TIERS
CS

ems with Actuator

deling in Optical

Using Conductive

hmic

ential Equations.

ve Methods

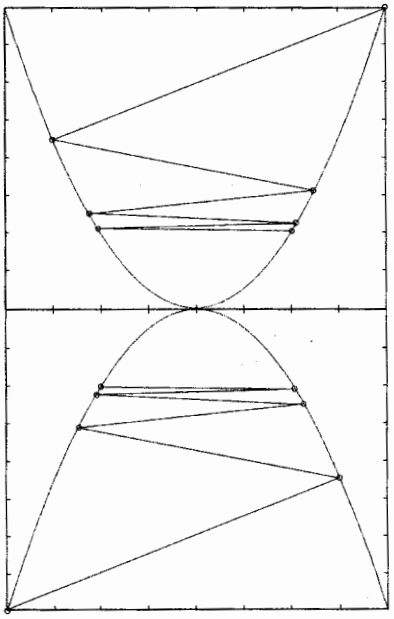
putational

juations.

ations

ons

Computational Methods for Inverse Problems



Curtis R. Vogel
Montana State University
Bozeman, Montana

siam
Society for Industrial and Applied Mathematics
Philadelphia

Contents

)ks/fr23) to make
)n of MATLAB*
aders to conduct
so provides tem-
ution techniques
ms to be used to

:tion of solutions

: friends Martin
'e strongly influ-
ıd corrections to
)ther colleagues
Esty, Luc Gilles,
ılly, I would like
ssistance.

A 01760, USA, tel.

Chapter 1
Introduction

Inverse problems arise in a variety of important applications in science and industry. These range from biomedical and geophysical imaging to groundwater flow modeling. See, for example, [6, 7, 35, 70, 87, 90, 107, 108] and the references therein. In these applications the goal is to estimate some unknown attributes of interest, given measurements that are only indirectly related to these attributes. For instance, in medical computerized tomography, one wishes to image structures within the body from measurements of X-rays that have passed through the body. In groundwater flow modeling, one estimates material parameters of an aquifer from measurements of pressure of a fluid that immerses the aquifer. Unfortunately, a small amount of noise in the data can lead to enormous errors in the estimates. This instability phenomenon is called ill-posedness. Mathematical techniques known as regularization methods have been developed to deal with ill-posedness. This chapter introduces the reader to the concepts ill-posedness and regularization. Precise definitions are given in the next chapter.

1.1 An Illustrative Example

Consider the Fredholm first kind integral equation of convolution type in one space dimension:

$$g(x) = \int_0^1 k(x - x')\, f(x')\, dx' \stackrel{\text{def}}{=} (\mathcal{K}f)(x), \qquad 0 < x < 1. \tag{1.1}$$

This is a one-dimensional version of a model that occurs in two-dimensional optical imaging and is discussed in more detail in Chapter 5. In this application, f represents light source intensity as a function of spatial position, and g represents image intensity. The kernel k characterizes blurring effects that occur during image formation. A kernel that models the long-time average effects of atmospheric turbulence on light propagation is the Gaussian [7, 99]. Its one-dimensional version is

$$k(x) = C\, \exp(-x^2/2\gamma^2), \tag{1.2}$$

where C and γ are positive parameters.

The direct problem, or forward problem, associated with the model equation (1.1) is the following: Given the source f and the kernel k, determine the blurred image g. Figure 1.1 shows the blurred image corresponding to a piecewise smooth source. Since k is a smooth

function, the accurate approximation of $g = \mathcal{K}f$ using standard numerical quadrature is straightforward.

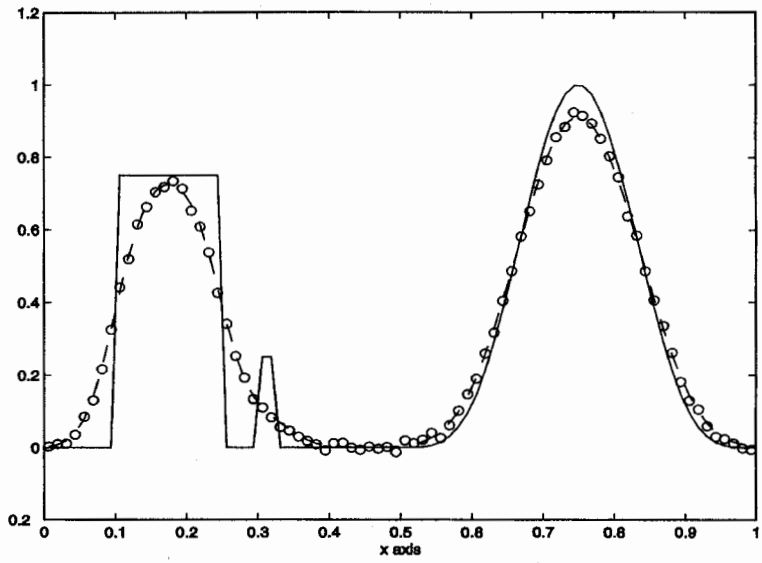


Figure 1.1. *One-dimensional image data. The source function f is represented by the solid line, the blurred image $g = \mathcal{K}f$ is represented by the dashed line, and the discrete noisy data* $\mathbf{d}$ *is represented by circles. The data were generated according to* (1.1)–(1.4) *with parameters $\gamma = 0.05$ and $C = 1/(\gamma\sqrt{2\pi})$. Midpoint quadrature was used to approximate integrals.*

An associated inverse problem of practical interest is as follows: Given the kernel k and the blurred image g, determine the source f. At first glance, the approximate solution to this inverse problem seems straightforward. One may simply discretize equation (1.1), e.g., using collocation in the independent variable x and quadrature in x', to obtain a discrete linear system $K\mathbf{f} = \mathbf{d}$. For instance, if midpoint quadrature is applied, then K has entries

$$[K]_{ij} = h\ C\ \exp\left(-\frac{((i-j)h)^2}{2\gamma^2}\right), \qquad 1 \le i, j \le n, \tag{1.3}$$

where $h = 1/n$. If the matrix K is nonsingular, one may then compute the discrete approximation $K^{-1}\mathbf{d}$ to f. To obtain an accurate quadrature approximation, n must be relatively large. Unfortunately, the matrix K becomes increasingly ill-conditioned as n becomes large, so errors in $\mathbf{d}$ may be greatly amplified. Certain errors, like those due to quadrature, can be controlled. Others, like the noise in the image recording device, cannot be controlled in a practical setting. Consequently, this straightforward solution approach is likely to fail.

1.2 Regularization by Filtering

Despite ill-conditioning, one can extract some useful information from the discrete linear system $K\mathbf{f} = \mathbf{d}$. To simplify the presentation, consider a discrete data model

$$\mathbf{d} = K\mathbf{f}_{\text{true}} + \eta \tag{1.4}$$

ical quadrature is

resented by the solid
rete noisy data **d** *is*
rameters $\gamma = 0.05$

Given the kernel k
roximate solution
ze equation (1.1),
o obtain a discrete
en K has entries

e discrete approx-
must be relatively
s n becomes large,
quadrature, can be
be controlled in a
likely to fail.

the discrete linear
odel

with

$$\delta \stackrel{\text{def}}{=} ||\eta|| > 0. \tag{1.5}$$

Here $||\cdot||$ denotes standard Euclidean norm, $\mathbf{f}_{\text{true}}$ represents the true discretized source, and η represents error in the data. The parameter δ is called the error level. For further simplicity, assume K is an invertible, real-valued matrix. It then has a singular value decomposition (SVD) [46],

$$K = U\ \text{diag}(s_i)\ V^T, \tag{1.6}$$

with strictly positive decreasing singular values s_i. The SVD and its connection with inverse problems are discussed in more detail in the next chapter; cf. Definition 2.15, and see [58]. At this point we require the following facts: The column vectors $\mathbf{v}_i$ of V, which are called right singular vectors, and the column vectors $\mathbf{u}_i$ of U, which are the left singular vectors, satisfy

$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}, \qquad \mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}, \tag{1.7}$$

$$K\mathbf{v}_i = s_i \mathbf{u}_i, \qquad K^T \mathbf{u}_i = s_i \mathbf{v}_i. \tag{1.8}$$

Here δ_{ij} denotes the Kronecker delta (equation (2.2)), and $U^T = U^{-1}$ and $V^T = V^{-1}$. Note that if K is symmetric and positive definite, then the singular values s_i are the eigenvalues of K, and $U = V$ has columns consisting of orthonormalized eigenvectors. The singular values and vectors for our discretized one-dimensional imaging problem are represented graphically in Figure 1.2.

Using properties (1.7)–(1.8),

$$K^{-1}\mathbf{d} \;=\; V\ \text{diag}(s_i^{-1})\ U^T\mathbf{d} \;=\; \mathbf{f}_{\text{true}} + \sum_{i=1}^{n} s_i^{-1}(\mathbf{u}_i^T \eta)\mathbf{v}_i. \tag{1.9}$$

Instability arises due to division by small singular values. One way to overcome this instability is to modify the s_i^{-1}'s in (1.9), e.g., by multiplying them by a regularizing filter function $w_\alpha(s_i^2)$ for which the product $w_\alpha(s^2)s^{-1} \to 0$ as $s \to 0$. This filters out singular components of $K^{-1}\mathbf{d}$ corresponding to small singular values and yields an approximation to $\mathbf{f}_{\text{true}}$ with a representation

$$\begin{aligned} \mathbf{f}_\alpha &= V\ \text{diag}(w_\alpha(s_i^2)s_i^{-1})\ U^T\mathbf{d} \\ &= \sum_{i=1}^{n} w_\alpha(s_i^2)s_i^{-1}(\mathbf{u}_i^T\mathbf{d})\mathbf{v}_i. \end{aligned} \tag{1.10}$$

To obtain some degree of accuracy, one must retain singular components corresponding to large singular values. This is done by taking $w_\alpha(s^2) \approx 1$ for large values of s^2. An example of such a filter function is

$$w_\alpha(s^2) = \begin{cases} 1 & \text{if} \quad s^2 > \alpha, \\ 0 & \text{if} \quad s^2 \le \alpha. \end{cases} \tag{1.11}$$

The approximation (1.10) then takes the form

$$\mathbf{f}_\alpha = \sum_{s_i^2 > \alpha} s_i^{-1}(\mathbf{u}_i^T\mathbf{d})\mathbf{v}_i \tag{1.12}$$

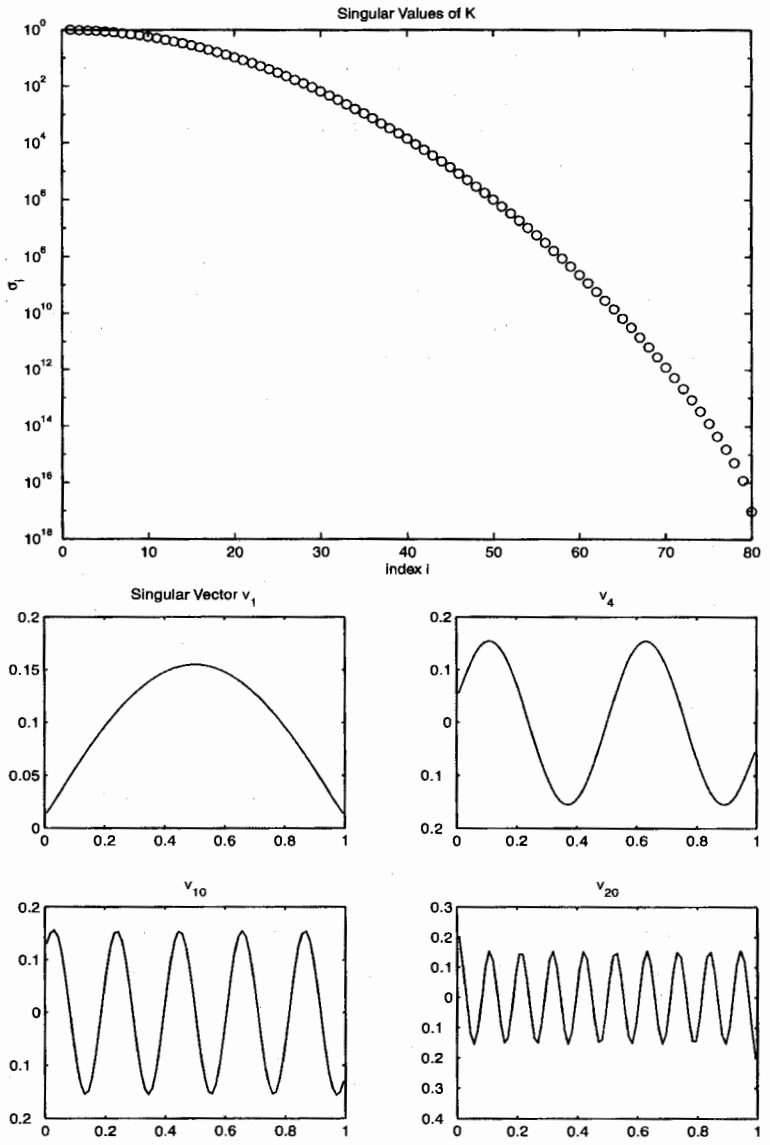


Figure 1.2. *SVD of the matrix K having ijth entry $k(x_i - x_j)\,h$, where $h = 1/n$, with $n = 80$ and $x_i = (i + 1/2)h$, $i = 1, \ldots, n$. The top plot shows the distribution of the singular values s_i of K. The subplots below it show a few of the corresponding singular vectors $\mathbf{v}_i$. (K is symmetric, so the left and right singular vectors are the same.) The components $[\mathbf{v}_i]_j$ are plotted against the x_j's. At the middle left is the singular vector $\mathbf{v}_1$ corresponding to the largest singular value s_1 of K. At the middle right is the singular vector $\mathbf{v}_4$ corresponding to the fourth largest singular value s_4. The bottom left and bottom right subplots show, respectively, the singular vectors corresponding to the* 10th *and* 20th *largest singular values.*

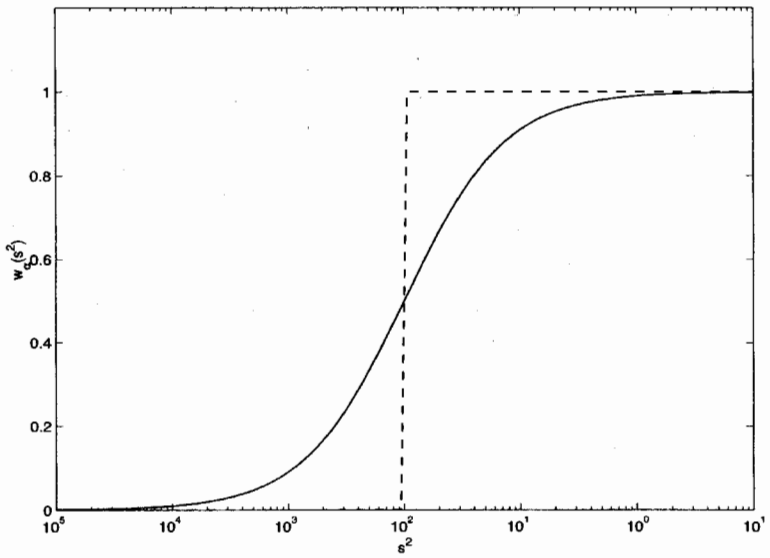


Figure 1.3. *Semilog plots of filter functions w_α corresponding to TSVD regularization (dashed line) and Tikhonov regularization (solid line) as functions of squared singular values s^2. The value of the regularization parameter is $\alpha = 10^{-2}$.*

and is known as the truncated SVD (TSVD) solution to $K\mathbf{f} = \mathbf{d}$. Another example is the Tikhonov filter function:

$$w_\alpha(s^2) = \frac{s^2}{s^2 + \alpha}. \tag{1.13}$$

The corresponding regularized approximation (1.10) can be expressed as

$$\mathbf{f}_\alpha = \sum_{i=1}^{n} \frac{s_i(\mathbf{u}_i^T \mathbf{d})}{s_i^2 + \alpha} \mathbf{v}_i \tag{1.14}$$

$$= (K^T K + \alpha I)^{-1} K^T \mathbf{d}. \tag{1.15}$$

Equation (1.15) is a consequence of (1.6)–(1.8). See Exercise 1.3. This yields a technique known as Tikhonov(–Phillips) regularization [106, 93].

The α in (1.11) and (1.13) is called a regularization parameter. In (1.11) this parameter determines the cut-off, or threshold, level for the TSVD filter. From the plots of the filter functions in Figure 1.3, it can be seen that the regularization parameter for Tikhonov regularization in (1.13) plays a similar role. Figure 1.4 illustrates how the TSVD solution $\mathbf{f}_\alpha$ varies with α. Similar behavior can be observed when Tikhonov regularization is applied. When α is very small, filtering of the noise is inadequate and $\mathbf{f}_\alpha$ is highly oscillatory. On the other hand, when α is large, the noise components are filtered out. Unfortunately, most components of the solution are also filtered out, and $\mathbf{f}_\alpha$ is overly smooth.

An obvious question arises: Can the regularization parameter be selected to guarantee convergence as the error level goes to zero? The answer to this question lies in the following analysis.

