intersect only at \( \mathbf{0} \), then \( \mathbf{x}_{\text{MAP}} \) is unique and is also the solution of the linear system of equations

\[
(\mathbf{A}^T \mathbf{A} + \alpha \mathbf{L}) \mathbf{x} = \mathbf{A}^T \mathbf{b}, \quad \text{where} \quad \alpha = \frac{\delta}{\lambda}. \tag{4.33}
\]

Note that (4.33) is the same as (2.9), but with \( \mathbf{L} \) defined by one of the above GMRF priors rather than \( \mathbf{I} \).

Regularization parameter selection methods for choosing \( \alpha \) can be implemented just as in Chapters 2 and 3. If \( \lambda = 1/\sigma^2 \) is known, UPRE or DP can be used, otherwise GCV or the L-curve method must be used. Here we use GCV and leave the implementations of DP, UPRE, and L-curve to the reader in Exercise 4.7. Recall that the GCV function has the form (see (2.24))

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

where \( \mathbf{A}_\alpha = (\mathbf{A}^* \mathbf{A} + \alpha \mathbf{L})^{-1} \mathbf{A}^* \), where ‘*’ denotes conjugate transpose.

In the one-dimensional deblurring and kernel reconstruction cases from Chapter 1, minimizing \( G \) is computationally feasible. If we compute \( \alpha \) using GCV for
the GMRF prior defined by (4.16), with $\mathbf{L}$ defined by (4.14), we obtain the reconstructions plotted in Figure 4.3.

In two-dimensional image deblurring, 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 on $x$, the matrices $\mathbf{A}$ and $\mathbf{L}$, as derived in Chapters 2 and 3, respectively, will both be BCCB, and hence will be diagonalizable by the 2D-DFT. Specifically, recall from Section 3.1.2 that $\mathbf{A}$ has spectral decomposition (3.28) and from Section 4.2.1 that $\mathbf{L}$ has spectral decomposition (4.19). Then, similar to the case of standard Tikhonov regularization with periodic boundary conditions (see (3.31)), the regularized solution can be expressed in the array form

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

Moreover, the GCV function can be written

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

where $\hat{\mathbf{B}} = n \text{DFT}(\mathbf{B})$, and $| \cdot |$ denotes component-wise modulus. Equation (4.35) can be derived from (2.25) and should be compared with (3.32), where instead of $\mathbf{1}_n$, the array of ones $\mathbf{1}$ is used instead. The blurred noisy data is the same as that considered in Chapter 3; see Figure 3.1. The reconstruction obtained using (4.34) with $\alpha$ equal to the nonnegative minimizer of (4.35) is presented in the upper-left in Figure 4.4.

**Figure 4.3.** One-dimensional, GMRF prior test cases with $\mathbf{L}$ defined by (4.14) and GCV for computing $\alpha$. On the left is the reconstruction obtained in the deblurring case. On the right is reconstruction obtained in the kernel reconstruction case. In both cases, $\alpha$ .

![Graph](image-url)
4.2. Choosing $p(x|\delta)$: Gaussian Markov random fields

![Figure 4.4](image_url)

**Figure 4.4.** Two-dimensional test case, IGMRF prior with $L = D_s^T A D_s + D_t^T A D_t$ and using GCV for computing $\alpha$. On the left is the reconstruction obtained using $A = I$, which corresponds to the auto regressive model (4.13). On the right is the reconstruction obtained using Algorithm 4.1 after ten iterations. In both cases, a periodic boundary condition was assumed.

**Edge Preserving Reconstruction using IGMRF Priors**

We now focus on independent increment examples. In one-dimension, this yields a prior of the form $p(x|\delta) \propto \exp\left(-\frac{1}{2}x^T D^T A D x\right)$, with conditional autoregressive model (4.28). In this subsection, we assume a periodic boundary condition, and hence

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

(4.36)

Now, if the goal is to preserve edges in reconstructions, an intuitive choice for $\Lambda = \text{diag}(w_1, \ldots, w_n)$ is $w_i^{-1} \propto |x_{i+1} - x_i|$, or in order to avoid division by zero, define

$$\Lambda(x) = \text{diag} \left(\frac{1}{\sqrt{(Dx)^2 + \beta}}\right),$$  

(4.37)

where vector operations are computed component-wise and $0 < \beta \ll 1$ (we use $\beta = 0.001$). We can motivate this choice by noting that in the deblurring example, if $\Lambda = \Lambda(x_{\text{true}})$, we obtain the reconstruction on the left in Figure 4.5, which is quite good.

In two-dimensions, again assuming a periodic boundary condition, the independent increment model (4.29) with $w_{ij} = w_i^s = w_i^t$ results in a prior of the form $p(x|\delta) \propto \exp\left(-\frac{1}{2}x^T (D_s^T A D_s + D_t^T A D_t) x\right)$, with conditional autoregressive model (4.31). Again, if the goal is to preserve edges, an intuitive choice for $\Lambda = \text{diag} \left(\text{vec}\left(\{w_{ij}\}_{i,j=1}^n\right)\right)$ is $w_{ij}^{-1} \propto \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{ij})^2}$, or in or-
Figure 4.5. One-dimensional test case, IGMRF prior with \( L = D^TAD \) and using GCV for computing \( \alpha \). On the left is the reconstruction obtained when \( |\Lambda_i|_1 = 1/\sqrt{|(Dx_{\text{true}})_i + \beta|} \). On the right is the reconstruction obtained after 10 iterations of Algorithm 4.1.

In order to avoid division by zero, define

\[
\Lambda(x) = \text{diag} \left( \frac{1}{\sqrt{(D_s x)_1^2 + (D_t x)_1^2 + \beta}} \right),
\]

where vector operations are computed component-wise, and \( D_s = I \otimes D \) and \( D_t = D \otimes I \).

Ideally, we would use \( \Lambda = \Lambda(x_{\text{true}}) \), but since we don’t know \( x_{\text{true}} \), it makes sense to estimate it iteratively and use these estimates to also update \( \Lambda \): in one-dimension via (4.37), and in two-dimensions via (4.38). This motivates the following algorithm.

**Algorithm 4.1. Edge-Preserving Reconstruction Using IGMRF Priors.**

0. Set \( k = 1 \) and \( k_{\text{max}} \) to be the maximum number of iterations. In one-dimension, set \( A^0 = I \), and in two-dimensions, set \( A_s^0 = A_t^0 = I \).

1. In one-dimension, define \( L_{k-1} = D^T \Lambda^{k-1} D \). In two-dimensions, define \( L_{k-1} = D_s^T \Lambda_s^{k-1} D_s + D_t^T \Lambda_t^{k-1} D_t \).

2. Compute the MAP estimator \( x_{k-1} = (A^T \Lambda + \alpha_{k-1} L_{k-1})^{-1} A^T b \), with \( \alpha_{k-1} \) chosen using a regularization parameter selection method such as GCV.

3. In one-dimensions, define \( \Lambda_k^A = \Lambda(x_{k-1}) \) via (4.37). In two-dimensions, define \( \Lambda_k^A = \Lambda_k^A = \Lambda(x_{k-1}) \) via (4.38).

4. If \( k = k_{\text{max}} \), stop iterations. Otherwise, set \( k = k + 1 \) and return to Step 1.

We present the reconstructions that results after 10 iterations of Algorithm 4.1, in one-dimension on the right in Figure 4.5, and in two-dimensions on the right...
4.3. Choosing $p(x|\delta)$: Laplace distributed increments

in Figure 4.4. We leave the application of this algorithm to the kernel reconstruction and computed tomography test cases to Exercise 8.

In steps 1 and 3 of Algorithm 4.1, in two-dimensions, we allow for different diagonal matrices $\Lambda_s$ and $\Lambda_t$, which is unnecessary here, but will be useful in the next section.

In step 2 of Algorithm 4.1, in two-dimensions, special structure in the problem must be exploited in order to make the computations feasible. Note that the matrix $L_{k-1}$ does not have circulant structure even if $x$ was assumed to be periodic. Thus in two-dimensions we must use PCG, Algorithm 3.3, to compute the solution $x_k$.

Our implementation of PCG is similar to that for the tomography and data driven boundary condition cases in Section 3.1.3, except that $B = A^T A + \alpha_{k-1} L_{k-1}$, where $L_{k-1} = D^T_s A^{k-1}_s D_s + D^T_t A^{k-1}_t D_t$.

Moreover, we use the preconditioner corresponding to $A^{k-1} = I$, given by

$$M^{-1} = \text{vec} \left( \text{IDFT} \left( \frac{1}{|\hat{a}_i|^2 + \alpha_{k-1} I} \odot \text{DFT} (R) \right) \right),$$

where $r = \text{vec}(R)$, and $\hat{I}_s$ is the $n \times n$ eigenvalue array of the 2D discrete negative-Laplacian with periodic BCs, as defined above. Moreover, the implementation of GCV, UPRE, and L-curve are all as in the data driven boundary condition case, where an additional application of PCG was required per iteration of the parameter selection method.

In Step 4 of Algorithm 4.1, the update of the regularization parameter $\alpha$ is at the user’s discretion: it could be fixed at the outset, updated at every iterations, and everything in between.

4.3 Choosing $p(x|\delta)$: Laplace distributed increments

Our discussion in this section follows that in [5]. An alternative assumption to Gaussian distributed increments is that increments are Laplace distributed, i.e.,

$$\Delta x_i \overset{\text{iid}}{\sim} \text{Laplace}(0, \delta^{-1}), \quad i = 1, \ldots, n-1,$$

where Laplace($\mu, \delta^{-1}$) has probability density function

$$p(x|\mu, \delta^{-1}) = \frac{\delta}{2} \exp \left( -\delta |x - \mu| \right).$$

Because of our assumption of independence, the joint density for $x$ (i.e., the prior) is given by

$$p(x|\delta) \propto \delta^{n-1} \exp \left( -\delta \sum_{i=1}^{n-1} |x_i - x_{i-1}| \right) = \delta^{n-1} \exp (-\delta \|Dx\|_1),$$

where $\| \cdot \|_1$ denotes the $\ell^1$-norm and $D$ is the forward difference matrix (4.22). As above, a Neumann boundary condition is assumed here, but we can easily modify $D$ so that a periodic (see (4.36)) or Dirichlet boundary condition is imposed instead.
Figure 4.6. The Laplace and Normal probability density functions with mean 0 and variance 1.

The assumption of Laplacian increments is motivated by the fact that in many signals, the increments sizes are typically small or close to a fixed mean, but outliers (large increments) are not uncommon. Due to the fact that the Laplace distribution has heavy tails, large increments are much more probable than if a Gaussian increment model (as in [42]) is assumed. To illustrate the difference between the Laplace and Gaussian probability densities, we plot them together in Figure 4.6.