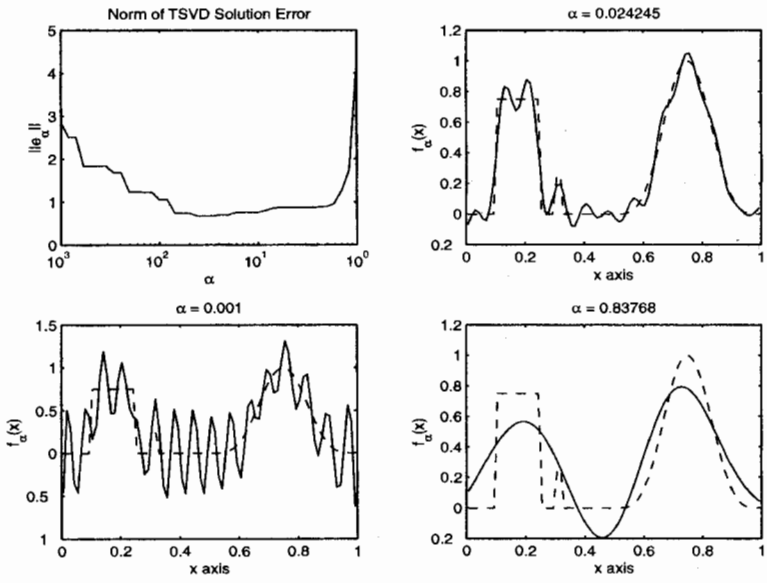


Figure 1.4. *TSVD regularized solutions. The upper left subplot shows the norm of the solution error, $||\mathbf{f}_\alpha - \mathbf{f}_{\text{true}}||$, versus the regularization parameter α. In the upper right subplot, the solid line represents the regularized solution $\mathbf{f}_\alpha$ for $\alpha = 0.0242$. This value of α minimizes the solution error norm. The lower left and lower right subplots, respectively, show $\mathbf{f}_\alpha$ for $\alpha = 0.001$ and $\alpha = 0.50119$. The dashed curves represent the true solution $\mathbf{f}_{\text{true}}$.*

1.2.1 A Deterministic Error Analysis

The right-hand side of (1.10) defines a linear regularization operator, which we denote by R_α. Hence $f_\alpha = R_\alpha \mathbf{d}$. From (1.4), the regularized solution error is given by

$$\begin{aligned} \mathbf{e}_\alpha &\stackrel{\text{def}}{=} \mathbf{f}_\alpha - \mathbf{f}_{\text{true}} \\ &= \mathbf{e}_\alpha^{\text{trunc}} + \mathbf{e}_\alpha^{\text{noise}}, \end{aligned} \tag{1.16}$$

where

$$\begin{aligned} \mathbf{e}_\alpha^{\text{trunc}} &\stackrel{\text{def}}{=} R_\alpha K \mathbf{f}_{\text{true}} - \mathbf{f}_{\text{true}} \\ &= \sum_{i=1}^{n} (w_\alpha(s_i^2) - 1)(\mathbf{v}_i^T \mathbf{f}_{\text{true}})\mathbf{v}_i \end{aligned} \tag{1.17}$$

and

$$\begin{aligned} \mathbf{e}_\alpha^{\text{noise}} &\stackrel{\text{def}}{=} R_\alpha \boldsymbol{\eta} \\ &= \sum_{i=1}^{n} w_\alpha(s_i^2) s_i^{-1} (\mathbf{u}_i^T \boldsymbol{\eta})\mathbf{v}_i. \end{aligned} \tag{1.18}$$

We call $\mathbf{e}_\alpha^{\text{trunc}}$ the solution truncation error due to regularization. It quantifies the loss of information due to the regularizing filter. The term $\mathbf{e}_\alpha^{\text{noise}}$ is called the noise amplification error. We will show that for both the TSVD filter (1.11) and the Tikhonov filter (1.13), the

ws the norm of the
ht subplot, the solid
:s the solution error
l and $\alpha = 0.50119$.

ıich we denote by
by

antifies the loss of
ıoise amplification
ov filter (1.13), the

regularization parameter α can be selected in a manner that guarantees that both these errors converge to zero as the error level $\delta \to 0$.

We first consider the truncation error. For both the TSVD filter (1.11) and the Tikhonov filter (1.13), for any fixed $s > 0$,

$$w_\alpha(s^2) \to 1 \quad \text{as} \quad \alpha \to 0, \tag{1.19}$$

and hence from (1.17),

$$\mathbf{e}_\alpha^{\text{trunc}} \to 0 \quad \text{whenever} \quad \alpha \to 0. \tag{1.20}$$

To deal with the noise amplification error, one can show (see Exercise 1.5) that both filter functions satisfy

$$w_\alpha(s^2)\, s^{-1} \leq \alpha^{-1/2}. \tag{1.21}$$

Consequently, from (1.18) and (1.5),

$$||\mathbf{e}_\alpha^{\text{noise}}|| \leq \alpha^{-1/2}\delta. \tag{1.22}$$

One can obtain $||\mathbf{e}_\alpha^{\text{noise}}|| \to 0$ as $\delta \to 0$ by choosing $\alpha = \delta^p$ with $p < 2$. If in addition $p > 0$, then (1.20) also holds. Since $\mathbf{e}_\alpha = \mathbf{e}_\alpha^{\text{trunc}} + \mathbf{e}_\alpha^{\text{noise}}$, the regularization parameter choice

$$\alpha = \delta^p, \quad 0 < p < 2, \tag{1.23}$$

guarantees that

$$\mathbf{e}_\alpha \to 0 \quad \text{as} \quad \delta \to 0 \tag{1.24}$$

when either TSVD or Tikhonov regularization is applied to data (1.4). A regularization method together with a parameter selection rule like (1.23) are called convergent if (1.24) holds.

1.2.2 Rates of Convergence

Consider the TSVD filter (1.11), and assume $\delta \geq s_n^2$, the square of the smallest singular value. If this assumption doesn't hold, then noise amplification is tolerable even without regularization. To obtain a convergence rate for the solution error, one needs bounds on the truncation error. Assume

$$\mathbf{f}_{\text{true}} = K^T\mathbf{z}, \qquad \mathbf{z} \in \mathbb{R}^n. \tag{1.25}$$

This is an example of a so-called source condition [35] or range condition. Since $|(K^T\mathbf{z})^T\mathbf{v}_i| = |\mathbf{z}^T K\mathbf{v}_i| = s_i|\mathbf{z}^T\mathbf{u}_i|$, one obtains

$$\begin{aligned} ||\mathbf{e}_\alpha^{\text{trunc}}||^2 &= \sum_{i=1}^{n} (w_\alpha(s_i^2) - 1)^2 s_i^2 |\mathbf{z}^T\mathbf{u}_i|^2 \\ &\leq \max_{1\leq i\leq n} (w_\alpha(s_i^2) - 1)^2 s_i^2 ||\mathbf{z}||^2 \\ &\leq \alpha||\mathbf{z}||^2. \end{aligned} \tag{1.26}$$

See Exercise 1.7. This bound is sharp in the sense that the inequality becomes an equality for certain combinations of α and the s_i's. See Exercise 1.8. Combining equation (1.26) with (1.22) gives

$$||\mathbf{e}_\alpha|| \leq \alpha^{1/2}||\mathbf{z}|| + \alpha^{-1/2}\delta. \tag{1.27}$$

The right-hand side is minimized by taking

$$\alpha = \frac{\delta}{||\mathbf{z}||}. \tag{1.28}$$

This yields

$$||\mathbf{e}_\alpha|| \leq 2\,||\mathbf{z}||^{1/2}\,\delta^{1/2}. \tag{1.29}$$

A regularization method together with a parameter choice rule for which $||\mathbf{e}_\alpha|| = \mathcal{O}(\sqrt{\delta})$ as $\delta \to 0$ is called order optimal given the information $\mathbf{f}_{\text{true}} \in \text{Range}(K^T)$. We have established that TSVD regularization with the parameter choice (1.29) is order optimal given $\mathbf{f}_{\text{true}} \in \text{Range}(K^T)$.

In a continuous setting, $f_{\text{true}} \in \text{Range}(K^T)$ is a condition on the smoothness of f_{true}. This conclusion carries over to the discrete case, although with less conciseness. From Figure 1.2, singular vectors corresponding to very small singular values are highly oscillatory. If $||\mathbf{z}||$ is not large, then either $\mathbf{f}_{\text{true}}$ is very small or the singular expansion of $\mathbf{f}_{\text{true}}$ is dominated by singular components corresponding to large singular values, and these components are smoothly varying.

Equation (1.28) is an a priori regularization parameter choice rule. It requires prior information about both the data noise level δ and the true solution, through assumption (1.25) and quantity $||\mathbf{z}||$. In practice, such prior information about the true solution is unlikely to be available.

1.2.3 A Posteriori Regularization Parameter Selection

A parameter choice rule is called a posteriori if the selection of the regularization parameter depends on the data but not on prior information about the solution. One such rule is the discrepancy principle due to Morozov [86], where in the case of either TSVD or Tikhonov regularization one selects the largest value of the regularization parameter α for which

$$||K\mathbf{f}_\alpha - \mathbf{d}|| \leq \delta. \tag{1.30}$$

It can be shown that both TSVD and Tikhonov regularization with the discrepancy principle parameter choice rule are convergent, and, assuming the source condition (1.25), both are order optimal [48, 35, 70].

What follows is a finite-dimensional version of the analysis for the discrepancy principle applied to Tikhonov regularization. To simplify notation, define the data discrepancy functional

$$D(\alpha) = ||K\mathbf{f}_\alpha - \mathbf{d}||.$$

We first establish conditions under which there exists a value of the regularization parameter for which $D(\alpha) = \delta$. From the representation (1.15) and the SVD (1.6)–(1.8),

$$\begin{aligned} D^2(\alpha) &= ||(I - K(K^TK + \alpha I)^{-1}K^T)\mathbf{d}||^2 \\ &= \sum_{i=1}^{n} \left(1 - \frac{s_i^2}{s_i^2 + \alpha}\right)^2 (\mathbf{u}_i^T\mathbf{d})^2. \end{aligned} \tag{1.31}$$

Thus $D(\alpha)$ is continuous and strictly increasing with $D(0) = 0$ and $D(\alpha) \to ||\mathbf{d}||$ as $\alpha \to \infty$. Thus $D(\alpha) = \delta$ has a unique solution provided that the data noise level is less than the data norm,

(1.32) $$\delta < ||\mathbf{d}||.$$

Assume that condition (1.32) holds, and let $\alpha(\delta)$ denote the unique solution to $D(\alpha) = \delta$. Then

(1.33) $$||\mathbf{f}_{\alpha(\delta)}|| \leq ||\mathbf{f}_{\text{true}}||.$$

To verify this,

$$\begin{aligned} \delta^2 + \alpha||\mathbf{f}_{\alpha(\delta)}||^2 &= ||K\mathbf{f}_{\alpha(\delta)} - \mathbf{d}||^2 + \alpha||\mathbf{f}_{\alpha(\delta)}||^2 \\ &\leq ||K\mathbf{f}_{\text{true}} - \mathbf{d}||^2 + \alpha||\mathbf{f}_{\text{true}}||^2 \\ &= \delta^2 + \alpha||\mathbf{f}_{\text{true}}||^2. \end{aligned}$$

The inequality follows from the variational representation (1.34) in section 1.3 (see Exercise 1.13), while the last equality follows from the data model (1.4)–(1.5).

Finally, we establish order optimality. Following the proof of Theorem 3.3 in [48],

$$\begin{aligned} ||\mathbf{f}_{\alpha(\delta)} - \mathbf{f}_{\text{true}}||^2 &= ||\mathbf{f}_{\alpha(\delta)}||^2 - 2\mathbf{f}_{\alpha(\delta)}^T\mathbf{f}_{\text{true}} + ||\mathbf{f}_{\text{true}}||^2 \\ &\leq 2||\mathbf{f}_{\text{true}}||^2 - 2\mathbf{f}_{\alpha(\delta)}^T\mathbf{f}_{\text{true}}, \quad \text{by} \quad (1.33) \\ &= 2(\mathbf{f}_{\text{true}} - \mathbf{f}_{\alpha(\delta)})^T K^T\mathbf{z}, \quad \text{by} \quad (1.25) \\ &= 2(K\mathbf{f}_{\text{true}} - \mathbf{d} + \mathbf{d} - K\mathbf{f}_{\alpha(\delta)})^T\mathbf{z} \\ &\leq 4\delta\ ||\mathbf{z}||. \end{aligned}$$

The last inequality follows from the Cauchy–Schwarz inequality, the triangle inequality, and the fact that $D(\alpha(\delta)) = ||K\mathbf{f}_{\text{true}} - \mathbf{d}|| = \delta$.

1.3 Variational Regularization Methods

For very large ill-conditioned systems, it is often impractical to directly implement regularization by filtering, since the representation (1.10) requires the SVD of a large matrix. However, the Tikhonov solution (1.14) has an alternate variational representation,

(1.34) $$\mathbf{f}_\alpha = \arg\min_{\mathbf{f}\in\mathbb{R}^n} ||K\mathbf{f} - \mathbf{d}||^2 + \alpha||\mathbf{f}||^2,$$

which may be easier to compute. This representation may have other advantages. For instance, in optics the source intensity f is nonnegative. Nonnegativity can be imposed as a constraint in (1.34). Moreover, the least squares term $||K\mathbf{f} - \mathbf{d}||^2$ can be replaced by other fit-to-data functionals. See section 4.2 for specific examples. The term $||\mathbf{f}||^2$ in (1.34) is called a penalty functional. Other penalty functionals can be used to incorporate a priori information. An example is the discrete one-dimensional total variation

$$\begin{aligned} TV(\mathbf{f}) &= \sum_{i=1}^{n-1} |f_{i+1} - f_i| \\ &= \sum_{i=1}^{n-1} \left| \frac{f_{i+1} - f_i}{\Delta x} \right| \Delta x. \end{aligned} \quad (1.35)$$

This penalizes highly oscillatory solutions while allowing jumps in the regularized solution. Note that for smooth f, the sum in (1.35) approximates the L^1 norm of the derivative, a nonquadratic function of f. Figure 1.5 illustrates that the reconstructions obtained with total variation can be qualitatively quite different from those obtained with methods like TSVD, (see Figure 1.4). Unfortunately, total variation reconstructions are much more difficult to compute. See Chapter 8 for further details.

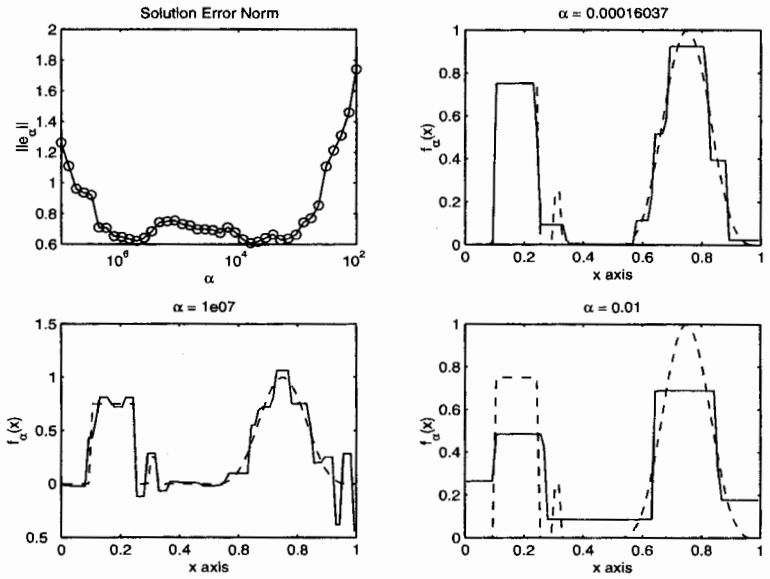


Figure 1.5. *One-dimensional total variation regularized solutions. The upper left subplot shows the norm of the solution error, $||\mathbf{f}_\alpha - \mathbf{f}_{\text{true}}||$, versus the regularization parameter α. In the upper right subplot, the solid line represents the regularized solution $\mathbf{f}_\alpha$ for $\alpha = 1.604 \times 10^{-4}$. This value of α minimizes the solution error norm. The lower left and lower right subplots, respectively, show $\mathbf{f}_\alpha$ for $\alpha = 10^{-6}$ and $\alpha = 1.0$. The dashed curves represent the true solution $\mathbf{f}_{\text{true}}$.*

1.4 Iterative Regularization Methods

We illustrate the concept of iterative regularization with a simple example. Consider the scaled least squares fit-to-data functional

$$J(\mathbf{f}) = \frac{1}{2}||K\mathbf{f} - \mathbf{d}||^2. \tag{1.36}$$

This has as its gradient grad $J(\mathbf{f}) = K^T(K\mathbf{f} - \mathbf{d})$. Consider the iteration

$$\mathbf{f}_{\nu+1} = \mathbf{f}_\nu - \tau \text{ grad } J(\mathbf{f}_\nu), \qquad \nu = 0, 1, \ldots. \tag{1.37}$$

If at each iteration ν the scalar τ is chosen to minimize $\tilde{J}(\tau) = J(\mathbf{f}_\nu - \tau \text{ grad } J(\mathbf{f}_\nu))$, then one obtains the steepest descent method. See section 3.1 for details. If one fixes τ with $0 < \tau < 1/||K||^2$, one obtains a method known as Landweber iteration [72]. With either choice of τ, if one takes the initial guess $\mathbf{f}_0 = \mathbf{0}$ and one assumes that K is invertible, one can show that the iterates $\mathbf{f}_\nu$ converge to $\mathbf{f}_* = K^{-1}\mathbf{d}$. This is not desirable if error is present in the data. From the plot of solution error norm versus iteration count ν shown in Figure 1.6, we see that the iteration count appears to play the role of a regularization parameter. Very small values of ν yield overly smooth approximate solutions. On the other hand, as ν becomes large, the reconstructions become highly oscillatory. This phenomenon is called semiconvergence.

To explain this phenomenon, one can show that when $\mathbf{f}_0 = \mathbf{0}$, $\mathbf{f}_\nu$ has the representation (1.10) with the filter function

$$w_\nu(s^2) = 1 - (1 - \tau s^2)^\nu. \tag{1.38}$$

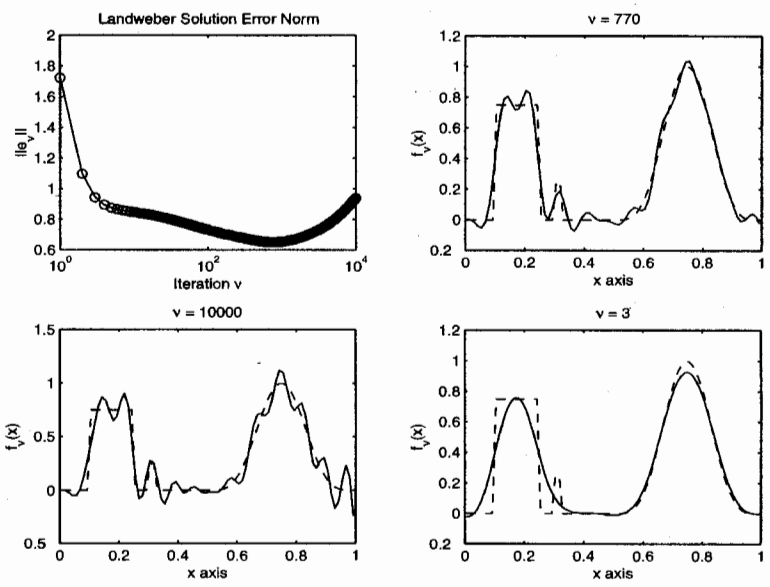


Figure 1.6. *Results for Landweber iteration. The upper left subplot shows the norm of the solution error, $\|\mathbf{f}_\nu - \mathbf{f}_{\text{true}}\|$, versus the iteration count ν. In the upper right subplot, the solid line represents the regularized solution $\mathbf{f}_\nu$ for $\nu = 766$. This value of ν minimizes the solution error norm. The lower left and lower right subplots, respectively, show $\mathbf{f}_\nu$ for $\nu = 10^4$ and $\nu = 3$. The dashed curves represent the true solution $\mathbf{f}_{\text{true}}$.*

See Exercise 1.15. The iteration count ν is indeed a regularization parameter. One can show [35, section 6.1] that the discrepancy principle is order optimal for Landweber iteration. Unfortunately, the method tends to require many iterations to generate accurate regularized solutions, thereby limiting its practical use.