The prior (4.40) yields total variation regularization [49]. The connection between the Laplacian increment model (4.39) and total variation regularization for one-dimensional signals is discussed in some detail in [23]. Here, we also use the Laplace increment model for two-dimensional signals, which yields a regularization that is very similar to two-dimensional total variation, yielding visually indistinguishable regularized solutions. Moreover, in [5], it is shown that in the infinite dimensional (function space) setting, both regularizations yield solutions in the space of bounded variation. In two-dimensions, we assume

\[ \Delta_s x_{ij} \text{iid} \sim \text{Laplace}(0, \delta^{-1}), \quad i = 1, \ldots, n - 1, \quad j = 1, \ldots, n, \]
\[ \Delta_t x_{ij} \text{iid} \sim \text{Laplace}(0, \delta^{-1}), \quad i = 1, \ldots, n, \quad j = 1, \ldots, n - 1. \] (4.41)

Then the probability density function for \( \mathbf{x} \) has the form (see [42, Chapter 3])

\[
p(\mathbf{x}|\delta) \propto \exp \left( -\frac{\delta}{2} \left( \sum_{j=1}^{n} \sum_{i=1}^{n-1} |x_{i+1,j} - x_{ij}| + \sum_{i=1}^{n} \sum_{j=1}^{n-1} |x_{i,j+1} - x_{ij}| \right) \right)
\]

\[= \exp \left( -\frac{\delta}{2} (\|D_s \mathbf{x}\|_1 + \|D_t \mathbf{x}\|_1) \right). \] (4.42)
4.3. Choosing \( p(x|\delta) \): Laplace distributed increments

Note that in contrast the two-dimensional total variation prior has the form

\[
p(x|\delta) \propto \exp \left( -\frac{\delta}{2} \sum_{i=1}^{n(n-1)} \sqrt{|D_{s}x|_{i}^{2} + |D_{t}x|_{i}^{2}} \right). \tag{4.43}
\]

It is well-known that when the total variation prior is used, the resulting reconstructed images have a cartoon texture, or in other words are approximately piecewise constant. This is also the case for the Laplace prior. Moreover, because the variance \( \delta \) can be chosen differently for the vertical and horizontal increments in (4.41), this regularization is sometimes referred to as anisotropic total variation. Such an approach is discussed in the Gaussian case in [42], and one could also mix the Laplace and Gaussian increments, if for example, one knew that edges occurred only along one of the coordinate directions.

4.3.1 Maximum a posteriori estimation for the Laplace prior

For Laplace increments, the prior, which is given by (4.40) in one dimension and (4.42) in two dimensions, is not differentiable, hence we use the following differentiable approximations: in one-dimension,

\[
p(x|\delta) \propto \exp \left( -\frac{\delta}{2} \sum_{i=1}^{n-1} \psi((x_{i+1} - x_{i})^{2}) \right),
\]

where \( \psi(t) = 2\sqrt{t + \beta} \); whereas in two-dimensions,

\[
p(x|\delta) \propto \exp \left( -\frac{\delta}{2} \left( \sum_{j=1}^{n} \sum_{i=1}^{n-1} \psi((x_{i+1,j} - x_{ij})^{2}) + \sum_{i=1}^{n} \sum_{j=1}^{n-1} \psi((x_{i,j+1} - x_{ij})^{2}) \right) \right).
\]

In this case, the MAP estimation problem (4.32) is no longer a quadratic optimization problem, and hence, \( x_{\text{MAP}} \) is not the solution of a linear system of equations, as in the Gaussian case. Hence an iterative, nonlinear optimization method must be used.

One of the most standard methods for solving (4.32) when using the Laplace prior (or total variation regularization) is the lagged-diffusivity fixed point iteration [49], which can be motivated as follows. First, the cost function to be minimized is given by \( \ell(x) \overset{\text{def}}{=} -\ln p(x|b, \lambda, \delta) \), which has gradient (with respect to \( x \))

\[
\nabla \ell(x) = \lambda A^{T}(Ax - b) + \delta L(x)x,
\]

where (see [49]), in one-dimension

\[
L(x) = D_{s}^{T}\text{diag}((\psi'((D_{s}x)^{2}))D_{s}, \tag{4.44}
\]

while in two-dimensions

\[
L(x) = D_{s}^{T}\text{diag}((\psi'((D_{s}x)^{2}))D_{s} + D_{t}^{T}\text{diag}((\psi'((D_{t}x)^{2}))D_{t}, \tag{4.45}
\]
Chapter 4. Bayes' Law, Markov Random Field Priors, and MAP Estimation

Note that \( \psi'(Dx^2) = 1/\sqrt{(Dx)^2 + \beta I} \), which is the same as \( \Lambda(x) \) defined in (4.37). Newton's method for minimizing \( \ell(x) \) is given by [39]

\[
x_k = x_{k-1} - \nabla^2 \ell(x_{k-1})^{-1} \nabla \ell(x_{k-1}),
\]

where \( \nabla^2 \ell(x) \) is the Hessian of \( \ell(x) \) with respect to \( x \) and has the form

\[
\nabla^2 \ell(x) = \lambda A^T A + \delta (L(x) + L'(x)x),
\]

with \( L'(x) \) the Jacobian of \( L(x) \). Lagged-diffusivity can then be obtained by dropping \( L'(x)x \) from the Hessian, yielding the quasi-Newton iteration

\[
x_k = x_{k-1} - (\lambda A^T A + \delta L(x_{k-1}))^{-1}(\lambda A^T (Ax_{k-1} - b) + \delta L(x_{k-1})x_{k-1}).
\] (4.46)

If \( A^T A \) is positive definite, one can prove that this iteration converges to the MAP estimator (see [49] and the references therein). It is a straightforward exercise (see Exercise 9) to show that (4.46) can be equivalently expressed

\[
x_k = (\lambda A^T A + \delta L(x_{k-1}))^{-1} \lambda A^T b,
\] (4.47)

which can be seen to be the MAP estimator for the Gaussian posterior density function

\[
p(x|b, x_{k-1}) \propto \exp \left( -\frac{\lambda}{2} \| Ax - b \|^2 - \frac{\delta}{2} x^T L(x_{k-1})x \right).
\] (4.48)

From the final observation, it is evident that lagged-diffusivity is exactly the iterative, edge-preserving reconstruction method given in Algorithm 4.1, with the minor modification that step 3 is replaced by

3. In one-dimension, define \( A^k = \Lambda(x_{k-1}) \) via (4.37). In two-dimensions, define \( \Lambda^k = \text{diag} \left( 1/\sqrt{(D_s x_{k-1})^2 + \beta I} \right) \) and \( A^k = \text{diag} \left( 1/\sqrt{(D_t x_{k-1})^2 + \beta I} \right) \).

Thus lagged-diffusivity can be viewed as a method that iteratively approximates the non-Gaussian posterior density in the Laplace prior case with a Gaussian probability density of the form (4.48). Given that lagged-diffusivity is implemented in Algorithm 4.1, the results of the method are given on the right in Figure 4.5, for the one-dimensional case, and on the right in Figure 4.4, for the two-dimensional case.

4.4 The Infinite-Dimensional Limit

In our treatment above, the dimensions \( m \) and \( n \) of the problem are fixed by the numerical mesh, making the problems are discrete. However, we have also taken pains to relate the discrete problems to underlying continuous differential and integral equations. Since \( m \) is the dimension of our measurement vector, we assume that it is a fixed value, however, we are free to choose \( n \) as we please, and the behavior of our approaches in the infinite limit, i.e., as \( n \to \infty \), is an important question, which is dealt with in depth in, e.g., [44].
4.4. The Infinite-Dimensional Limit

Using our notation, established in this and previous chapters, we define

$$A_m x = \lim_{n \to \infty} Ax,$$

where recall that $A \in \mathbb{R}^{m \times n}$ is a matrix resulting from a full discretization (i.e., of both the domain and range) of one of the integral operators described in Chapter 1 and 3. The operator $A_m : L^2(\Omega) \to \mathbb{R}^m$, where $\Omega = [0, 1]$ or $[0, 1] \times [0, 1]$, is what results after discretization with respect to the range only, e.g., for the one-dimensional deblurring model (1.4),

$$[A_m x]_i = \int_{-1}^{2} a(s_i - s') x(s') ds', \quad i = 1, \ldots, m.$$

Moreover, the prior models defined in Section 4.2 all have the form $p(x|\delta) \propto \delta^{(n-1)/2} \exp\left(-\frac{1}{2} x^T L x\right)$, where $\lim_{n \to \infty} n^2 L x = Lx$, with $L$ given by one of the continuous, second-order differential operators found in Table 4.1, plus boundary conditions. Furthermore, if $(x, y)_{L^2}$ denotes the $L^2$ inner-product, then

$$\lim_{n \to \infty} n^2 c(n) x^T L x = (x, Lx)_{L^2},$$

where if mid-point quadrature is used, $c(n) = 1/n$ in one-dimension, and $c(n) = 1/n^2$ in two-dimensions. Thus, finally, if $\delta_n \equiv c(n)n^2\delta$, we have

$$\lim_{n \to \infty} n^2 c(n) x^T L x = (x, Lx)_{L^2},$$

as well as

$$\lim_{n \to \infty} \left\{ \frac{\lambda}{2} \|Ax - b\|^2 + \frac{\delta}{2} x^T L x \right\} = \frac{\lambda}{2} \|A_m x - b\|^2 + \frac{\delta}{2} (x, Lx)_{L^2}, \quad (4.49)$$

where ‘$\ast$’ denotes operator adjoint. Note that in one-dimension $\delta_n = \delta/n$, whereas in two-dimensions, $\delta_n = \delta$. Hence in two-dimensions, with $L$ defined as in Section 4.2, no rescaling of $\delta$ is needed for the infinite dimensional limit to satisfy (4.49) and (4.50). For treatments of inverse problems in the infinite dimensional setting, which include regularization by a differential operator as in (4.49), see [16, 49].