Exercises

1.1. Verify that when midpoint quadrature is applied to the integral operator $\mathcal{K}$ in (1.1)–(1.2), one obtains matrix K in (1.3).

1.2. Use properties (1.7)–(1.8) to obtain (1.9).

1.3. Use the decomposition (1.6)–(1.8) to confirm the equality (1.14)–(1.15).

1.4. Confirm equations (1.16)–(1.18) using (1.10) and (1.4).

1.5. Verify that equation (1.21) is satisfied for both the TSVD filter function (1.11) and the Tikhonov filter function (1.13).

1.6. Using (1.18) and (1.5), show that equation (1.22) holds.

1.7. Confirm the inequality (1.26).

1.8. Show that the inequality (1.26) is sharp. To do this, give the vector $\mathbf{z}$ for which equality holds.

1.9. Show that the right-hand side of (1.27) is minimized with the choice (1.28). Then confirm (1.29).

1.10. Mimic the analysis of section 1.2.2 to show that TSVD with $\alpha \sim \delta^{2/3}$ is order optimal for $\mathbf{f}_{\text{true}} \in \text{Range}(K^T K)$.

1.11. Show that for the continuous operator (1.1) the source condition $f_{\text{true}} = K^* z$, $z \in L^2(0, 1)$, implies that f_{true} is smooth.

1.12. Confirm that the operator representation (1.15) is equivalent to the Tikhonov filter representation (1.10), (1.13). To do this, use properties of the SVD to verify that

$$(K^T K + \alpha I)^{-1} K^T \mathbf{d} = V \ \text{diag}(s_i/(s_i^2 + \alpha))\ U^T \mathbf{d}.$$

1.13. Confirm that the variational representation (1.34) is equivalent to the Tikhonov filter representation (1.10), (1.13). Verify that

$$\begin{aligned} ||K\mathbf{f} - \mathbf{d}||^2 + \alpha||\mathbf{f}||^2 &= \mathbf{f}^T(K^T K + \alpha I)\mathbf{f} - 2\mathbf{f}^T K^T \mathbf{d} + ||\mathbf{d}||^2 \\ &= \tilde{\mathbf{f}}^T \text{diag}(s_i^2 + \alpha)\tilde{\mathbf{f}} - 2\tilde{\mathbf{f}}^T \text{diag}(s_i)\ \tilde{\mathbf{d}} + ||\mathbf{d}||^2, \end{aligned}$$

where $\tilde{\mathbf{f}} = V^T \mathbf{f}$ and $\tilde{\mathbf{d}} = U^T \mathbf{d}$. Then minimize with respect to $\tilde{\mathbf{f}}$ and take $\mathbf{f} = V\tilde{\mathbf{f}}$.

1.14. Numerically implement standard Tikhonov regularization for the test problem presented in Figure 1.1. The true solution is given by

$$f_{\text{true}}(x) = \begin{cases} 0.75, & 0.1 < x < 0.25, \\ 0.25, & 0.3 < x < 0.32, \\ \sin^4(2\pi x), & 0.5 < x < 1, \\ 0 & \text{otherwise.} \end{cases}$$

1.15. Confirm that Landweber iteration yields a representation (1.10) with filter function (1.38). To do this, show that $\mathbf{f}_\nu = \sum_{j=0}^{\nu-1} G^j \mathbf{b}$, where $G = I - \tau K^T K$ and $\mathbf{b} = \tau K^T \mathbf{d}$. Then apply the SVD.

1.16. Generate plots of the Landweber filter function $w_\nu(s^2)$ in equation (1.38). Let the independent variable s^2 range between 0 and 1. How does the behavior of $w_\nu(s^2)$ change as ν and τ vary?

)tain a matrix S for

) gives a probability

bject to the equality

$\lambda = r$. Verify also

Chapter 5
Image Deblurring

In this chapter we consider a two-dimensional analogue of (1.1):

$$g(x, y) = \int_0^1 \int_0^1 k(x - x', y - y')\, f(x', y')\, dx'\, dy'. \tag{5.1}$$

In image reconstruction, the estimation of f from observations of g is referred to as the two-dimensional image deblurring problem [64]. An imaging application is presented in section 5.1. Since the integral operator has convolution form, (5.1) is also known as the two-dimensional deconvolution problem.

The kernel function k in (5.1) is typically smooth, so from Example 2.13, the corresponding integral operator is compact. Hence by Theorem 2.14, the deblurring problem is ill-posed and some form of regularization must be applied to accurately reconstruct f from noisy data. Numerical implementation is complicated by the fact that the discrete systems arising from equation (5.1) may have a very large number of unknowns. This complication can be alleviated by taking advantage of convolution structure. In particular, certain discretizations of (5.1) give rise to linear systems with Toeplitz block structure. Such systems can be efficiently solved using conjugate gradient iteration with preconditioners based on the fast Fourier transform (FFT). Computational methods are described in detail in sections 5.2 and 5.3.

5.1 A Mathematical Model for Image Blurring

A very general model for the blurring of images [7] is

$$g(x, y) = \int\int_{\mathbb{R}^2} k(x, x', y, y')\, f(x', y')\, dx'\, dy'. \tag{5.2}$$

In optics, f is called the light source, or object. The kernel function k is known as the point spread function (PSF), and g is called the (blurred) continuous image. Equation (5.2) can be used to model the diffraction of light from the source as it propagates through a medium like the atmosphere [99]. It can also model distortion due to imperfections in optical devices like telescopes and microscopes. See [7] for other applications.

The continuous image g represents an energy density, with units of energy per unit area or, equivalently, number of photons per unit area. The image is often recorded with a device known as a CCD camera. This consists of an array of disjoint rectangles called pixels, Ω_{ij},

$0 \le i \le n_x - 1, 0 \le j \le n_y - 1$, onto which the photons fall and are counted. The energy falling on an individual array element is then given by

$$g_{ij} = \int\int_{\Omega_{ij}} g(x, y)\, dx\, dy. \tag{5.3}$$

A stochastic model for the data recorded by the ijth pixel of a CCD array is given in the notation of Chapter 4 by

$$D_{ij} \sim \text{Poisson}(g_{ij}) + \text{Normal}(0, \sigma^2). \tag{5.4}$$

The Poisson component models the photon count, while the additive Gaussian term accounts for background noise in the recording electronics [99]. We denote a realization of the random variable D_{ij} by d_{ij}. The $n_x \times n_y$ array d, whose components are the d_{ij}'s, is called the (noisy, blurred) discrete image. For each index pair (i, j), d_{ij} is a realization of a Gaussian random variable with zero mean and variance σ^2 added to a realization of a Poisson random variable with mean and variance g_{ij}; see Examples 4.13 and 4.14. These random variables are assumed to be independent of each other and independent of the random variables corresponding to the other pixels.

A fully discrete model may be obtained by truncating the region of integration in (5.2) to be the union of the Ω_{ij}'s and then applying midpoint quadrature to both (5.2) and (5.3). Assume that each Ω_{ij} has area $\Delta x \times \Delta y$, and let (x_i, y_j) denote the midpoint. Then

$$g_{ij} = \sum_{\mu=0}^{n_x-1} \sum_{\nu=0}^{n_y-1} k(x_i, x_\mu, y_j, y_\nu)\, f(x_\mu, y_\nu)\, \Delta x\, \Delta y + \epsilon_{ij}^{\text{quad}}, \tag{5.5}$$

where $\epsilon_{ij}^{\text{quad}}$ denotes quadrature error. Combining (5.4) and (5.5),

$$d_{ij} = \sum_{\mu=0}^{n_x-1} \sum_{\nu=0}^{n_y-1} t_{i,\mu,j,\nu} f_{\mu,\nu} + \eta_{ij}, \tag{5.6}$$

where now $f_{\mu,\nu} = f(x_\mu, y_\nu)$, the term η_{ij} incorporates the various stochastic error realizations and quadrature errors, and $t_{i,\mu,j,\nu} = k(x_i, x_\mu, y_j, y_\nu)\, \Delta x\, \Delta y$. We refer to the array t as the discrete PSF.

The blurring process is sometimes assumed to be invariant under spatial translation. This means that the PSF can be represented as a function of two variables, rather than four variables,

$$k(x, x', y, y') = k(x - x', y - y'), \tag{5.7}$$

and equation (5.2) reduces to (5.1). This representation greatly simplifies computations. Since the integral in (5.1) then has convolution form, given the source $f = f(x, y)$ and the PSF $k = k(x, y)$, one can in principle compute the continuous image g using the convolution theorem,

$$g = \mathcal{F}^{-1}\{\mathcal{F}\{k\}\, \mathcal{F}\{f\}\}. \tag{5.8}$$

Here the continuous Fourier transform of a (possibly complex-valued) function f defined on $\mathbb{R}^d$ ($d = 2$ for two-dimensional imaging) is given by

$$\mathcal{F}\{f\}(\boldsymbol{\omega}) = \int_{\mathbb{R}^d} f(\mathbf{x})\, e^{-i2\pi \mathbf{x}^T \boldsymbol{\omega}}\, d\mathbf{x}, \qquad \boldsymbol{\omega} \in \mathbb{R}^d, \tag{5.9}$$

with $\hat{\imath} \stackrel{\text{def}}{=} \sqrt{-1}$. The inverse continuous Fourier transform is given by

$$\mathcal{F}^{-1}\{g\}(\mathbf{x}) = \int_{\mathbb{R}^d} g(\boldsymbol{\omega})\, e^{\hat{\imath} 2\pi \mathbf{x}^T \boldsymbol{\omega}}\, d\omega, \qquad \mathbf{x} \in \mathbb{R}^d. \tag{5.10}$$

One can formally derive from (5.8) the Fourier inversion formula,

$$f = \mathcal{F}^{-1}\left\{\frac{\mathcal{F}\{g\}}{\mathcal{F}\{k\}}\right\}. \tag{5.11}$$

If $\mathcal{F}\{k\}$ takes on zero values, this formula is not valid. If it takes on small nonzero values, this reconstructed f is unstable with respect to perturbations in the data g. These situations correspond to violations of conditions (ii) and (iii) in Definition 2.7 of well-posedness.

Discrete computations are also greatly simplified by the representation (5.7). In equation (5.6), the discrete PSF can be represented as a two-dimensional array, rather than as a four-dimensional array,

$$t_{i,\mu,j,\nu} = t_{i-\mu,j-\nu}, \qquad 0 \le i, \mu \le n_x - 1,\ 0 \le j, \nu \le n_y - 1. \tag{5.12}$$

Equation (5.6) then reduces to

$$d_{ij} = \sum_{\mu=0}^{n_x-1} \sum_{\nu=0}^{n_y-1} t_{i-\mu,j-\nu} f_{\mu,\nu} + \eta_{ij} \tag{5.13}$$

with

$$t_{ij} = k(i\Delta x, j\Delta y)\, \Delta x \Delta y. \tag{5.14}$$

The discrete convolution product in (5.13) defines a linear operator. A discrete analogue of the continuous Fourier transform can be used to efficiently compute regularized solutions. Details are given in section 5.2.

5.1.1 A Two-Dimensional Test Problem

The two-dimensional test problem arises in atmospheric optics, an application described in detail in [99]. The data, a simulated image of a satellite in earth orbit viewed with a ground-based telescope, was generated according to the model (5.13). In this model, the continuous PSF takes the form

$$k = |\mathcal{F}^{-1}\{Ae^{i\phi}\}|^2. \tag{5.15}$$

Here A is called the aperture function and ϕ is the phase. The phase represents distortions in a planar wavefront, emanating from a point source at infinity, due to propagation through an optically thin layer of material (in this case, the atmosphere) with a variable index of refraction. The aperture function represents the region in the plane over which light is collected. For a large reflecting telescope, the aperture function is typically the indicator function for an annulus. Both the phase and the support of the aperture function can be seen in the top plot in Figure 5.1. The middle plot shows the corresponding PSF. Figure 5.2 shows a simulated light source, along with the corresponding blurred, noisy image. This was obtained by convolving the PSF with the source and then adding noise to the resulting blurred image.

The bottom plot in Figure 5.1 shows the power spectrum, $|\mathcal{F}\{k\}|^2$, of the PSF. Several phenomena can be seen from the power spectrum. First, its support (i.e., the region in

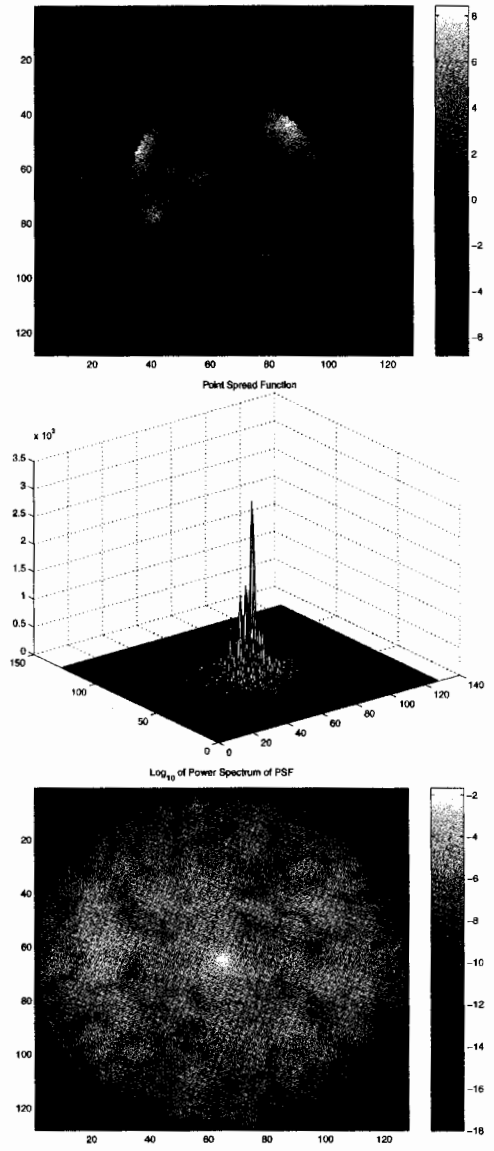


Figure 5.1. *Atmospheric image blurring operator. At the top is a gray-scale plot of the component-wise product $A\phi$ of the aperture function A and the phase ϕ. The middle plot shows the corresponding PSF, $k = |\mathcal{F}^{-1}\{Ae^{i\phi}\}|^2$. The bottom plot shows a logarithmically scaled gray-scale plot of the power spectrum of the PSF.*

which it is nonzero) is a disk. For this reason, the PSF is called band limited, or diffraction limited. See Exercise 5.3 for insight. The exterior of this disk corresponds to the null space of the convolution integral operator in (5.1), and the image deblurring problem violates the uniqueness condition (ii) of well-posedness in Definition 2.7. Functions in the null space of this operator are highly oscillatory. Thus high frequency information about the source

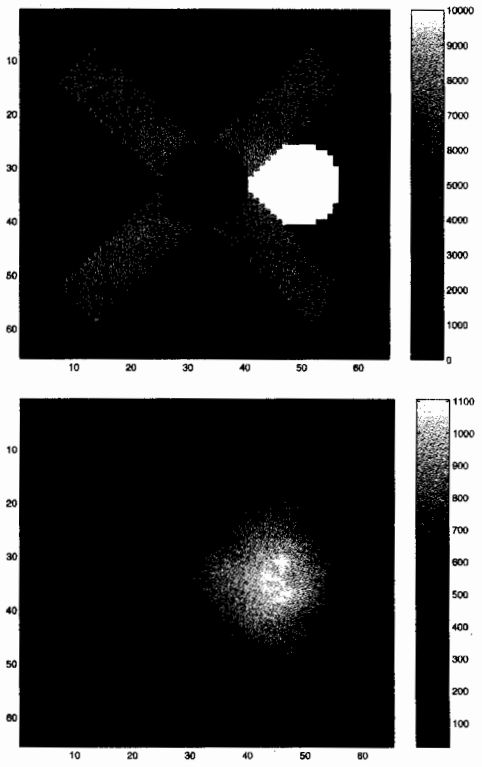


Figure 5.2. *Atmospheric image data. At the top is a gray-scale plot of the source or object. The bottom plot shows a gray-scale plot of the blurred, noisy image.*

(true image) is not present in the image data. Second, near the edge of the disk the power spectrum takes on very small, but nonzero, values. This implies that the stability condition (iii) of Definition 2.7 does not hold.

5.2 Computational Methods for Toeplitz Systems

Linear systems with block Toeplitz structure arise from the discrete convolution product in equation (5.13). We next discuss computational techniques to efficiently solve such systems. The symbol $\mathbb{C}^n$ denotes the set of vectors with n complex components. Here the indices of the components of a generic vector $\mathbf{f} \in \mathbb{C}^n$ will vary from 0 through $n-1$, i.e., $\mathbf{f} = (f_0, \ldots, f_{n-1})$. $\mathbb{C}^n$ is a Hilbert space under the Euclidean inner product and induced norm:

$$\langle \mathbf{f}, \mathbf{g} \rangle = \sum_{j=0}^{n-1} f_j \, \overline{g}_j, \qquad \|\mathbf{f}\| = \sqrt{\sum_{j=0}^{n-1} |f_j|^2}. \tag{5.16}$$

Given $z = \alpha + \hat{\imath}\beta \in \mathbb{C}$, $\overline{z} = \alpha - \hat{\imath}\beta$ denotes the complex conjugate, and $|z| = \sqrt{\alpha^2 + \beta^2}$ denotes magnitude. We denote the set of complex-valued $n_x \times n_y$ arrays by $\mathbb{C}^{n_x \times n_y}$. A generic array $f \in \mathbb{C}^{n_x \times n_y}$ will be indexed by f_{ij}, $i = 0, \ldots, n_x - 1$, $j = 0, \ldots, n_y - 1$.

5.2.1 Discrete Fourier Transform and Convolution

What follows is a discrete analogue of the continuous Fourier transform (5.9).

Definition 5.1. The discrete Fourier transform (DFT) is a mapping on $\mathbb{C}^n$ given by

$$[\mathcal{F}\{\mathbf{f}\}]_i = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} f_j \, e^{-\hat{\imath} 2\pi ij/n}, \qquad i = 0, 1, \ldots, n-1. \tag{5.17}$$

As in (5.9), $\hat{\imath} = \sqrt{-1}$. We can express the DFT as a matrix-vector product, $\mathcal{F}\{\mathbf{f}\} = F\mathbf{f}$, where $F \in \mathbb{C}^{n \times n}$ is the Fourier matrix. This has components

$$[F]_{ij} = \frac{e^{-\hat{\imath} 2\pi ij/n}}{\sqrt{n}}, \qquad 0 \le i, j \le n-1. \tag{5.18}$$

The inverse DFT is given by

$$\begin{aligned} [\mathcal{F}^{-1}\{\mathbf{g}\}]_i &= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} g_j \, e^{\hat{\imath} 2\pi ij/n} \\ &= [F^* \mathbf{g}]_i, \qquad i = 0, \ldots, n-1. \end{aligned} \tag{5.19}$$

This follows from the fact that the Fourier matrix F is a unitary matrix (see Exercise 5.5), i.e., $F^*F = I$. Here the superscript $*$ denotes matrix conjugate transpose.