Equations (4.49) and (4.50) illustrate the convergence properties of the MAP estimator as $n \to \infty$, however a question that remains is whether or not the resulting posterior density function,

$$p(x|b, \lambda, \delta) \propto \exp\left(-\frac{\lambda}{2} \|A_m x - b\|^2 - \frac{\delta}{2} (x, Lx)_{L^2}\right), \quad (4.51)$$

is well-defined. As can be found in [44], (4.51) is a well-defined posterior measure on $L^2(\Omega)$ provided $L^{-1}$ is a trace class operator. In one-dimension, this is the case for

$$L = -\frac{d}{ds} \left( w(s) \frac{d}{ds} \right), \quad 0 < s < 1.$$
with zero, Neumann, or periodic boundary conditions, and where \( w \) is continuous and bounded away from zero on \([0,1]\). In two-dimensions, if

\[
\mathcal{L} = -\frac{\partial}{\partial s} \left( w_x(s,t) \frac{\partial}{\partial s} \right) - \frac{\partial}{\partial t} \left( w_t(s,t) \frac{\partial}{\partial t} \right), \quad 0 < s, t < 1,
\]

then \( \mathcal{L}^{-1} \) is not a trace class operator, and the posterior measure (4.51) is not well-defined. However, if

\[
\mathcal{L} = \left( -\frac{\partial}{\partial s} \left( w_x(s,t) \frac{\partial}{\partial s} \right) - \frac{\partial}{\partial t} \left( w_t(s,t) \frac{\partial}{\partial t} \right) \right)^2, \quad 0 < s, t < 1, \tag{4.52}
\]

\( \mathcal{L}^{-1} \) is trace-class and (4.51) is well-defined, provided \( w_x, w_t \) are continuous and bounded away from zero on \([0,1] \times [0,1]\). For \( w_x = w_t = 1 \), (4.52) is called the biharmonic operator.

It is difficult to relate (4.52) to pixel-level probability assumptions about increments, except in the case that \( w_x = w_t = 1 \) [42]. A trace-class operator that can be easily related to pixel-level assumptions about increments is

\[
\mathcal{L} = -\frac{\partial^2}{\partial s^2} \left( w_x(s,t) \frac{\partial^2}{\partial s^2} \right) - \frac{\partial^2}{\partial t^2} \left( w_t(s,t) \frac{\partial^2}{\partial t^2} \right), \quad 0 < s, t < 1, \tag{4.53}
\]

together with zero, Neumann, or periodic boundary conditions.

### 4.4.1 Higher-order GMRFs

The PDE (4.53) is related to the Gaussian Markov random field defined by (analogous to (4.29))

\[
\Delta^2 x_{ij} \sim \mathcal{N}(0, (w^2_{ij}\delta)^{-1}) \quad \text{and} \quad \Delta^2 x_{ij} \sim \mathcal{N}(0, (w^2_{ij}\delta)^{-1}), \tag{4.54}
\]

for \( i, j = 1, \ldots, n \), where the increments are now second-order and are given by \( \Delta^2 x_{ij} = 2x_{ij} - x_{i+1,j} - x_{i-1,j} \) and \( \Delta^2 x_{ij} = 2x_{ij} - x_{ij+1} - x_{ij-1} \). Then the probability density function for \( x \) is the following intrinsic Gaussian:

\[
p(x|\delta) \propto \delta^{(n^2-2)/2} \exp \left( -\frac{\delta}{2} \left( \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^2 (\Delta^2 x_{ij})^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}^2 (\Delta^2 x_{ij})^2) \right) \right)
\]

\[
= \delta^{(n^2-2)/2} \exp \left( -\frac{\delta}{2} \left( \|A^1_{x}/(I \otimes L_{1D})x\|^2 + \|A^1_{x}/(L_{1D} \otimes I)x\|^2) \right) \right)
\]

\[
= \delta^{(n^2-2)/2} \exp \left( \frac{\delta}{2} x^T (L^T_{1D}A_{s}L_{s} + L^T_{1D}A_{L}L_{L})x \right), \tag{4.55}
\]

where \( L_s = I \otimes L_{1D} \) and \( L_L = L_{1D} \otimes I \) are the discrete horizontal and vertical second derivatives, respectively, \( A_s = \text{diag}(\{w_{ij}^s\}_{i,j=1}^{n}) \), and \( A_L = \text{diag}(\{w_{ij}^l\}_{i,j=1}^{n}) \). Note that the power \( n^2-2 \) is the rank of the precision matrix \( L^T_{1D}A_sL_s + L^T_{1D}A_sL_s \). Moreover, \( \lim_{\delta \to 0} 1/(h^2)(L^T_{1D}A_sL_s + L^T_{1D}A_sL_s) = \mathcal{L} \), where \( \mathcal{L} \) is defined in (4.53). And finally, if \( \delta_n = c(n)n^2 = n^2 \), then (4.49) and (4.50) hold.
An edge-preserving reconstruction method can be implemented using (4.55) that is very similar to Algorithm 4.1. We present it next and leave its implementation to the reader in Exercise 11.

Algorithm 4.2. Image Reconstruction with Second-Order, Adaptive IGMRF Priors.

0. Set $k = 1$. In one-dimension, set $A^0 = I$. In two-dimensions, set $A^0_s = A^0_t = I$.

Choose $\alpha_0$.

1. Define $L_{k-1} = L_s^{T} A_s^{k-1} L_s + L_t^{T} A_t^{k-1} L_t$.

2. Compute the MAP estimator $x_k = (A^T A + \alpha_k L_k)^{-1} A^T b$.

3. Define $A_s^k = \text{diag} \left( \frac{1}{\sqrt{\left(L_s x_k \right)^2 + \beta}} \right)$ and $A_t^k = \text{diag} \left( \frac{1}{\sqrt{\left(L_t x_k \right)^2 + \beta}} \right)$.

4. Update $\alpha_k$, if desired, using a parameter selection method such as GCV.

5. If $k = k_{\text{max}}$, stop iterations. Otherwise, set $k = k + 1$ and return to Step 1.

We also note that just as in Section 4.3, $\Delta_s^2 x_{ij}$ and $\Delta_t^2 x_{ij}$ can be assumed to be Laplace distributed, and that a lagged-diffusivity-type algorithm can be derived, as in Section 4.3, that coincides with Algorithm 4.2.

Exercises

4.1. Finite difference discretization of $x'(s)$.

a. Let $h = 1/n$ and $s_i = (i - 1/2)h$ for $i = 0, 1, \ldots, n$. Expand $x(s_i + h)$ in a three term Taylor series about $s_i$ and show that

$$x'(s_i) = \frac{x(s_{i+1}) - x(s_i)}{h} + E, \quad i = 0, 1, \ldots, n,$$  \hspace{1cm} (4.56)

where $E$ has order $h$. Assume that $x$ is sufficiently differentiable and define $x_i = x(s_i)$ for $i = 0, 1, \ldots, n + 1$.

b. Write down the expression on the right-hand side in (4.56) (ignoring $E$) for each $i$. Assume a zero boundary condition, i.e., $x_0 = x_{n+1} = 0$, and derive the $(n+1) \times n$ matrix $D$ such that these $n + 1$ equations can be expressed as $Dx$, where $x = (x_1, \ldots, x_n)$, and show that $D^T D = L_{1D}$ from (4.14).

c. Proceed as in part b, but assume a Neumann boundary condition, i.e., $x_0 = x_1$ and $x_{n+1} = x_n$. Derive the $(n-1) \times n$ matrix such that these $n + 1$ equations can be expressed $Dx$, and then show that $D$ is given by (4.22) and $D^T D = L_{1d}$ from (4.14), but with $[L_{1d}]_{11} = [L_{1d}]_{nn} = 1$. 


d. Proceed as in part b, but assume a periodic boundary condition, i.e.,

\[ x_0 = x_n \text{ and } x_{n+1} = x_1. \]

Derive the \( n \times n \) matrix such that these \( n + 1 \) equations can be expressed as \( D x \), and then show that \( D \) is given by (4.36) and \( D^T D = L_{\text{ID}} \) from (4.17).

4.2. Finite difference discretization of \( x''(s) \).

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

\[
-x''(s_i) = \frac{-x(s_{i-1}) + 2x(s_i) - x(s_{i+1})}{h^2} + E, \quad i = 1, \ldots, n, \quad (4.57)
\]

where \( E \) has order \( h^2 \). Assume that \( x \) is sufficiently differentiable.

b. Let \( x_1 = x(s_i) \) and write down the expression on the right-hand side in (4.57) (ignoring \( E \)) for \( i = 1, \ldots, n \). Assuming a zero boundary condition, i.e., \( x_0 = x_{n+1} = 0 \), show that the \( n \) equations can be equivalently expressed as \( L_{\text{ID}} x \), where \( x = (x_1, \ldots, x_n) \) and \( L_{\text{ID}} \) is defined in (4.14).

c. Proceed as in part b, but assume a Neumann boundary condition, i.e.,

\[ x'(0) = x'(1) = 0. \]

Use a centered difference approximation of \( x' \) to derive the conditions \( x_0 = x_1 \) and \( x_{n+1} = x_n \). Then from this show that the \( n \) equations can be equivalently expressed as \( L_{\text{ID}} x \), where \( L_{\text{ID}} \) is defined in (4.14), but with \( [L_{\text{ID}}]_{11} = [L_{\text{ID}}]_{nn} = 1 \).

d. Proceed as in part b, but assume a periodic boundary condition, i.e., \( x_0 = x_n \) and \( x_{n+1} = x_1 \). Show that the \( n \) equations can be equivalently expressed as \( L_{\text{ID}} x \), where \( L_{\text{ID}} \) is defined in (4.17).

e. Now consider the operator \(-\frac{d}{ds} (w(s) \frac{d}{ds}) x(s), 0 \leq s \leq 1. \) Choose one of the boundary conditions and associated computational grids above and discretize this operator. You should end up with a grid representation of the form \( D^T A D x \), where \( D \) is as derived in Exercise 4.1 for the assumed boundary condition and computational grid, and \( A \) is diagonal with \([A]_i = w(s_i)\).

4.3. Finite difference discretization of \( \frac{\partial}{\partial s} x(s, t), \frac{\partial}{\partial t} x(s, t), \) and \((- \frac{\partial^2}{\partial s^2} - \frac{\partial^2}{\partial t^2}) x(s, t)\).

a. Let \( h = 1/n, s_i = (i - 1/2)h \) and \( t_j = (j - 1/2)h, \) for \( i, j = 1, \ldots, n \).