Remark 5.2. Our definitions of the DFT and its inverse are somewhat nonstandard. Typically the DFT is defined without the factor of $1/\sqrt{n}$ in (5.17), and the inverse DFT is defined with a factor of $1/n$ rather than $1/\sqrt{n}$ in (5.19). This coincides with the functions **fft**$(\cdot)$ and **ifft**$(\cdot)$, which are given in section 5.2.2. We selected our definitions so that both the DFT and its inverse preserve Euclidean inner products and norms (see Exercise 5.6).

Definition 5.3. The discrete convolution product of vectors $\mathbf{t} = (t_{1-n}, \ldots, t_0, t_1, \ldots, t_{n-1})$ and $\mathbf{f} \in \mathbb{C}^n$ is given by

$$[\mathbf{t} \star \mathbf{f}]_i = \sum_{j=0}^{n-1} t_{i-j} f_j, \qquad i = 0, \ldots, n-1. \tag{5.20}$$

Definition 5.4. A discrete vector $\mathbf{t}$ is called n-periodic if

$$t_i = t_j \qquad \text{whenever } i = j \bmod n. \tag{5.21}$$

Given $\mathbf{t} = (t_0, t_1, \ldots, t_{n-1}) \in \mathbb{C}^n$, by the periodic extension of $\mathbf{t}$ of size $2n-1$, we mean the n-periodic vector $\mathbf{t}^{ext} = (t^{ext}_{1-n}, \ldots, t^{ext}_0, t^{ext}_1, \ldots, t^{ext}_{n-1})$ for which $t^{ext}_i = t_i$ for $i = 0, \ldots, n-1$.

Definition 5.5. We adopt the following notation for component-wise multiplication and component-wise division of vectors. For $\mathbf{f}, \mathbf{g} \in \mathbb{C}^n$,

$$[\mathbf{f}. * \mathbf{g}]_i = f_i g_i, \qquad [\mathbf{f}./\mathbf{g}]_i = f_i / g_i, \; g_i \neq 0. \tag{5.22}$$

This notation extends to two-dimensional arrays in an obvious manner.

The next proposition relates the discrete Fourier transform to the discrete convolution product.

Proposition 5.6. *If* $\mathbf{t}, \mathbf{f} \in \mathbb{C}^n$ *and* $\mathbf{t}^{ext}$ *is the periodic extension of* $\mathbf{t}$ *of size* $2n-1$, *then*

$$\frac{1}{\sqrt{n}}\,\mathbf{t}^{ext} \star \mathbf{f} = \mathcal{F}^{-1}\{\mathcal{F}\{\mathbf{t}\}. * \mathcal{F}\{\mathbf{f}\}\}. \tag{5.23}$$

Proof. Set $w = \exp(-\hat{\imath}2\pi/n)$. Then by (5.18), $\sqrt{n}[F]_{ij} = w^{ij}$. Consequently,

$$\begin{aligned}
\sqrt{n}\,[\mathcal{F}\{\mathbf{t}_{ext} \star \mathbf{f}\}]_k &= \sum_{i=0}^{n-1}\left(\sum_{j=0}^{n-1} t_{i-j}^{ext} f_j\right) w^{ik} \\
&= \sum_{j=0}^{n-1} f_j \left(\sum_{i=0}^{n-1} t_{i-j}^{ext} w^{ik}\right) \\
&= \sum_{j=0}^{n-1} f_j \left(\sum_{\ell=-j}^{n-1-j} t_{\ell}^{ext} w^{(\ell+j)k}\right) \\
&= \sum_{j=0}^{n-1} f_j w^{jk} \sum_{\ell=-j}^{n-1-j} t_{\ell}^{ext} w^{\ell k} \\
&= \sqrt{n}\mathcal{F}\{\mathbf{f}\}\,\sqrt{n}\mathcal{F}\{\mathbf{t}\}.
\end{aligned}$$

The last equality follows from the n-periodicity of both $\mathbf{t}^{ext}$ and $w^{\cdot,k}$. □

Proposition 5.6 can be extended to compute two-dimensional discrete convolution products.

Definition 5.7. The two-dimensional DFT is the mapping on $\mathbb{C}^{n_x \times n_y}$ given by

$$[\mathcal{F}\{f\}]_{ij} = \frac{1}{\sqrt{n_x n_y}} \sum_{i'=0}^{n_x-1}\sum_{j'=0}^{n_y-1} f_{i',j'} e^{-\hat{\imath}2\pi\,(ii'/n_x + jj'/n_y)}, \tag{5.24}$$

$0 \le i \le n_x - 1,\ 0 \le j \le n_y - 1$. The inverse two-dimensional DFT is obtained by replacing $-\hat{\imath}$ by $\hat{\imath}$ in equation (5.24).

Definition 5.8. The (two-dimensional) discrete convolution product of an array t, having components t_{ij}, $1 - n_x \le i \le n_x - 1$, $1 - n_y \le j \le n_y - 1$, with an array $f \in \mathbb{C}^{n_x \times n_y}$, is given by

$$[t \star f]_{ij} = \sum_{i'=0}^{n_x-1}\sum_{j'=0}^{n_y-1} t_{i-i',j-j'} f_{i',j'}, \qquad 0 \le i \le n_x - 1,\ 0 \le j \le n_y - 1. \tag{5.25}$$

Definition 5.9. A two-dimensional array t is called (n_x, n_y)-periodic if

$$\begin{aligned}
t_{i,j} &= t_{i',j} \qquad \text{whenever } i = i' \bmod n_x, \\
t_{i,j} &= t_{i,j'} \qquad \text{whenever } j = j' \bmod n_y.
\end{aligned}$$

Let $t \in \mathbb{C}^{n_x \times n_y}$. By the periodic extension of t of size $(2n_x - 1) \times (2n_y - 1)$, we mean the (n_x, n_y)-periodic array t^{ext}, with components t^{ext}_{ij}, $1-n_x \le i \le n_x - 1$, $1-n_y \le j \le n_y - 1$, for which $t^{ext}_{ij} = t_{ij}$ whenever $0 \le i \le n_x - 1$, $0 \le j \le n_y - 1$.

Proposition 5.10. *If t, $f \in \mathbb{C}^{n_x \times n_y}$ and t^{ext} is the periodic extension of t of size $(2n_x - 1) \times (2n_y - 1)$, then*

$$\frac{1}{\sqrt{n_x n_y}}\, t^{ext} \star f = \mathcal{F}^{-1}\{\mathcal{F}\{t\} .* \mathcal{F}\{f\}\}. \tag{5.26}$$

5.2.2 The FFT Algorithm

If the one-dimensional DFT (5.17) were implemented using conventional matrix-vector multiplication, then its computational cost would be $\mathcal{O}(n^2)$, where n is the length of the vector being transformed. The FFT algorithm reduces this computational cost to $\mathcal{O}(n \log n)$. First discovered by Cooley and Tukey [26], this algorithm is used in a broad range of applications in addition to image processing, ranging from time series analysis to the numerical solution of differential equations.

To derive the FFT algorithm, first define $\tilde{F}_n = \sqrt{n}F$, where F is the $n \times n$ Fourier matrix (see (5.18)). The components of $\tilde{F}_n$ are w_n^{ij} with

$$w_n = e^{-\hat{\imath} 2\pi/n}. \tag{5.27}$$

In the computations to follow, we assume that n is an even integer, and we set $m = n/2$ and $w_m = w_n^2 = \exp(-\hat{\imath} 2\pi/m)$.

Given any $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) \in \mathbb{C}^n$, for $i = 0, 1, \ldots, n-1$,

$$\begin{aligned}
[\tilde{F}_n \mathbf{f}]_i &= \sum_{\ell=0}^{m-1} w_n^{i(2\ell)} f_{2\ell} + \sum_{\ell=0}^{m-1} w_n^{i(2\ell+1)} f_{2\ell+1} \\
&= \sum_{\ell=0}^{m-1} w_m^{i\ell} f'_\ell + w_n^i \sum_{\ell=0}^{m-1} w_m^{i\ell} f''_\ell \\
&= [\tilde{F}_m \mathbf{f}']_i + w_n^i [\tilde{F}_m \mathbf{f}'']_i,
\end{aligned} \tag{5.28}$$

where $\mathbf{f}' = (f_0, f_2, \ldots, f_{n-2})$ and $\mathbf{f}'' = (f_1, f_3, \ldots, f_{n-1})$. Note that for $i = 0, 1, \ldots, m-1$,

$$[\tilde{F}_m \mathbf{f}']_{m+i} = \sum_{\ell=0}^{m-1} w_m^{m\ell} w_m^{i\ell} f'_\ell = [\tilde{F}_m \mathbf{f}']_i, \tag{5.29}$$

since $w_m^m = 1$. Similarly,

$$[\tilde{F}_m \mathbf{f}'']_{m+i} = [\tilde{F}_m \mathbf{f}'']_i. \tag{5.30}$$

In addition, since $w_n^m = \exp(-\hat{\imath}\pi) = -1$,

$$w_n^{m+i} = -w_n^i. \tag{5.31}$$

Combining (5.28)–(5.31) and replacing m by $n/2$, we obtain

$$[\tilde{F}_n \mathbf{f}]_i = [\tilde{F}_{n/2} \mathbf{f}']_i + w_n^i [\tilde{F}_{n/2} \mathbf{f}'']_i, \qquad i = 0, 1, \ldots, n/2 - 1, \tag{5.32}$$

$$[\tilde{F}_n \mathbf{f}]_{n/2+i} = [\tilde{F}_{n/2} \mathbf{f}']_i - w_n^i [\tilde{F}_{n/2} \mathbf{f}'']_i, \qquad i = 0, 1, \ldots, n/2 - 1. \tag{5.33}$$

The recursion (5.32)–(5.33) forms the basis for the FFT algorithm (see [110] for implementation details). It reduces the computation of the transform of an n-vector to a pair of transforms of vectors of size $n/2$.

To analyze the computational cost, let FFT(n) represent the number of floating point multiplications required to evaluate $\tilde{F}_n\mathbf{f}$. In equations (5.32)–(5.33), we see that given the vectors $\tilde{F}_{n/2}\mathbf{f}'$ and $\tilde{F}_{n/2}\mathbf{f}''$, only $n/2$ multiplications are needed to evaluate $\tilde{F}_n\mathbf{f}$. Hence,

$$\text{FFT}(n) = n/2 + 2\,\text{FFT}(n/2). \tag{5.34}$$

Since FFT(1) = 0, if we assume that $n = 2^k$, then

$$\text{FFT}(n) = n/2 \times k = n/2 \times \log_2(n). \tag{5.35}$$

See Exercise 5.8.

Note that the inverse DFT (5.19) differs from the forward transform (5.17) only in the replacement of $w_n = \exp(-\hat{\imath}2\pi/n)$ by its complex conjugate, $\overline{w}_n = \exp(\hat{\imath}2\pi/n)$. Thus the algorithm and the cost for computing the inverse discrete Fourier transform are both essentially the same as for the forward transform.

In the material to follow, we indicate multiplication by the scaled Fourier matrix $\tilde{F}_n$ by **fft**$(\cdot)$. Consequently, given $\mathbf{f} = (f_0, f_1, \ldots, f_{n-1}) \in \mathbb{C}^n$,

$$[\mathbf{fft}(\mathbf{f})]_i = \sqrt{n}\,[\mathcal{F}\{\mathbf{f}\}]_i = \sum_{j=0}^{n-1} f_j\, e^{-\hat{\imath}2\pi ij/n}, \qquad i = 0, 1, \ldots, n-1. \tag{5.36}$$

The inverse of **fft** is given by

$$[\mathbf{ifft}(\mathbf{f})]_i = \frac{1}{\sqrt{n}}\,[\mathcal{F}^{-1}\{\mathbf{f}\}]_i = \frac{1}{n}\sum_{j=0}^{n-1} f_j\, e^{\hat{\imath}2\pi ij/n}, \qquad i = 0, 1, \ldots, n-1. \tag{5.37}$$

Thus the discrete convolution result (5.23) can be expressed as

$$\mathbf{t}^{ext} \star \mathbf{f} = \mathbf{ifft}(\mathbf{fft}(\mathbf{t}) .* \mathbf{fft}(\mathbf{f})). \tag{5.38}$$

Two-Dimensional FFTs

We now address the computation of two-dimensional DFTs (see Definition 5.7). Setting $e^{-\hat{\imath}2\pi\,(ii'/n_x + jj'/n_y)} = e^{-\hat{\imath}2\pi\, ii'/n_x} e^{-\hat{\imath}2\pi\, jj'/n_y} \stackrel{\text{def}}{=} w_{n_x}^{ii'} w_{n_y}^{jj'}$, we obtain from (5.24)

$$\sqrt{n_x n_y}\,[\mathcal{F}\{f\}]_{ij} = \sum_{j'=0}^{n_y-1} \left(\sum_{i'=0}^{n_x-1} f_{i'j'} w_{n_x}^{ii'} \right) w_{n_y}^{jj'}. \tag{5.39}$$

The quantity inside the parentheses can be expressed as **fft**$(f_{\cdot,j'})$. By this, we mean that the one-dimensional scaled DFT (5.36) has been applied to each of the n_y columns of the array f. Denote the result by $\hat{f}_x$. Applying **fft** to each of the n_x rows of $\hat{f}_x$ then gives the result in (5.39). From this, we see that the number of multiplications required to evaluate (5.39) is given by

$$\text{FFT2}(n_x, n_y) = n_y \times \text{FFT}(n_x) + n_x \times \text{FFT}(n_y) = \frac{N}{2}\log_2(N), \tag{5.40}$$

where $N = n_x n_y$ is the number of components in the array. Evaluation of two-dimensional inverse DFTs using one-dimensional inverse DFTs can be carried out in the same manner, and the computational cost is the same.

In a manner analogous to (5.36)–(5.37), we define the two-dimensional scaled DFT and its inverse,

$$[\mathbf{fft2}(f)]_{ij} = \sqrt{n_x n_y}[\mathcal{F}\{f\}]_{ij} = \sum_{i'=0}^{n_x-1}\sum_{j'=0}^{n_y-1} f_{i'j'} \exp(-\hat{\imath}2\pi(ii'/n_x + jj'/n_y)),$$

$$[\mathbf{ifft2}(f)]_{ij} = \frac{[\mathcal{F}^{-1}\{f\}]_{ij}}{\sqrt{n_x n_y}} = \frac{1}{n_x n_y}\sum_{j'=0}^{n_y-1} f_{i'j'} \exp(\hat{\imath}2\pi(ii'/n_x + jj'/n_y)).$$

The two-dimensional discrete convolution result (5.26) can then be rewritten as

$$t^{ext} \star f = \mathbf{ifft2}(\mathbf{fft2}(t). * \mathbf{fft2}(f)). \tag{5.41}$$

We next examine matrix representations of discrete convolution operators.

5.2.3 Toeplitz and Circulant Matrices

Definition 5.11. A matrix is called Toeplitz if it is constant along diagonals. An $n \times n$ Toeplitz matrix T has the form

$$T = \begin{bmatrix} t_0 & t_{-1} & \cdots & t_{2-n} & t_{1-n} \\ t_1 & t_0 & t_{-1} & \ddots & t_{2-n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ t_{n-2} & \ddots & t_1 & t_0 & t_{-1} \\ t_{n-1} & t_{n-2} & \cdots & t_1 & t_0 \end{bmatrix}. \tag{5.42}$$

For any vector $\mathbf{f} \in \mathbb{C}^n$, the matrix-vector product $T\mathbf{f}$ has discrete convolution form

$$[T\mathbf{f}]_i = \sum_{j=0}^{n-1} t_{i-j} f_j = [\mathbf{t} \star \mathbf{f}]_i, \qquad i = 0, \ldots, n-1,$$

where $\mathbf{t} = (t_{1-n}, \ldots, t_{-1}, t_0, t_1, \ldots, t_{n-1}) \in \mathbb{C}^{2n-1}$. We indicate this situation by $T = \mathbf{toeplitz}(\mathbf{t})$.

Definition 5.12. An $n \times n$ matrix C is called circulant if it is Toeplitz and its rows are circular right shifts of the elements of the preceding row. In this case we can write

$$C = \begin{bmatrix} c_0 & c_{n-1} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ c_{n-2} & \ddots & c_1 & c_0 & c_{n-1} \\ c_{n-1} & c_{n-2} & \cdots & c_1 & c_0 \end{bmatrix}. \tag{5.43}$$

Then $C = \mathbf{toeplitz}(\mathbf{c}^{ext})$, where $\mathbf{c}^{ext} = (c_1, \ldots, c_{n-1}, c_0, c_1 \ldots, c_{n-1})$ is the n-periodic extension of size $2n-1$ of $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$. We indicate this situation by $C = \mathbf{circulant}(\mathbf{c})$. Note that $\mathbf{c} = C_{\cdot,1}$, the first column of C.

For a detailed discussion of circulant matrices and their properties, see [29]. We next establish the relationship between circulant matrices and the discrete Fourier transform.

Definition 5.13. The *circulant right shift matrix* is given by

$$R = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}. \tag{5.44}$$

It has the property that $R(x_0, x_1, \ldots, x_{n-2}, x_{n-1}) = (x_{n-1}, x_0, \ldots, x_{n-3}, x_{n-2})$.