Use the result in Exercise 4.1 to obtain

\[
\frac{\partial}{\partial s} x(s_i, t_j) = \frac{x(s_{i+1}, t_j) - x(s_{i-1}, t_j)}{h} + E, \quad i, j = 1, \ldots, n, \quad (4.58)
\]

where \( E \) has order \( h \). Assume a periodic boundary condition, i.e.,

\[ x(s_0, t_j) = x(s_n, t_j) \text{ and } x(s_{n+1}, t_j) = x(s_1, t_j) \text{ for } j = 1, \ldots, n, \]

and let \( X \) be the \( n \times n \) array with entries \([X]_{ij} = x(s_i, t_j)\). Show that the right-hand side of (4.58) (ignoring \( E \)) can then be expressed \( D_{s} x \), where \( D_{s} = I \otimes D \) with \( D \) given by (4.36) and \( x = \text{vec}(X) \). An analogous argument shows that the grid representation for \( \frac{\partial}{\partial t} x \), assuming a periodic boundary condition, is \( D_{t} x = (D \otimes I) x \), where \( D \) is given by (4.36).
b. Assuming the same computational grid as in part a, use the result in Exercise 4.2 to obtain the grid approximation
\[
-\frac{\partial^2}{\partial s^2} x(s_i,t_j) = \frac{-x(s_{i-1},t_j) - x(s_{i+1},t_j) + 2x(s_i,t_j) + E}{h^2}, \tag{4.59}
\]
for \(i,j = 1, \ldots, n\), where \(E\) has order \(h^2\). Repeating your arguments in part (a), again assuming a periodic boundary condition, show that the right-hand side of (4.59) (ignoring \(E\)) can be expressed \((I \otimes L_{1D})x\), where \(L_{1D}\) is given by (4.17).

c. Proceed as in part (b), but assume a zero boundary condition on the computational grid: \(h = 1/n, s_i = (i-1/2)h\) and \(t_j = (j-1/2)h\), for \(i,j = 1, \ldots, n\). In this case, \(D\) is the \((n+1) \times n\) matrix derived in Exercise 4.1(b), and \(L_{1D}\) is defined by (4.14). Use Kronecker product properties to show that \((D^T D)\) is given by (4.17), thus the grid representation of the negative Laplacian operator \(-\partial^2/\partial^2\) is given by \(L_{2D} = I \otimes L_{1D} + L_{1D} \otimes I\). Use Kronecker product properties to show that \(L_{2D} = D^T D_s + D^T D_t\), where \(D_s\) and \(D_t\) are defined in part (a).

d. Proceed as in part (b), but assume a Neumann boundary condition on the computational grid: \(h = 1/n, s_i = (i-1/2)h\) and \(t_j = (j-1/2)h\), for \(i,j = 1, \ldots, n\). In this case, \(D\) is the \((n-1) \times n\) matrix derived in Exercise 4.1 d, and \(L_{1D}\) is defined by (4.14), but with \([L_{1D}]_{11} = [L_{1D}]_{nn} = 1\).

e. An analogous argument to that in part (b) shows that the grid representation for \(-\partial^2/\partial^2\), assuming a periodic boundary condition, is \((L_{1D} \otimes I)x\), where \(L_{1D}\) is given by (4.17), thus the grid representation of the negative Laplacian operator \(-\partial^2/\partial^2\) is given by \(L_{2D} = I \otimes L_{1D} + L_{1D} \otimes I\). Use Kronecker product properties to show that \(L_{2D} = D^T D_s + D^T D_t\), where \(D_s\) and \(D_t\) are defined in part (a).

f. Now consider the operator 
\[-\frac{\partial^2}{\partial s^2} (w_s(s,t) \frac{\partial}{\partial s}) + \frac{\partial}{\partial t} (w_t(s,t) \frac{\partial}{\partial t}) x(s,t), \quad 0 \leq s,t \leq 1\]
Choose one of the boundary conditions and associated computational grids above and discretize this operator. You should end up with a grid representation of the form \((D^T \Lambda_s D_s + D^T \Lambda_t D_t) x\), where \(D_s\) and \(D_t\) are as derived in part (a) for the assumed boundary condition and computational grid, and \(\Lambda_s = \text{diag (vec}(W_s))\) and \(\Lambda_t = \text{diag (vec}(W_t))\), where \([W_s]_{ij} = w_s(s_i,t_j)\) and \([W_t]_{ij} = w_t(s_i,t_j)\) are the two-dimensional arrays consisting of the values of \(w_s\) and \(w_t\) on the computational grid.

4.4. The MAP estimator for a linear, iid Gaussian measurement model and GMRF prior.

a. Show that the gradient and Hessian of the objective function \(\ell(x)\) defined in (4.32) are given by \(\nabla \ell(x) = \lambda A^T (Ax - b) + \delta L x\) and \(\nabla^2 \ell(x) = \lambda A^T + \delta L\), respectively.

b. Show that \(\nabla \ell(x) = 0\) if and only if (4.5) holds.

c. Show that \(\nabla^2 \ell(x)\) is symmetric positive definite provided that the null spaces of \(A\) and \(L\) intersect only at \(0\). Assume \(L\) is defined as a GRMF (or IGMRF) prior, so that either \(L = D^T WD\) (in one-dimension) or \(L = D^T WD_s + D^T WD_t\) (in two-dimensions), where \(W\) is a diagonal matrix with nonnegative values along the diagonal.
d. Given the above results and the assumptions in part (c), show that \( \nabla \ell(x) = 0 \) has a unique solution, which we will call \( x_{\text{MAP}} \). Then show that \( x_{\text{MAP}} \) is also the unique minimizer of \( \ell(x) \). *Hint:* use part (c) to show that \( \ell(x) \) is a strictly convex function.

4.5. **Diagonalizing the discrete negative Laplacian with periodic boundary conditions, in one and two dimensions, using the DFT.**

a. Let \( L_{1D} \) be defined by (4.17). Note that it is a circulant matrix, and hence, it can be diagonalized by the DFT. Specifically, by (2.39), we have

\[
L_{1D} = F^* \text{diag}(\hat{c}) F, \quad \hat{c} = \sqrt{n} Fc,
\]

where \( c \) is the first column of \( L_{1D} \). Write down \( c \), then use this decomposition to generate the matrix \( L_{1D} \) in MATLAB with \( n = 10 \). Verify that it agrees with (4.17).

*Note:* in MATLAB, \( \hat{c} = \text{fft}(c) \), and \( L_{1D}x = \text{real}(\text{ifft}(\hat{c} \ast \text{fft}(x))) \).

b. Let \( L_{2D} \) be defined by (4.15), where \( L_{1D} \) is defined by (4.17). Verify that \( L_{2D} \) is a block circulant with circulant blocks (BCCB) matrix. Thus, as in (3.28), \( L_{2D} \) is diagonalizable by the two-dimensional DFT. To compute the \( n \times n \) eigenvalue array \( \hat{l}_s \) of \( L_{2D} \), note that multiplication by \( L_{2D} \) is equivalent to discrete convolution (3.3), with a periodic boundary condition, by the \( n \times n \) kernel

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

where \( l_{0,0} = 4, l_{-1,0} = l_{1,0} = l_{0,-1} = -1, \) and \( l_{ij} = 0 \) otherwise.

Thus, following (3.28), the \( n \times n \) eigenvalue array for \( L_{2D} \) is given by \( \hat{l}_s = n \text{DFT}(l_s) \), where \( l_s \) is defined as in (3.21). Hence, analogous to (3.30), if \( X \) is \( n \times n \) and \( x = \text{vec}(X) \), then

\[
Lx = \text{vec}\left(\text{IDFT}(l_s \odot \text{DFT}(X))\right).
\]

Use this decomposition to generate the matrix \( L_{2D} \) for \( n = 4 \) (in MATLAB for example) and show that it agrees with the Kronecker product definition (4.15). Note that in MATLAB \( l_s = \text{fft2}(l_s) \), where \( l_s = \text{fftshift}(l) \).

4.6. **The MAP estimator and regularization parameter selection for GMRF priors with periodic boundary conditions**

a. Show that if \( L \) is defined by (4.19) and \( A \) is defined by (3.29), then the solution of (4.33) is given by (4.34).
b. Derive the GCV function (4.35) in this case, as well as one or all of the following regularization parameter selection methods: i) discrepancy principle; ii) unbiased predictive risk estimator; and iii) L-curve.

c. Modify Deblur2dGMRF.m so that it implements whatever subset of regularization parameter selection methods you derived in part (b).

4.7. UPRE, DP, and L-curve regularization parameter selection methods for GMRF priors.

a. Modify Deblur1dGMRF.m and Deblur1dIGMRF.m so that some subset of the following regularization parameter selection methods are implemented instead: i) discrepancy principle; ii) unbiased predictive risk estimator; and iii) L-curve. Note that you should only have to change GCV_fn.m.

b. Modify Deblur2dIGMRF.m so that it implements one or all of the following regularization parameter selection methods: i) discrepancy principle; ii) unbiased predictive risk estimator; and iii) L-curve. Note that you should only have to change GCV_fn2d.m.

4.8. GMRF and IGMRF priors for kernel reconstruction and computed tomography:

a. Modify Deblur1dGMRF.m and Deblur1dIGMRF.m so that in place of image deblurring, the kernel reconstruction inverse problem from Chapters 1 and 2 is implemented.

b. Modify TomographyCG.m from Chapter 3 so that the prior \( p(\mathbf{x}|\delta) \propto \exp\left(-\frac{1}{2}\mathbf{x}^T L_{2D} \mathbf{x}\right) \), where \( L_{2D} \) is given by (4.15), (4.14), i.e., the smoothness prior GMRF with zero boundary conditions.

c. Modify Deblur2dIGMRF.m so that in place of image deblurring, the computed tomography example from Chapter 3 is implemented, as found in TomographyCG.m.

4.9. Lagged-diffusivity fixed point iteration:

a. Show that (4.46) and (4.47) are equivalent, and moreover, that (4.47) is the MAP estimator for the posterior density function (4.48).

b. Modify Deblur2dIGMRF.m so that it implements the alternate step 3 presented in the last paragraph of Section 4.3.1. Do you notice a difference in the reconstructions?

4.10. Data Driven Boundary Condition:

a. Modify Deblur1dDataDriven.m from Chapter 3 so that it implements Algorithm 4.1.

b. Modify Deblur2dDataDriven.m from Chapter 3 so that it implements Algorithm 4.1.

4.11. Higher-order Markov Random Fields:
a. Modify Deblur2dGMRF.m so that it uses the prior with precision matrix \( L \)
given by discretizing the biharmonic operator (4.52) with zero boundary
conditions. Note that this can be easily implemented by simply squaring
the negative Laplacian matrix. How does the reconstruction compare
with that obtained using the negative Laplacian?

b. Implement Algorithm 4.2 by modifying Deblur2dIGMRF.m,
Bmult_IGMRF.m, and GCV_fn2d.m.

c. Assume \( \Delta^2 x_{ij} = 2x_{ij} - x_{i+1,j} - x_{i-1,j} \) and \( \Delta^4 x_{ij} = 2x_{ij} - x_{ij+1} - x_{ij-1} \)
are Laplace distributed, and derive the corresponding lagged-diffusivity-
type fixed point iteration, mimicking the steps found in Section 4.3.
Verify that the resulting algorithm agrees with Algorithm 4.2.
Bibliography


Bibliography