Proposition 5.14. *If* $C = \mathbf{circulant}(c_0, c_1, \ldots, c_{n-1})$, *then*

$$C = \sum_{j=0}^{n-1} c_j R^j. \tag{5.45}$$

Moreover, $\{\frac{1}{\sqrt{n}} R^j\}_{j=0}^{n-1}$ *forms an orthonormal set under the Frobenius inner product.*

Lemma 5.15. *Let* $w = \exp(-\hat{\imath} 2\pi/n)$. *Then*

$$R = F^* \operatorname{diag}(1, w, w^2, \ldots, w^{n-1})\, F. \tag{5.46}$$

Proof. From (5.18), $[F]_{ij} = w^{ij}/\sqrt{n}$. Consequently,

$$[F^* \operatorname{diag}(1, w, w^2, \ldots, w^{n-1})\, F]_{ij} = \frac{1}{n} \sum_{k=0}^{n-1} \overline{w}^{ik} w^k w^{jk} = \frac{1}{n} \sum_{k=0}^{n-1} w^{(-i+1+j)k}.$$

Equation (5.46) follows from Exercise 5.4 and from (5.44). □

Corollary 5.16. *If* $C = \mathbf{circulant}(\mathbf{c})$, *then*

$$C = F^* \operatorname{diag}(\hat{\mathbf{c}})\, F, \tag{5.47}$$

where F *is the Fourier matrix* (5.18) *and*

$$\hat{\mathbf{c}} = \sqrt{n} F \mathbf{c} = \mathbf{fft}(\mathbf{c}). \tag{5.48}$$

The components of $\hat{\mathbf{c}}$ *are the eigenvalues of* C, *and the columns of* F^* *are the corresponding eigenvectors.*

Proof. From (5.45)–(5.46),

$$C = \sum_{j=0}^{n-1} c_j F^* \left[\operatorname{diag}(w^i)\right]^j F = F^* \operatorname{diag}\left(\sum_{j=0}^{n-1} c_j w^{ij}\right) F.$$

Equations (5.47)–(5.48) now follow from (5.36). □

Remark 5.17. From (5.47)–(5.48) and section 5.2.2, circulant matrix-vector products $\mathbf{v} = C\mathbf{f}$ can be computed at $\mathcal{O}(n \log n)$ cost by (i) computing $\hat{\mathbf{f}} = \mathbf{fft}(\mathbf{f})$; (ii) computing the component-wise vector product $\hat{\mathbf{v}} = \hat{\mathbf{c}} .* \hat{\mathbf{f}}$; and (iii) computing $\mathbf{v} = \mathbf{ifft}(\hat{\mathbf{v}})$. Similarly, nonsingular circulant systems can be solved at $\mathcal{O}(n \log n)$ cost using the fact that

$$C^{-1} = F^* \mathrm{diag}(1./\hat{\mathbf{c}}) F. \tag{5.49}$$

Remark 5.18. Toeplitz matrix-vector products can be efficiently computed by combining circulant embedding with FFTs. Let T be the $n \times n$ Toeplitz matrix in (5.42), let $\mathbf{v} \in R^n$, and define $S = \mathbf{toeplitz}(\mathbf{s})$, with

$$\mathbf{s} = (t_1, t_2, \ldots, t_{n-1}, 0, t_{1-n}, \ldots, t_{-2}, t_{-1}).$$

Then

$$C^{ext} \begin{bmatrix} \mathbf{v} \\ \mathbf{0}_{n\times 1} \end{bmatrix} = \begin{bmatrix} T\mathbf{v} \\ S\mathbf{v} \end{bmatrix}, \qquad \text{where} \quad C^{ext} = \begin{bmatrix} T & S \\ S & T \end{bmatrix}.$$

The $2n \times 2n$ block matrix C^{ext}, which we call the circulant extension of T, can be expressed as $C^{ext} = \mathbf{circulant}(\mathbf{c}^{ext})$, where

$$\mathbf{c}^{ext} = (t_0, t_1, \ldots, t_{n-1}, 0, t_{1-n}, \ldots, t_{-1}) \in \mathbb{C}^{2n}.$$

Consequently, to compute $\mathbf{w} = T\mathbf{v}$, (i) compute $\hat{\mathbf{c}}^{ext} = \mathbf{fft}(\mathbf{c}^{ext})$ and $\hat{\mathbf{v}}^{ext} = \mathbf{fft}((\mathbf{v},\ \mathbf{0}_{n\times 1}))$; (ii) compute $\hat{\mathbf{w}}^{ext} = \hat{\mathbf{c}}^{ext} .* \hat{\mathbf{v}}^{ext}$; (iii) compute $\mathbf{w}^{ext} = \mathbf{ifft}(\hat{\mathbf{w}}^{ext})$; and (iv) extract the first n components of $\mathbf{w}^{ext}$ to obtain $\mathbf{w}$.

5.2.4 Best Circulant Approximation

In this section we consider the computation of circulant approximations to square matrices. These approximations can be used to construct preconditioners for Toeplitz systems and will form the basic components of the level 1 and level 2 block circulant preconditioners discussed in section 5.3.3.

Let $\mathcal{C}_n$ denote the set of $n \times n$ circulant matrices. That this is a linear subspace of $\mathbb{C}^{n\times n}$ is an immediate consequence of the Proposition 5.14. We now apply approximation theoretic tools from section 2.

Definition 5.19. Given $A \in \mathbb{C}^{n\times n}$, the best circulant approximation to A in the Frobenius norm is given by

$$C(A) = \arg\min_{C\in\mathcal{C}_n} ||C - A||_{Fro}. \tag{5.50}$$

Proposition 5.20. *Let $A \in \mathbb{R}^{n\times n}$. Then $C(A) = \mathbf{circulant}(c_0, c_1, \ldots, c_{n-1})$, where*

$$c_j = \frac{1}{n}\langle A, R^j\rangle_{Fro}, \qquad j = 0, 1, \ldots, n-1.$$

The following lemma can be used to verify that best circulant approximation preserves symmetry and positive definiteness. For a proof, see Exercise 5.16.

Lemma 5.21.

$$C(A) = F^* \Lambda F, \tag{5.51}$$

where F is the Fourier matrix and Λ is the diagonal matrix whose diagonal entries are the same as those of FAF^*.

Theorem 5.22. *The mapping $A \mapsto C(A)$ is a (linear) projection operator. This mapping preserves symmetry and positive definiteness.*

$C(A)$ has some nice theoretical approximation properties which make it suitable for preconditioning Toeplitz systems. For details, see [65, Chapter 5].

The following result indicates that the cost of setting up the best circulant approximation to a Toeplitz matrix is $\mathcal{O}(n)$. See Exercise 5.18 for proof.

Corollary 5.23. *Let $T = \mathbf{toeplitz(t)}$, where $\mathbf{t} = (t_{1-n}, \ldots, t_{-1}, t_0, t_1, \ldots, t_{n-1})$. Then $C(T) = \mathbf{circulant(c)}$, where $\mathbf{c}$ has entries*

$$c_j = \frac{(n-j)t_j + jt_{j-n}}{n}, \qquad j = 0, 1, \ldots, n-1.$$

One can also replace the best circulant approximation by the best cosine and sine approximations with respect to the Frobenius norm. See [15, 16] for details. See also Exercise 5.19.

5.2.5 Block Toeplitz and Block Circulant Matrices

We next examine analogues of Toeplitz and circulant matrices that arise in two-dimensional convolution.

Definition 5.24. An $n_x n_y \times n_x n_y$ matrix T is called block Toeplitz with Toeplitz blocks (BTTB) if it has the block form

$$T = \begin{bmatrix} T_0 & T_{-1} & \cdots & T_{1-n_y} \\ T_1 & T_0 & T_{-1} & \vdots \\ \vdots & \ddots & \ddots & T_{-1} \\ T_{n_y-1} & \cdots & T_1 & T_0 \end{bmatrix}, \tag{5.52}$$

where each block T_j is an $n_x \times n_x$ Toeplitz matrix.

Definition 5.25. Given an array $v \in \mathbb{C}^{n_x \times n_y}$, one can obtain a vector $\mathbf{v} \in \mathbb{C}^{n_x n_y}$ by stacking the columns of v. This defines a linear operator $\mathbf{vec} : \mathbb{C}^{n_x \times n_y} \to \mathbb{C}^{n_x n_y}$,

$$\mathbf{vec}(v) = [v_{1,1} \ldots v_{n_x,1} v_{1,2} \ldots v_{n_x,2} \ldots v_{1,n_y} \ldots v_{n_x,n_y}]^T. \tag{5.53}$$

This corresponds to lexicographical column ordering of the components in the array v. The symbol **array** denotes the inverse of the **vec** operator,

$$\mathbf{array}(\mathbf{vec}(v)) = v, \qquad \mathbf{vec}(\mathbf{array}(\mathbf{v})) = \mathbf{v}, \tag{5.54}$$

whenever $v \in \mathbb{C}^{n_x \times n_y}$ and $\mathbf{v} \in \mathbb{C}^{n_x n_y}$.

We can now relate block Toeplitz matrix-vector multiplication to discrete two-dimensional convolution.

Proposition 5.26. *The two-dimensional convolution product* (5.25) *can be expressed as*

$$t \star f = \mathbf{array}(T\ \mathbf{vec}(f)), \tag{5.55}$$

where T is the $n_x n_y \times n_x n_y$ BTTB matrix of the form (5.52) *with $T_j = \mathbf{toeplitz}(t_{\cdot,j})$. Here $t_{\cdot,j}$ denotes the jth column of the $(2n_x - 1) \times (2n_y - 1)$ array t. We indicate this situation by $T = \mathbf{bttb}(t)$.*

Definition 5.27. An $n_x n_y \times n_x n_y$ matrix C is block circulant with circulant blocks (BCCB) if (i) C is BTTB; (ii) the $n_x \times n_x$ block rows of C are all circulant right shifts of each other; and (iii) each block is a circulant matrix. In other words,

$$C = \begin{bmatrix} C_0 & C_{n_y-1} & \cdots & C_2 & C_1 \\ C_1 & C_0 & C_{n_y-1} & \ddots & C_2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ C_{n_y-2} & \ddots & C_1 & C_0 & C_{n_y-1} \\ C_{n_y-1} & C_{n_y-2} & \cdots & C_1 & C_0 \end{bmatrix}, \tag{5.56}$$

where each C_j is an $n_x \times n_x$ circulant matrix.

Proposition 5.28. *Suppose we have a BTTB matrix $C = \mathbf{bttb}(c^{ext})$, where c^{ext} is the periodic extension of $c \in \mathbb{C}^{n_x \times n_y}$ of size $(2n_x - 1) \times (2n_y - 1)$. Then C is BCCB, and we can obtain c from the first column $C_{\cdot,1}$ of C in the representation* (5.56) *by taking*

$$c = \mathbf{array}(C_{\cdot,1}). \tag{5.57}$$

Moreover, we can generate the jth block in (5.56) *from the jth column of c via*

$$C_j = \mathbf{circulant}(c_{\cdot,j}).$$

We indicate this situation by $C = \mathbf{bccb}(c)$.

The computation of the two-dimensional DFT in (5.24) can be carried out by first applying the one-dimensional DFT to the columns of the array f and then applying the one-dimensional DFT to the rows of the resulting array. This can be expressed in terms of matrices.

Definition 5.29. The tensor product of an $m \times n$ matrix A and a $p \times q$ matrix B is the $(mp) \times (nq)$ matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \ldots & a_{1n}B \\ a_{21}B & a_{22}B & \ldots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \ldots & a_{mn}B \end{bmatrix}. \tag{5.58}$$

Proposition 5.30. *Given $f \in \mathbb{C}^{n_y \times n_y}$,*

$$\mathcal{F}\{f\} = \mathbf{array}((F_y \otimes F_x)\ \mathbf{vec}(f)),$$

where F_y and F_x are, respectively, Fourier matrices (5.18) *of size $n_y \times n_y$ and $n_x \times n_x$.*

The following result indicates how to compute BCCB matrix-vector products using two-dimensional DFTs, and it provides the eigenvalues of the BCCB matrix.

Proposition 5.31. *Let $C = \mathbf{bccb}(c)$, where $c \in \mathbb{C}^{n_x \times n_y}$. Then*

$$C = F^* \operatorname{diag}(\mathbf{vec}(\hat{c}))\, F, \tag{5.59}$$

where $F = F_y \otimes F_x$ and

$$\hat{c} = \sqrt{n_x n_y}\,\mathcal{F}\{c\} = \mathbf{fft2}(c).$$

The components of $\hat{c}$ are the eigenvalues of C.

This proposition leads to the following scheme for computing BCCB matrix-vector products. If $\mathbf{f} = \mathbf{vec}(f)$ and C has a representation (5.59), then

$$C\mathbf{f} = \mathbf{vec}(\mathcal{F}^{-1}\{\hat{c}. * \mathcal{F}\{f\}) = \mathbf{vec}(\mathbf{ifft2}(\hat{c}. * \mathbf{fft2}(f))). \tag{5.60}$$

In a manner analogous to the one-dimensional case (see Remark 5.18), BTTB matrix-vector products $T\mathbf{f}$ can be computed using block circulant extension combined with the two-dimensional DFT. Let $T = \mathbf{bttb}(t)$, where $t \in \mathbb{C}^{(2n_x-1)\times(2n_y-1)}$. We first extend t by zeros along the top and left margins, obtaining a $(2n_x) \times (2n_y)$ array

$$\tilde{t} = \begin{bmatrix} 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & t_{1-n_x,1-n_y} & \cdots & t_{1-n_x,0} & \cdots & t_{1-n_x,n_y-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & t_{0,1-n_y} & \cdots & t_{0,0} & \cdots & t_{0,n_y-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & t_{n_x-1,1-n_y} & \cdots & t_{n_x-1,0} & \cdots & t_{n_x-1,n_y-1} \end{bmatrix}. \tag{5.61}$$

Next, partition $\tilde{t}$ into four $n_x \times n_y$ subblocks,

$$\tilde{t} = \begin{bmatrix} \tilde{t}_{11} & \tilde{t}_{12} \\ \tilde{t}_{21} & \tilde{t}_{22} \end{bmatrix}, \tag{5.62}$$

and then reorder the blocks to generate

$$c^{ext} = \begin{bmatrix} \tilde{t}_{22} & \tilde{t}_{21} \\ \tilde{t}_{12} & \tilde{t}_{11} \end{bmatrix}. \tag{5.63}$$

Note that the entry $t_{0,0}$ lies in the upper left corner of c^{ext}. The matrix $C^{ext} = \mathbf{bccb}(c^{ext})$ is the block analogue of the circulant extension in Remark 5.18.

Now to compute $T\mathbf{f}$, where $\mathbf{f} \in \mathbb{C}^{n_x n_y}$, first construct $f = \mathbf{array}(\mathbf{f}) \in \mathbb{C}^{n_x \times n_y}$. Then extend f by zeros to obtain $f^{ext} \in \mathbb{C}^{2n_x \times 2n_y}$,

$$f^{ext} = \begin{bmatrix} f & 0_{n_x \times n_y} \\ 0_{n_x \times n_y} & 0_{n_x \times n_y} \end{bmatrix}. \tag{5.64}$$

Next, compute $\mathbf{g} = C^{ext}\mathbf{vec}(f_{ext})$. Finally, $T\mathbf{f}$ can be obtained by extracting the leading $n_x \times n_y$ subblock of $\mathbf{array}(\mathbf{g})$ and then applying to this subblock the **vec** operator. The next algorithm follows from (5.60).

Algorithm 5.2.1. BTTB Matrix-Vector Product Computation by Block Circulant Extension.
Let $f = \mathbf{array}(\mathbf{f}) \in \mathbb{C}^{n_x \times n_y}$, and let $T = \mathbf{bttb}(t)$, where $t \in \mathbb{C}^{(2n_x-1)\times(2n_y-1)}$. To compute $\mathbf{g} = T\mathbf{f}$ or, equivalently, to compute $g = t \star f$,

Construct $c^{ext} \in \mathbb{C}^{2n_x \times 2n_y}$ from t via (5.61)–(5.63).
$\hat{c}^{ext} := \mathbf{fft2}(c^{ext})$.
Extend f to a $(2n_x) \times (2n_y)$ array, f^{ext} via (5.64).
$\hat{f}^{ext} := \mathbf{fft2}(f^{ext})$.
$\hat{g}^{ext} := \hat{c}^{ext} .* \hat{f}^{ext}$.
$g^{ext} := \mathbf{ifft2}(\hat{g}^{ext})$.
Extract the leading $n_x \times n_y$ subblock of g^{ext} to obtain g.
Then $\mathbf{g} = \mathbf{vec}(g)$.

5.3 Fourier-Based Deblurring Methods

The discrete, noisy data model (5.13) has a matrix-vector representation

$$\mathbf{d} = T\mathbf{f} + \boldsymbol{\eta}, \tag{5.65}$$

where $T = \mathbf{bttb}(t)$ (see Proposition 5.26), $\mathbf{d} = \mathbf{vec}(d)$, $\mathbf{f} = \mathbf{vec}(f)$, and $\boldsymbol{\eta} = \mathbf{vec}(\eta)$. We refer to T as the blurring matrix. Given T and $\mathbf{d}$, we wish to estimate $\mathbf{f}$. For now we assume that the (unknown) error term $\boldsymbol{\eta}$ is predominantly Gaussian. This suggests a least squares fit-to-data functional; see Example 4.26. We consider the discrete Tikhonov functional with a quadratic penalty term:

$$\mathcal{T}_\alpha(\mathbf{f}) = \frac{1}{2}||T\mathbf{f} - \mathbf{d}||^2 + \frac{\alpha}{2}\mathbf{f}^* L\mathbf{f}. \tag{5.66}$$

Here L is symmetric positive definite and is called the penalty matrix, and $\alpha > 0$ is the regularization parameter. A minimizer of $\mathcal{T}_\alpha$ solves the linear system

$$(T^*T + \alpha L)\mathbf{f} = T^*\mathbf{d}. \tag{5.67}$$

Consider the squared L^2-norm penalty functional $J(f)$; see (2.46). Applying midpoint quadrature on an equispaced grid as in section 5.1, we obtain

$$J(f) = \frac{1}{2}\int\int f(x,y)^2\, dx\, dy \approx \frac{\Delta x\, \Delta y}{2} \sum_{i=0}^{n_x-1}\sum_{j=0}^{n_y-1} f_{ij}^2. \tag{5.68}$$

By incorporating the factor of $\Delta x\, \Delta y$ into the regularization parameter α and reordering the f_{ij}'s into a column vector $\mathbf{f}$, the right-hand side of (5.68) corresponds to the penalty matrix $L = I$, the identity matrix, in (5.66)–(5.67).

To incorporate an a priori assumption of smoothness, one can apply the Sobolev H^1 penalty functional (2.47). The corresponding penalty operator is the negative Laplacian; cf. (2.49) with diffusion coefficient $\kappa = 1$. Standard finite difference approximation of derivatives yields (up to a constant multiple) the negative discrete Laplacian,

$$[\mathcal{L}f]_{ij} = 4f_{ij} - f_{i+1,j} - f_{i-1,j} - f_{i,j+1} - f_{i,j-1}. \tag{5.69}$$

If we assume periodic boundary conditions

$$f_{0,j} = f_{n_x+1,j} \quad \text{and} \quad f_{i,0} = f_{i,n_y+1},$$

and we apply lexicographical column ordering of the unknowns, we obtain the $n_x n_y \times n_x n_y$ penalty matrix with $n_y \times n_y$ block representation

$$L = \begin{bmatrix} L_0 & -I & \Theta & \dots & \Theta & -I \\ -I & L_0 & -I & \Theta & \ddots & \Theta \\ \Theta & -I & L_0 & -I & \Theta & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \\ \Theta & \ddots & \Theta & -I & L_0 & -I \\ -I & \Theta & \dots & \Theta & -I & L_0 \end{bmatrix}. \tag{5.70}$$

Here I represents the $n_x \times n_x$ identity matrix, Θ represents the $n_x \times n_x$ zero matrix, and L_0 is the $n_x \times n_x$ matrix of the form

$$L_0 = \begin{bmatrix} 4 & -1 & 0 & \dots & 0 & -1 \\ -1 & 4 & -1 & 0 & \ddots & 0 \\ 0 & -1 & 4 & -1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & 0 & -1 & 4 & -1 \\ -1 & 0 & \dots & 0 & -1 & 4 \end{bmatrix}. \tag{5.71}$$

Remark 5.32. To replace periodic boundary conditions with homogeneous Dirichlet boundary conditions, replace the upper right and lower left $-I$'s in (5.70) with Θ's, and drop the corner -1's from L_0 in (5.71). See [102]. Additional modifications to the main diagonal of L are needed to incorporate homogeneous Neumann, or no-flux, boundary conditions. In this case, each of the rows of L must sum to zero, so the 4's on the main diagonal may be replaced by 3's or 2's, depending on the number of off-diagonal -1's in a given row. Again, see [102]. In both the Dirichlet and Neumann cases, BCCB structure is lost. In the Dirichlet case, the matrix L is BTTB.

Remark 5.33. Regardless of whether the boundary conditions are periodic, Dirichlet, or Neumann, the matrix L corresponding to (5.69) is sparse, and matrix-vector products can be computed in $5n_x n_y + \mathcal{O}(1)$ operations. L is also symmetric and positive semidefinite. In the Dirichlet case, L is positive definite.

5.3.1 Direct Fourier Inversion

If both the blurring matrix T and the penalty matrix L are BCCB, then the Tikhonov system (5.67) can be solved directly using two-dimensional FFTs. In this case, L has an array representation

$$\mathcal{L}f = \mathbf{ifft2}(\hat{\ell} .* \mathbf{fft2}(f)), \qquad f \in \mathbb{R}^{n_x \times n_y}. \tag{5.72}$$

For example, the identity matrix has such a representation with $\hat{\ell} = 1_{n_x \times n_y}$, the $n_x \times n_y$ array of ones. For the discrete Laplacian with periodic boundary conditions (see (5.70)–(5.71)),

one takes $\hat{\ell} = \textbf{fft2}(\textbf{array}(L_{\cdot,1}))$, where $L_{\cdot,1}$ denotes the first column of the matrix L in (5.70). Let $T = \textbf{bccb}(t)$ with $t \in \mathbb{R}^{n_x \times n_y}$ (Proposition 5.28). The following algorithm yields the solution to (5.67).

Algorithm 5.3.1. Tikhonov Regularization for BCCB Systems.
Given $d = \textbf{array}(\textbf{d}) \in \mathbb{R}^{n_x \times n_y}$, given $t \in \mathbb{R}^{n_x \times n_y}$ for which $T = \textbf{bccb}(t)$, and given $\ell \in \mathbb{R}^{n_x \times n_y}$ for which $L = \textbf{bccb}(\ell)$, to solve the system
$(T^*T + \alpha L)\mathbf{f} = \mathbf{d}$,

$\hat{\ell} := \textbf{fft2}(\ell)$; *% Fourier representer for L*
$\hat{t} := \textbf{fft2}(t)$; *% Fourier representer for T*
$\hat{d} := \textbf{fft2}(d)$;
$\hat{f} := \text{conj}(\hat{t}) .* \hat{d} ./ (|\hat{t}|^2 + \alpha\hat{\ell})$;
$f := \textbf{ifft2}(\hat{f})$;

Then $\mathbf{f} = \textbf{vec}(f)$.

Figure 5.3 shows some two-dimensional image reconstructions computed using this algorithm. Two penalty matrices L are employed: the identity $L = I$ and the negative discrete Laplacian with periodic boundary conditions; see (5.70)–(5.71). As should be expected, the best reconstruction obtained with the identity penalty matrix is less smooth than the corresponding best reconstruction obtained using the discrete Laplacian.

If either L or T is not BCCB, then Fourier transform techniques cannot be used directly to solve system (5.67). The use of direct matrix decomposition methods like the LU factorization or the Cholesky factorization [46] to solve this system is often precluded by the size $n_x n_y$ and the fact that T is not sparse. With certain boundary conditions, other fast transform techniques, e.g., fast cosine transform for Neumann boundary conditions, may be directly applied to (5.67). Otherwise, iterative techniques like the conjugate gradient Algorithm 3.2 must be used.

5.3.2 CG for Block Toeplitz Systems

Assume now that the blurring matrix T in (5.66) is BTTB and the penalty matrix L is symmetric positive semidefinite with $\mathcal{O}(n)$ nonzero entries, where $n = n_x n_y$ denotes the size of T and L. Let

$$A = T^*T + \alpha L \tag{5.73}$$

denote the coefficient matrix in (5.67). We assume that A is nonsingular, so it is SPD. Corollary 3.8 then guarantees that the CG Algorithm 3.2 will generate iterates that converge to the solution of system (5.67). From Algorithm 5.2.5, section 5.2.2, and Remark 5.33 we see that the cost of applying the matrix A to a vector is $\mathcal{O}(n \log n)$. This dominates the $\mathcal{O}(n)$ cost of the inner product computations in each CG iteration; see Remark 3.10. Given this relatively low cost per iteration, CG is often a viable method for solving (5.67), even when the convergence rate is somewhat slow.

Remark 5.34. CG iterations can be applied to minimize the unregularized least squares cost functional $J(\mathbf{f}) = ||T\mathbf{f} - \mathbf{d}||^2$ or, equivalently, to solve the "normal equations":

$$T^*T\mathbf{f} = T^*\mathbf{d}.$$

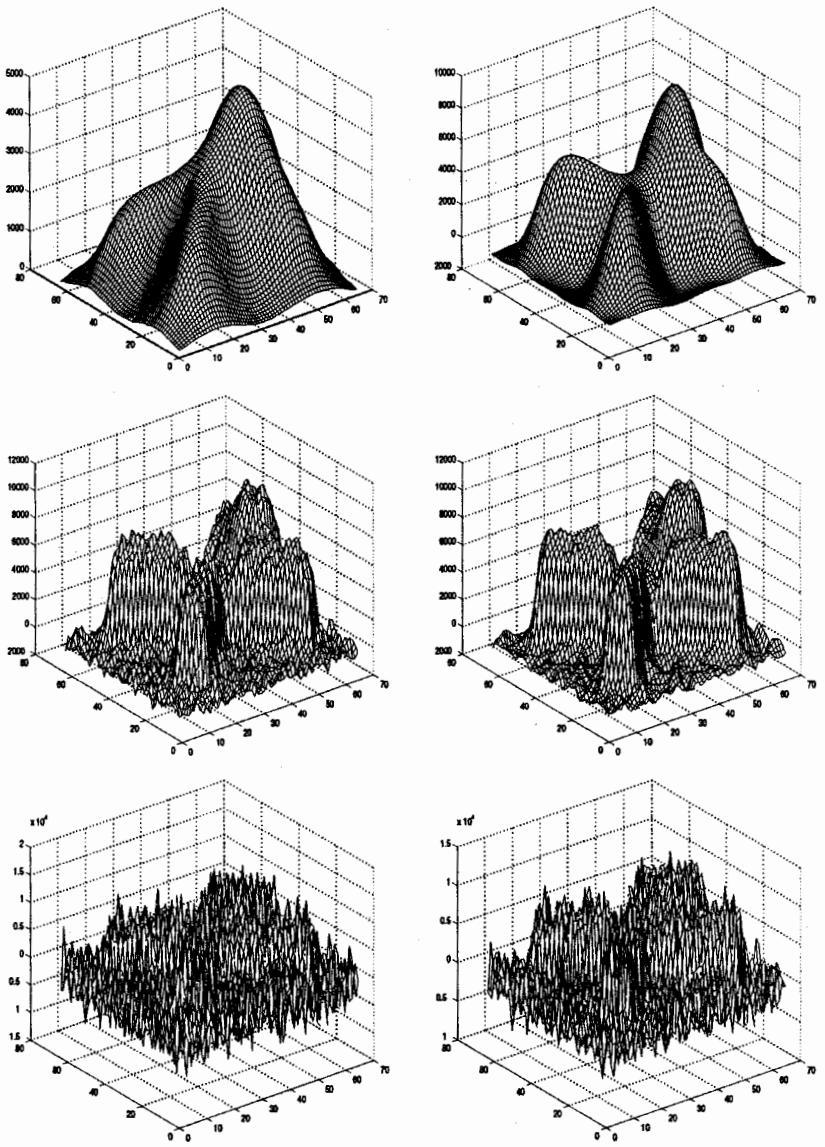


Figure 5.3. *Reconstructions from data in Figures* 5.1–5.2, *generated using Algorithm* 5.3.1. *Reconstructions on the left were obtained using the identity penalty matrix. Those on the right were obtained using the negative Laplacian with periodic boundary conditions. Images in the top row were generated with regularization parameter* $\alpha = 10^{-2}$, *in the second row,* $\alpha = 10^{-4}$, *and in the third row,* $\alpha = 10^{-6}$.

See [9] for implementation details. As with Landweber iteration, (section 1.4), the CG iteration count plays the role of the regularization parameter. See [50] and the references therein, or see [35, Chapter 7].

Preconditioning can sometimes significantly speed up CG convergence for system (5.67).

5.3.3 Block Circulant Preconditioners

The Block Circulant Extension Preconditioner

The inversion of Toeplitz systems by circulant preconditioning was first suggested by Strang [105]. See [14] for an extensive review of this subject. In this section we discuss various block generalizations of circulant preconditioners.

The following preconditioner is based on the extension idea underlying Algorithm 5.2.5. It is similar to the Toeplitz approximate inverse preconditioners of Hanke and Nagy [53]. To apply this preconditioner to the Tikhonov system (5.67), we require both the blurring matrix T and the penalty matrix L to be BTTB.

Algorithm 5.3.2. Preconditioning the Tikhonov System by Block Circulant Extension. Let $T = \mathbf{bttb}(t)$ and $L = \mathbf{bttb}(\ell)$, where $t, \ell \in \mathbb{C}^{(2n_x-1)\times(2n_y-1)}$. Let $r \in \mathbb{C}^{n_x \times n_y}$ and $\mathbf{r} = \mathbf{vec}(r)$. To compute $\mathbf{s} = M^{-1}\mathbf{r}$, where M is the block circulant extension preconditioner for $A = T^*T + \alpha L$,

Assemble $c_t \in \mathbb{C}^{2n_x \times 2n_y}$ and $c_\ell \in \mathbb{C}^{2n_x \times 2n_y}$ from t and ℓ
via (5.61)–(5.63).
$\hat{c}_t := \mathbf{fft2}(c_t)$.
$\hat{c}_\ell := \mathbf{fft2}(c_\ell)$.
Extend r to a $(2n_x) \times (2n_y)$ array, $r_{ext} = \begin{bmatrix} r & 0_{n_x \times n_y} \\ 0_{n_x \times n_y} & 0_{n_x \times n_y} \end{bmatrix}$.
$\hat{r}_{ext} := \mathbf{fft2}(r_{ext})$.
$\hat{s}_{ext} := \hat{r}_{ext} ./ (|\hat{c}_t|^2 + \alpha\, \hat{c}_\ell)$.
$s_{ext} := \mathbf{ifft2}(\hat{s}_{ext})$.
Extract the leading $n_x \times n_y$ subblock of s_{ext} to obtain s.

Then $\mathbf{s} = \mathbf{vec}(s)$.

Level 1 and level 2 block circulant preconditioners are derived from the best circulant approximation, presented in section 5.2.4. See [20, 17, 18] for further details. Unlike the block circulant extension preconditioner, these can be applied even when the matrices T and L are not BTTB.

Level 1 Block Circulant Preconditioning

We begin with the construction of the level 1 block circulant approximation. Assume for simplicity that T is BTTB with representation (5.52). Its level 1 approximation, which we denote by $C_1(T)$ is obtained by replacing each of the blocks T_j by its best circulant

approximation $C(T_j)$, i.e.,

$$C_1(T) = \begin{bmatrix} C(T_0) & C(T_{-1}) & \cdots & C(T_{1-n_y}) \\ C(T_1) & C(T_0) & C(T_{-1}) & \vdots \\ \vdots & \ddots & \ddots & C(T_{-1}) \\ C(T_{n_y-1}) & \cdots & C(T_1) & C(T_0) \end{bmatrix}. \tag{5.74}$$

Note that $C_1(T)$ is block Toeplitz with circulant blocks. More generally, if T is a block matrix with block components T_{ij}, then $C_1(T)$ is the block matrix with circulant blocks $C(T_{ij})$. Similarly, one can compute the level 1 approximation to the penalty matrix L. For instance, for the discrete Laplacian (5.70) with Dirichlet boundary conditions,

$$C_1(L) = \begin{bmatrix} C(L_0) & -I & \Theta & \dots & \Theta & \Theta \\ -I & C(L_0) & -I & \Theta & \ddots & \Theta \\ \Theta & -I & C(L_0) & -I & \Theta & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \Theta \\ \Theta & \ddots & \Theta & -I & C(L_0) & -I \\ \Theta & \Theta & \dots & \Theta & -I & C(L_0) \end{bmatrix}; \tag{5.75}$$

see Remark 5.32. This computation is greatly simplified by the fact that $C(I) = I$ and $C(\Theta) = \Theta$, consequences of Theorem 5.22. If Neumann boundary conditions are used instead of Dirichlet boundary conditions, the upper left and lower right $C(L_0)$ blocks must be modified.

As a consequence of Theorem 5.22, the level 1 circulant approximation to general block matrices is linear, i.e.,

$$C_1(\alpha A + \beta B) = \alpha C_1(A) + \beta C_1(B), \tag{5.76}$$

for any block matrices A and B and scalars α and β. One can also show [20] that $C_1(A)$ is SPD whenever A is.

Now consider the construction of a preconditioner for $A = T^*T + \alpha L$. From (5.76), $C_1(A) = C_1(T^*T) + \alpha C_1(L)$. Unfortunately, T^*T need not be BTTB (see Exercise 5.26), and $C_1(T^*T)$ may be difficult to compute. However, $C_1(T^*T)$ may be well approximated by $C_1(T)^*\, C_1(T)$. Hence, we take as a preconditioner for A

$$M_1(A) = C_1(T)^*\, C_1(T) + \alpha C_1(L). \tag{5.77}$$

To implement this level 1 preconditioner, we rely on the fact that the discrete Fourier transform can be used to diagonalize circulant matrices; see Corollary 5.16. Applying this to the blocks of (5.74) gives

$$C(T_j) = F_x^* \Lambda_j F_x, \qquad j = 1 - n_y, \dots, 0, \dots, n_y - 1,$$

where F_x denotes the $n_x \times n_x$ Fourier matrix, and Λ_j denotes the $n_x \times n_x$ diagonal matrix whose diagonal entries are the eigenvalues of $C(T_j)$. These entries are the components of **fft2**($\mathbf{c}_j$), where $C(T_j) =$ **circulant**($\mathbf{c}_j$). From this we obtain

$$C_1(T) = (I_y \otimes F_x)^*\, T(\Lambda)\, (I_y \otimes F_x), \tag{5.78}$$

where I_y denotes the $n_y \times n_y$ identity matrix and $T(\Lambda)$ is the block Toeplitz matrix with the Λ_j's as its diagonal blocks,

$$T(\Lambda) = \begin{bmatrix} \Lambda_0 & \Lambda_{-1} & \cdots & \Lambda_{1-n_y} \\ \Lambda_1 & \Lambda_0 & \Lambda_{-1} & \vdots \\ \vdots & \ddots & \ddots & \Lambda_{-1} \\ \Lambda_{n_y-1} & \cdots & \Lambda_1 & \Lambda_0 \end{bmatrix}.$$

There exists a permutation matrix P, corresponding to a reindexing of unknowns from column lexicographical order to row lexicographical order, for which $P^T\, T(\Lambda)\, P$ is block diagonal. Hence,

$$C_1(T) = (I_y \otimes F_x)^*\, P^T\, \mathrm{diag}(D_1, \ldots, D_{n_x})\, P\, (I_y \otimes F_x), \tag{5.79}$$

where the diagonal blocks D_k are each dense $n_y \times n_y$ matrices with components

$$[D_k]_{ij} = [\Lambda_{i-j}]_{k,k}, \qquad 1 \le i, j \le n_y, \quad k = 1, \ldots, n_x.$$

Assume that L has an analogous representation:

$$C_1(L) = (I_y \otimes F_x)^*\, P^T\, \mathrm{diag}(E_1, \ldots, E_{n_x})\, P\, (I_y \otimes F_x). \tag{5.80}$$

From (5.79)–(5.80) and the fact that $I_y \otimes F_x$ and P are both unitary matrices, we obtain the following representation for the preconditioner (5.77):

$$M_1(A) = (I_y \otimes F_x)^*\, P^T\, \mathrm{diag}(D_k^2 + \alpha E_k)\, P\, (I_y \otimes F_x). \tag{5.81}$$

Now consider the computation of

$$\mathbf{w} = M_1(A)^{-1}\mathbf{v} = (I_y \otimes F_x)^*\, P^T\, \mathrm{diag}(D_k^2 + \alpha E_k)^{-1}\, P\, (I_y \otimes F_x),$$

where $\mathbf{v} = \mathbf{vec}(v)$ and v is an $n_x \times n_y$ array. The matrix-vector product $\hat{\mathbf{v}} = (I_y \otimes F_x)\mathbf{v}$ corresponds to applying one-dimensional DFTs to the columns of v. Let $\hat{v} = \mathbf{array}(\hat{\mathbf{v}})$. The computation $\hat{\mathbf{w}} = P^T\, \mathrm{diag}(D_k^2 + \alpha E_k)^{-1}\, P\, \hat{\mathbf{v}}$ can be carried out by solving linear systems

$$(D_k^2 + \alpha E_k)\, \hat{w}_{k,\cdot} = \hat{v}_{k,\cdot}, \qquad k = 1, \ldots, n_x, \tag{5.82}$$

each of size $n_y \times n_y$, where $\hat{v}_{k,\cdot}$ denotes the kth row of $\hat{v}$, and storing $\hat{w}_{k,\cdot}$ as the kth row of $\hat{w} = \mathbf{array}(\hat{\mathbf{w}})$. Finally, the computation $\mathbf{w} = (I_y \otimes F_x)^*\hat{\mathbf{w}}$ corresponds to applying inverse DFTs to the columns of $\hat{w}$. This yields $w = \mathbf{array}(\mathbf{w})$.

Level 2 Block Circulant Preconditioning

Suppose we have a block matrix T with a level 1 block circulant approximation $C_1(T)$ of the form (5.79). The level 2 block circulant approximation to T, which we denote by $C_2(T)$, can be obtained simply by replacing each of the diagonal block matrices D_k by its best circulant approximation $C(D_k)$. Let

$$C(D_k) = F_y^*\, \mathrm{diag}(\hat{\mathbf{d}}_k)\, F_y,$$

where $\mathrm{diag}(\hat{\mathbf{d}}_k)$ denotes the diagonal matrix whose diagonal entries comprise the components of the vector $\hat{\mathbf{d}}_k \in \mathbb{C}^{n_y}$. This yields the representation

$$\begin{aligned} C_2(T) &= (I_y \otimes F_x)^*\, P^T\, (I_y \otimes F_y)^* \\ &\quad \times \mathrm{diag}(\hat{\mathbf{d}}_1, \ldots, \hat{\mathbf{d}}_{n_x})\, (I_y \otimes F_y)\, P\, (I_y \otimes F_x) \\ &= (F_y \otimes F_x)^*\, \mathrm{diag}(\hat{\mathbf{d}}_1, \ldots, \hat{\mathbf{d}}_{n_x})\, (F_y \otimes F_x), \end{aligned} \tag{5.83}$$

where now $\text{diag}(\mathbf{d}_1, \ldots, \mathbf{d}_{n_x})$ denotes the $n_x n_y \times n_x n_y$ diagonal matrix with diagonal components equal to those of $\mathbf{vec}([\mathbf{d}_1 \ldots \mathbf{d}_{n_x}])$.

As was the case with the level 1 approximation, the level 2 approximation is linear (see (5.76)) and preserves symmetry and positive definiteness [20].

Based on (5.77) and (5.83), as a preconditioner for $A = T^*T + \alpha L$ we take

$$\begin{aligned} M_2(A) &= C_2(T)^* C_2(T) + \alpha C_2(L) \\ &= (F_y \otimes F_x)^* \, \text{diag}(|\hat{\mathbf{d}}_k|.^2 + \hat{\mathbf{e}}_k) \, (F_y \otimes F_x). \end{aligned} \tag{5.84}$$

The computation $\mathbf{w} = M_2(A)^{-1}\mathbf{v}$ is straightforward. The matrix-vector product $\hat{\mathbf{v}} = (F_y \otimes F_x)\,\mathbf{v}$ corresponds to applying the two-dimensional DFT to $v = \mathbf{array}(\mathbf{v})$. Next, to compute $\hat{\mathbf{w}} = \text{diag}(|\hat{\mathbf{d}}_k|.^2 + \hat{\mathbf{e}}_k)^{-1}\hat{\mathbf{v}}$, take $\hat{w} = \mathbf{array}(\hat{\mathbf{w}})$ to consist of columns

$$\hat{w}_{:,k} = \hat{v}_{:,k}./(|\hat{\mathbf{d}}_k|.^2 + \hat{\mathbf{e}}_k), \qquad k = 1, \ldots, n_y.$$

Finally, apply the inverse two-dimensional DFT to $\hat{w}$ to obtain $w = \mathbf{array}(\mathbf{w})$.

5.3.4 A Comparison of Block Circulant Preconditioners

We now compare the numerical performance of the various block circulant preconditioners. The test problem used here is an atmospheric optics deblurring problem very similar to that presented in Figures 5.1–5.2. The reconstruction method is Tikhonov regularization with the identity regularization operator, and the value used for the regularization parameter is $\alpha = 10^{-4}$. The image lies on a 128×128 pixel grid. Hence the number of unknowns is $n_x n_y = 128^2 = 16384$. Figure 5.4 shows the iterative solution error norm, $||f_\alpha^\nu - f_\alpha||$. Here f_α denotes the exact solution to the regularized system (5.67) and f_α^ν denotes the approximate solution obtained after ν PCG iterations. Note that unpreconditioned CG converged the most slowly, followed by PCG with level 2 preconditioning and PCG with level 1 preconditioning. PCG with block circulant extension preconditioning converged the fastest. Keep in mind that convergence rates may change as one varies parameters like n_x, n_y, and α.

The total computational cost of an iterative method equals the cost per iteration multiplied by the number of iterations required to meet a particular stopping tolerance. Except for the level 1 preconditioner, the cost per iteration is dominated by two-dimensional forward and inverse FFTs. As in section 5.2.2, let $\text{FFT2}(m, n)$ denote the cost of applying a forward or inverse two-dimensional FFT to an $m \times n$ array. Without preconditioning, each CG iteration costs $4 \times \text{FFT2}(2n_x, 2n_y) + \mathcal{O}(n_x n_y)$. The FFT costs are $2 \times \text{FFT2}(2n_x, 2n_y)$ to apply the BTTB matrix T on a $2n_x \times 2n_y$ grid and $2 \times \text{FFT2}(2n_x, 2n_y)$ to apply T^*; see Algorithm 5.2.5. Each application of the block circulant extension preconditioner costs an additional $2 \times \text{FFT2}(2n_x, 2n_y) + \mathcal{O}(n_x n_y)$; see Algorithm 5.3.3. This brings the total cost of each iteration of PCG with the block circulant extension preconditioner to essentially $6 \times \text{FFT2}(2n_x, 2n_y)$. Given the dramatic improvement in the convergence rate shown in Figure 5.4, it is clear that block circulant preconditioning greatly reduces the total cost when compared to unpreconditioned CG.

The cost of applying the level 2 block preconditioner is dominated by $2 \times \text{FFT2}(n_x, n_y)$ (note that no block circulant extension is required). Hence, the total cost per iteration of level 2 PCG is slightly more than two-thirds that of block circulant extension PCG. However, because of the differences in convergence rates, the overall cost of level 2 PCG is less than that of unpreconditioned CG but more than that for PCG with block circulant extension preconditioning.

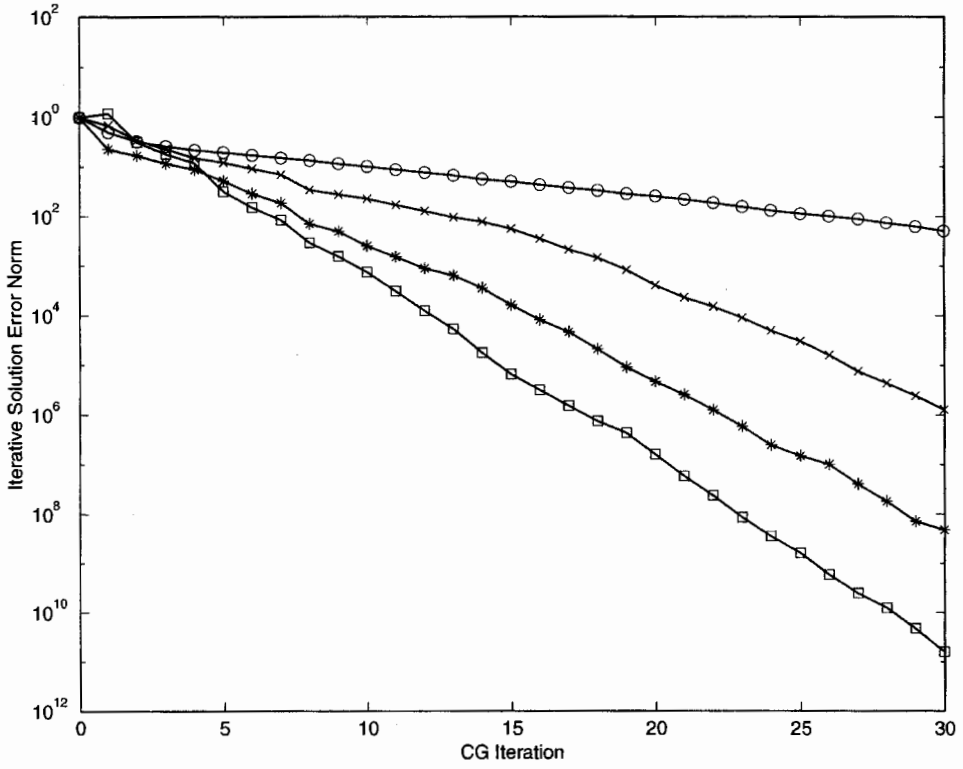


Figure 5.4. *Performance of block circulant preconditioners for a two-dimensional image deblurring problem. The iterative solution error norm, $||f_\alpha^\nu - f_\alpha||$, is plotted against iteration count ν. The circles denote unpreconditioned CG; x's denote PCG with level 2 preconditioning; asterisks denote PCG with level 1 preconditioning; and squares denote PCG with preconditioning by block circulant extension.*

Finally, we examine the cost of the level 1 preconditioner. Let FFT(n) denote the cost of applying a forward or inverse FFT to a vector of length n. The cost of the FFTs in each application of this preconditioner is then $2n_y \times \text{FFT}(n_x)$. However, one must also solve the block systems (5.82). Assuming that Cholesky factorizations of the (dense) blocks have been precomputed, this cost is $n_y \times (n_x^2 + \mathcal{O}(n_x))$. This dominates the FFT costs. To compare this with the cost of the other preconditioners, assume $n_x = n_y$. Then for large n_x the level 1 cost is essentially n_x^3. One the other hand, the cost of the other preconditioners and of unpreconditioned CG is $\mathcal{O}(n_x^2 \log n_x)$—a quantity that becomes substantially *smaller* than n_x^3 for large n_x. Given this, and given the convergence behavior seen in Figure 5.4, PCG with level 1 preconditioning is more expensive than either unpreconditioned CG or PCG with any of the other preconditioners.

5.4 Multilevel Techniques

Multilevel techniques, which include wavelet and multigrid methods, are interesting alternatives to Fourier-based methods for the solution to the large linear systems that arise in image reconstruction. Rieder [97] combined wavelet decomposition techniques with Jacobi and

Gauss–Seidel type iterative methods to solve Fredholm first kind integral equations. Hanke and Vogel [55, 118] used Rieder's ideas to develop a class of two-level preconditioners to solve linear systems arising from regularized inverse problems. See also Jacobsen's MSc. thesis [63]. Riley and Vogel [98] showed that two-level preconditioners can be competitive with block circulant preconditioners in image reconstruction applications.

Unlike circulant preconditioners, multilevel techniques do not require Toeplitz structure. Work by Vogel [117] indicates that no advantage is gained by using more than two levels.

Exercises

5.1. Apply midpoint quadrature to obtain the discrete convolution in (5.13) from the continuous version (5.1). Show that $t_{ij} = k((i-1)\Delta x, (j-1)\Delta y)\ \Delta x \Delta y$, where $\Delta x = 1/n_x$, $\Delta y = 1/n_y$.

5.2. Interpret the DFT (5.17) as a truncation of domain combined with a quadrature applied to the continuous Fourier transform (5.9). Obtain an error bound for this approximation in terms of grid spacing and domain truncation.

5.3. Let A be in indicator function for the interval $[-1, 1]$,

$$A(x) = \begin{cases} 1, & -1 \le x \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

Prove that $\hat{k} = \mathcal{F}\{|\mathcal{F}^{-1}\{A\}|^2\}$ is the ramp function,

$$\hat{k}(x) = \begin{cases} 2 - |x|, & -2 \le x \le 2, \\ 0 & \text{otherwise.} \end{cases}$$

5.4. Let $\omega = \exp(-\hat{\imath} 2\pi/n)$, where $\hat{\imath} = \sqrt{-1}$. Show that

$$\sum_{j=0}^{n-1} \omega^{jk} = \begin{cases} n & \text{if } k = 0, \\ 0 & \text{if } k = 1, 2, \ldots, n-1. \end{cases}$$

5.5. Prove that the Fourier matrix (5.18) is unitary. *Hint:* Show that $n[F^* F]_{jk} = \sum_{\ell=0}^{n-1} w^{\ell(-j+k)}$, where $w = \exp(-\hat{2}\pi/n)$. Then apply Exercise 5.4.

5.6. Prove that the DFT preserves Euclidean inner products and hence that it preserves Euclidean norms.

5.7. Prove Proposition 5.10.

5.8. Derive equation (5.35), given (5.34), $\text{FFT}(1) = 0$, and $n = 2^k$.

5.9. Verify that (5.23) and (5.38) are equivalent.

5.10. Verify the equalities in (5.40).

5.11. Prove Proposition 5.14.

5.12. Use Corollary 5.16 to prove Proposition 5.6.

5.13. Given the Fourier eigendecomposition (5.47) of a circulant matrix C, explain how to directly obtain the SVD of C.

5.14. Prove Proposition 5.20.

5.15. Show that if U is a unitary matrix, then $||UA||_{Fro} = ||A||_{Fro}$.

5.16. Prove Lemma 5.21. *Hint:* If C is circulant, then by Corollary 5.16 $C = F^*\Lambda F$ for some diagonal matrix Λ. Now use the fact that unitary transformations like F preserve Frobenius norm.

5.17. Prove Theorem 5.22.

5.18. Prove Corollary 5.23.

5.19. The $n \times n$ Cosine matrix Cos has entries

$$[\text{Cos}]_{ij} = \begin{cases} \frac{1}{\sqrt{n}}, & j = 1,\ i = 1, \ldots, n, \\ \sqrt{\frac{2}{n}} \cos\left(\frac{(2i-1)(j-1)\pi}{2n}\right), & j = 2, \ldots, n,\ i = 1, \ldots, n. \end{cases}$$

Show that this is a unitary matrix.

5.20. Prove Proposition 5.26.

5.21. Prove Proposition 5.28.

5.22. Prove Proposition 5.30.

5.23. Prove that the tensor product of Fourier matrices is unitary.

5.24. Prove Proposition 5.31.

5.25. Explain how to modify Algorithm 5.2.5 to handle the case when the array t has size $n_x \times n_y$ rather than $(2n_x - 1) \times (2n_y - 1)$. This happens when the PSF is taken to be the recorded image of an approximate point source.

5.26. Provide a simple counterexample to show that the product of Toeplitz matrices need not be Toeplitz.

5.27. Verify equation (5.78).

5.28. Give an explicit representation of the column-to-row permutation matrix P in equation (5.79).

5.29. Show that $(I_y \otimes F_y)\ P\ (I_y \otimes F_x) = F_y \otimes F_x$, thereby verifying equation (5.83).

5.30. Let T denote the Toeplitz matrix arising from the one-dimensional test problem of Chapter 1. Solve the linear system $(T^*T + \alpha I)\mathbf{f} = T^*\mathbf{d}$ using the preconditioned CG method with a preconditioner constructed from $C(T)$. Compare the results with unpreconditioned CG in terms of convergence rates and computational cost. How does the choice of the regularization parameter α affect convergence rates?

5.31. In the computations of section 5.3.4, the penalty matrix L was taken to be the identity. Replace this L with the negative discrete Laplacian with Dirichlet boundary conditions, and repeat these computations.

5.32. Conduct a careful numerical study, similar to that presented in section 5.3.4, in which level 1 and level 2 block circulant preconditioners are replaced by block cosine preconditioners. See [15, 16] for implementation details.

Chapter 8

Total Variation Regularization

In section 1.3 of Chapter 1 we provided a very brief introduction to total variation regularization. In this chapter we take a closer look at both computational and theoretical issues.

8.1 Motivation

In a real analysis course [100], one sometimes sees the following definition of the total variation (TV) of a function f defined on the interval $[0, 1]$:

$$\mathrm{TV}(f) \stackrel{\text{def}}{=} \sup \sum_i |f(x_i) - f(x_{i-1})|, \tag{8.1}$$

where the supremum is taken over all partitions $0 = x_0 < x_1 < \cdots < x_n = 1$ of the interval. If f is piecewise constant with a finite number of jump discontinuities, then $\mathrm{TV}(f)$ gives the sum of magnitudes of the jumps. If f is smooth, one can multiply and divide the right-hand side of (8.1) by $\Delta x_i = x_i - x_{i-1}$ and take the limit as the $\Delta x_i \to 0$ to obtain the representation

$$\mathrm{TV}(f) = \int_0^1 \left| \frac{df}{dx} \right| dx. \tag{8.2}$$

An obvious generalization of (8.2) to two space dimensions is

$$\mathrm{TV}(f) = \int_0^1 \int_0^1 |\nabla f| \, dx \, dy, \tag{8.3}$$

where $\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$ denotes the gradient and $|(x, y)| = \sqrt{x^2 + y^2}$ denotes the Euclidean norm. An extension of this representation, valid even when f is not smooth, is

$$\mathrm{TV}(f) = \sup_{\vec{v} \in \mathcal{V}} \int_0^1 \int_0^1 f(x, y) \, \mathrm{div}\, \vec{v} \, dx \, dy, \tag{8.4}$$

where $\mathcal{V}$ consists of vector-valued functions $\vec{v} = (v_1(x, y), v_2(x, y))$ whose Euclidean norm is bounded by 1 and whose components v_i are continuously differentiable and vanish on the boundary of the unit square. $\mathrm{div}\, \vec{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}$ gives the divergence of $\vec{v}$. We will take (8.4) to be the definition of total variation; see [45] and section 8.4.

From the expression (8.4), one can develop theoretical properties of TV. For instance, one can establish that minimization of the Tikhonov–TV functional

$$T_\alpha(f) = \frac{1}{2}||Kf - g||^2 + \alpha\ \mathrm{TV}(f) \tag{8.5}$$

yields a regularization scheme in the sense of section 2.2 for the operator equation $Kf = g$; see [2] and section 8.4.

$\mathrm{TV}(f)$ can be interpreted geometrically as the lateral surface area of the graph of f. In particular, let S be a region with a smooth boundary ∂S contained within the unit square. Take $f(x, y) = H > 0$ for (x, y) in the interior of S and $f(x, y) = 0$ in the exterior. $\mathrm{TV}(f)$ is then the length of the boundary ∂S multiplied by the height H of the jump in f. For example, if S is the disk of radius $1/4$ centered at $(1/2, 1/2)$, then $\mathrm{TV}(f) = 2\pi \times 1/4 \times H$. With this geometric insight, one can begin to understand why total variation is an effective regularization functional. If f has many large amplitude oscillations, then it has large lateral surface area, and hence $\mathrm{TV}(f)$ is large. This is a property that TV shares with the more standard Sobolev H^1 "squared norm of the gradient" regularization functionals; see (2.47). Unlike the H^1 functional, with total variation one can effectively reconstruct functions with jump discontinuities. This is illustrated in one dimension in Figure 1.5. In two-dimensional image deblurring, total variation regularization tends to produce qualitatively correct reconstructions of blocky images [34]. By blocky, we mean the image is nearly piecewise constant with jump discontinuities, and the length of the curves on which the discontinuities occur is relatively small. The image in Figure 5.2 is blocky. The reconstruction in Figure 8.1, obtained with total variation regularization, does a much better job of preserving this blocky structure than do the reconstructions in Figure 5.3, which were generated using conventional regularization techniques.

We now turn our attention to numerical implementation.

8.2 Numerical Methods for Total Variation

We wish to obtain regularized solutions to operator equations $Kf = g$. In principle, this can be done by minimizing the Tikhonov–TV functional (8.5). However, the representations (8.2) and (8.3) are not suitable for the implementation of the numerical methods of Chapter 3, due to the nondifferentiability of the Euclidean norm at the origin. To overcome this difficulty, one can take an approximation to the Euclidean norm $|\mathbf{x}|$ like $\sqrt{|\mathbf{x}|^2 + \beta^2}$, where β is a small positive parameter. This yields the following approximation to $\mathrm{TV}(f)$, valid for a smooth function f defined on the unit interval in one dimension:

$$J_\beta(f) = \int_0^1 \sqrt{\left(\frac{df}{dx}\right)^2 + \beta^2}\, dx. \tag{8.6}$$

In two space dimensions, this becomes

$$J_\beta(f) = \int_0^1 \int_0^1 \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 + \beta^2}\, dx\, dy. \tag{8.7}$$

In the following section we consider minimization of the functional

$$T(\mathbf{f}) = \frac{1}{2}||K\mathbf{f} - \mathbf{d}||^2 + \alpha J(\mathbf{f}), \tag{8.8}$$

where J is a discretization of an approximation to the one-dimensional TV functional like (8.6), $\mathbf{d}$ represents discrete data, and K is a matrix; see the example in section 1.1.

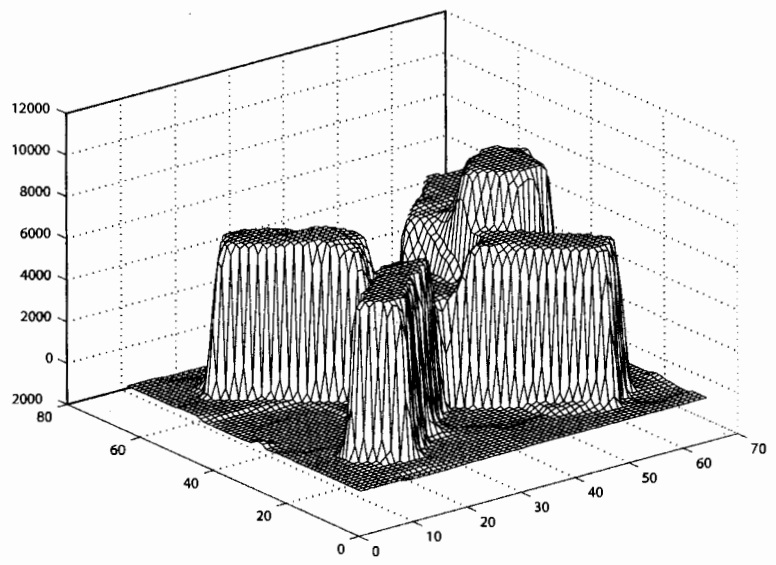


Figure 8.1. *Reconstructed image obtained with total variation regularization. The data for this reconstruction are described in section* 5.1.1. *A comparison with reconstructions in Figure* 5.3 *obtained with standard regularization techniques clearly shows that edge discontinuities and blocky structures are much better preserved with total variation.*

8.2.1 A One-Dimensional Discretization

To make the presentation less abstract, suppose $f(x)$ is a smooth function defined on the unit interval in $\mathbb{R}^1$ and $\mathbf{f} = (f_0, \ldots, f_n)$ with $f_i \approx f(x_i)$, $x_i = i\Delta x$, $\Delta x = 1/n$. Take the derivative approximation

$$D_i\mathbf{f} = (f_i - f_{i-1})/\Delta x, \qquad i = 1, \ldots, n. \tag{8.9}$$

Note the $(n+1) \times 1$ matrix representation, $D_i = [0, \ldots, 0, -1/\Delta x, 1/\Delta x, 0, \ldots, 0]$.

We assume a discretized penalty functional of the form

$$J(\mathbf{f}) = \frac{1}{2}\sum_{i=1}^{n} \psi\left((D_i\mathbf{f})^2\right)\,\Delta x, \tag{8.10}$$

where ψ is a smooth approximation to twice the square root function with the property

$$\psi'(t) > 0 \quad \text{whenever} \quad t > 0. \tag{8.11}$$

For example, the choice

$$\psi(t) = 2\sqrt{t + \beta^2} \tag{8.12}$$

leads to an approximation to (8.6). Another example [33] is

$$\psi(t) = \begin{cases} \frac{t}{\epsilon}, & t \le \epsilon^2, \\ 2\sqrt{t} - \epsilon, & t > \epsilon^2. \end{cases} \tag{8.13}$$

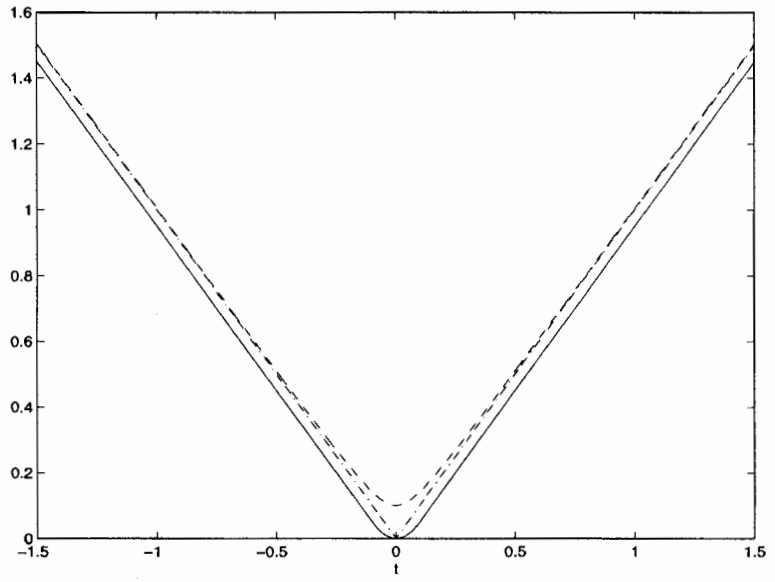


Figure 8.2. *Smooth approximations to the absolute value function. The dot-dashed curve represents the absolute value function; the solid curve represents the Huber function* $\varphi_\epsilon(t) = \psi_\epsilon(t^2)/2$ *(see* (8.13)*) with parameter* $\epsilon = 0.1$*; and the dashed curve represents the approximation* $\varphi_\beta(t) = \sqrt{t^2 + \beta^2}$ *(see* (8.12)*) with parameter* $\beta = 0.1$.

The composite function $\frac{1}{2}\psi(t^2)$ is the well-known Huber function from robust statistics. See Figure 8.2 for plots of (8.12) and (8.13).

To minimize (8.8) using the optimization techniques of Chapter 3, we need the gradient of J. For any $\mathbf{v} \in \mathbb{R}^{n+1}$,

$$\frac{d}{d\tau} J(\mathbf{f} + \tau\mathbf{v}) = \sum_{i=1}^{n} \psi'\left([D_i\mathbf{f}]^2\right)\ (D_i\mathbf{f})(D_i\mathbf{v})\Delta x \tag{8.14}$$

$$= \Delta x\ (D\mathbf{v})^T\ \mathrm{diag}(\psi'(\mathbf{f}))\ (D\mathbf{f})$$

$$= \langle \Delta x\ D^T\ \mathrm{diag}(\psi'(\mathbf{f}))\ D\mathbf{f}, \mathbf{v}\rangle, \tag{8.15}$$

where $\mathrm{diag}(\psi'(\mathbf{f}))$ denotes the $n \times n$ diagonal matrix whose ith diagonal entry is $\psi'\left((D_i\mathbf{f})^2\right)$, D is the $n \times (n+1)$ matrix whose ith row is D_i (see (8.9)), and $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product on $\mathbb{R}^{n+1}$. From this we obtain the gradient

$$\mathrm{grad}\, J(\mathbf{f}) = L(\mathbf{f})\mathbf{f}, \tag{8.16}$$

where

$$L(\mathbf{f}) = \Delta x\ D^T\ \mathrm{diag}(\psi'(\mathbf{f}))\ D \tag{8.17}$$

is a symmetric $(n+1) \times (n+1)$ matrix. $L(\mathbf{f})$ is positive semidefinite provided condition (8.11) holds.

To obtain the Hessian of J, from (8.14),

$$\frac{\partial^2 J}{\partial\tau\partial\xi}(\mathbf{f}+\tau\mathbf{v}+\xi\mathbf{w})|_{\tau,\xi=0} = \sum_{i=1}^{n}\psi'([D_i\mathbf{f}]^2)(D_i\mathbf{w})(D_i\mathbf{v})\Delta x$$

$$+\sum_{i=1}^{n}\psi''\left([D_i\mathbf{f}]^2\right)(D_i\mathbf{f})(D_i\mathbf{v})\,2(D_i\mathbf{f})(D_i\mathbf{w})\Delta x$$

$$= \langle \Delta x\,[\mathrm{diag}(\psi'(\mathbf{f})) + \mathrm{diag}(2(D\mathbf{f})^2\psi_i''(\mathbf{f}))]\,D\mathbf{v},\, D\mathbf{w}\rangle, \tag{8.18}$$

where $\mathrm{diag}(2(D\mathbf{f})^2\psi''(\mathbf{f}))$ denotes the $n \times n$ diagonal matrix whose ith diagonal entry is $2(D_i\mathbf{f})^2\,\psi''\left([D_i\mathbf{f}]^2\right)$. Consequently,

$$\mathrm{Hess}\, J(\mathbf{f}) = L(\mathbf{f}) + L'(\mathbf{f})\mathbf{f}, \tag{8.19}$$

where $L(\mathbf{f})$ is given in (8.17) and

$$L'(\mathbf{f})\mathbf{f} = \Delta x\; D^T\,\mathrm{diag}(2(D\mathbf{f})^2\psi''(\mathbf{f}))\; D. \tag{8.20}$$

From (8.8) and (8.16)–(8.17), we obtain the gradient of the penalized least squares cost functional,

$$\mathrm{grad}\, T(\mathbf{f}) = K^T(K\mathbf{f}-\mathbf{d}) + \alpha L(\mathbf{f})\mathbf{f}. \tag{8.21}$$

From this and (8.19)–(8.20), we obtain the Hessian,

$$\mathrm{Hess}\; T(\mathbf{f}) = K^T K + \alpha L(\mathbf{f}) + \alpha L'(\mathbf{f})\mathbf{f}. \tag{8.22}$$

8.2.2 A Two-Dimensional Discretization

We now consider minimization of the penalized least squares functional (8.8), where J is a discretization of a two-dimensional total variation approximation like (8.7). The matrix K is a discretization of a linear operator which acts on functions of two variables, and the vector $\mathbf{d}$ denotes discrete data. See, for example, section 5.1.

Suppose $f = f_{ij}$ is defined on an equispaced grid in two space dimensions, $\{(x_i, y_j) \mid x_i = i\Delta x, y_j = j\Delta y,\ i = 0,\ldots,n_x, j = 0,\ldots,n_y\}$. In a manner analogous to the one-dimensional case, we define the discrete penalty functional $J : \mathbb{R}^{(n_x+1)\times(n_y+1)} \to \mathbb{R}$ by

$$J(f) = \frac{1}{2}\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\psi\left((D_{ij}^x f)^2 + (D_{ij}^y f)^2\right), \tag{8.23}$$

where

$$D_{ij}^x f = \frac{f_{i,j} - f_{i-1,j}}{\Delta x}, \qquad D_{ij}^y f = \frac{f_{i,j} - f_{i,j-1}}{\Delta y}. \tag{8.24}$$

To simplify notation, we dropped a factor of $\Delta x\,\Delta y$ from the right-hand side of (8.23). This factor can be absorbed in the regularization parameter α in (8.8). Gradient computations are similar to those in one dimension:

$$\frac{d}{d\tau}J(f+\tau v)|_{\tau=0} = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\psi'_{ij}\left[(D_{ij}^x f)(D_{ij}^x v) + (D_{ij}^y f)(D_{ij}^y v)\right], \tag{8.25}$$

where $\psi'_{ij} = \psi'((D_{ij}^x f)^2 + (D_{ij}^y f)^2)$.

Now let $\mathbf{f} = \mathbf{vec}(f)$ and $\mathbf{v} = \mathbf{vec}(v)$, corresponding to lexicographical column ordering of the two-dimensional array components (see Definition 5.25); let D_x and D_y denote the resulting $n_x n_y \times (n_x + 1)(n_y + 1)$ matrices corresponding to the grid operators in (8.24); let $\mathrm{diag}(\psi'(\mathbf{f}))$ denote the $n_x n_y \times n_x n_y$ diagonal matrix whose diagonal entries are the ψ'_{ij}s; and let $\langle \cdot, \cdot \rangle$ denote the Euclidean inner product on $\mathbb{R}^{(n_x+1)(n_y+1)}$. Then

$$\frac{d}{d\tau} J(f + \tau v)|_{\tau=0} = \langle \mathrm{diag}(\psi'(\mathbf{f}))\, D_x\mathbf{f}, D_x\mathbf{v} \rangle + \langle \mathrm{diag}(\psi'(\mathbf{f})) D_y\mathbf{f}, D_y\mathbf{v} \rangle.$$

From this we obtain a gradient representation (8.16), but now

$$\begin{aligned} L(\mathbf{f}) &= D_x^T \,\mathrm{diag}(\psi'(\mathbf{f}))\, D_x + D_y^T \,\mathrm{diag}(\psi'(\mathbf{f}))\, D_y \\ &= [D_x^T \; D_y^T] \begin{bmatrix} \mathrm{diag}(\psi'(\mathbf{f})) & 0 \\ 0 & \mathrm{diag}(\psi'(\mathbf{f})) \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix}. \end{aligned} \tag{8.26}$$

Remark 8.1. The matrix $L(\mathbf{f})$ can be viewed as a discretization of a steady-state diffusion operator

$$\begin{aligned} \mathcal{L}(f)u &= -\nabla \cdot \left(\psi' \, \nabla u\right) \\ &= -\frac{\partial}{\partial x}\left(\psi' \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\psi' \frac{\partial u}{\partial y}\right) \end{aligned} \tag{8.27}$$

with the diffusion coefficient

$$\psi' = \psi'(|\nabla f|^2) = \psi'\left(\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right)$$

and with "natural" (homogeneous Neumann) boundary conditions. Expression (8.27) gives the directional derivative in the direction u of the functional

$$J(f) = \frac{1}{2}\int_0^1 \int_0^1 \psi(|\nabla f|^2)\, dx\, dy. \tag{8.28}$$

An alternative approach to obtain the discrete operator $L(\mathbf{f})$ is to apply a discretization scheme directly to the continuous operator $\mathcal{L}(f)$ in (8.27). An example is the cell-centered finite difference scheme utilized in [120].

As in the one-dimensional case, one can compute a representation (8.19) for the Hessian of the penalty functional, with $L(\mathbf{f})$ given in (8.26) and

$$L'(\mathbf{f})\mathbf{f} = [D_x^T \; D_y^T] \begin{bmatrix} \mathrm{diag}(2(D_x\mathbf{f})^2\psi''(\mathbf{f})) & \mathrm{diag}(2(D_x\mathbf{f})(D_y\mathbf{f})\psi''(\mathbf{f})) \\ \mathrm{diag}(2(D_y\mathbf{f})(D_x\mathbf{f})\psi''(\mathbf{f})) & \mathrm{diag}(2(D_y\mathbf{f})^2\psi''(\mathbf{f})) \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix}. \tag{8.29}$$

8.2.3 Steepest Descent and Newton's Method for Total Variation

In either the one- or the two-dimensional case, the gradient of the regularized cost functional has the form (8.21). To minimize (8.8), Algorithm 3.1 then gives us the following.

Algorithm 8.2.1. Steepest Descent for Total Variation–Penalized Least Squares.

$\nu := 0;$
$\mathbf{f}_0 :=$ initial guess;
begin steepest descent iterations
$\mathbf{g}_\nu := K^T(K\mathbf{f}_\nu - \mathbf{d}) + \alpha\, L(\mathbf{f}_\nu)\mathbf{f}_\nu;$ *% gradient*
$\tau_\nu := \arg\min_{\tau>0} T(\mathbf{f}_\nu - \tau\, \mathbf{g}_\nu);$ *% line search*
$\mathbf{f}_{\nu+1} := \mathbf{f}_\nu - \tau_\nu \mathbf{g}_\nu;$ *% update approximate solution*
$\nu := \nu + 1;$
end steepest descent iterations

Remark 8.2. Algorithm 8.2.3 is very similar to the discretized artificial time evolution approach of Rudin, Osher, and Fatemi [101]. In principle, to obtain a regularized solution to the operator equation $Kf = g$, they computed a steady-state solution of the time-dependent diffusion equation

$$\frac{\partial f}{\partial t} = -\alpha \mathcal{L}(f) f - K^*(Kf - g),$$

where $\mathcal{L}(f)$ is given in (8.27). After spatial discretization, they used explicit time marching with a fixed time step $\tau = \Delta t$ in place of the line search parameter τ_ν. See Exercise 8.4.

In both the one- and two-dimensional cases, the Hessian of the total variation–penalized least squares functional (8.8) has form (8.22). What follows is an implementation of Newton's method (section 3.3) to minimize (8.8). Some sort of globalization is essential to guarantee convergence of the Newton iterates [119]. Here we incorporate a line search.

Algorithm 8.2.2. Newton's Method for Total Variation–Penalized Least Squares.

$\nu := 0;$
$f_0 :=$ initial guess;
begin primal Newton iterations
$\mathbf{g}_\nu := K^T(K\mathbf{f}_\nu - \mathbf{d}) + \alpha\, L(\mathbf{f}_\nu)\mathbf{f}_\nu;$ *% gradient*
$H_J := L(\mathbf{f}_\nu) + L'(\mathbf{f}_\nu)\mathbf{f}_\nu;$ *% Hessian of penalty functional*
$H := K^T K + \alpha\, H_J;$ *% Hessian of cost functional*
$\mathbf{s}_\nu := -H^{-1}\mathbf{g}_\nu;$ *% Newton step*
$\tau_\nu := \arg\min_{\tau>0} T(\mathbf{f}_\nu + \tau\, \mathbf{s}_\nu);$ *% line search*
$\mathbf{f}_{\nu+1} := \mathbf{f}_\nu + \tau_\nu \mathbf{s}_\nu;$ *% update approximate solution*
$\nu := \nu + 1;$
end primal Newton iterations

8.2.4 Lagged Diffusivity Fixed Point Iteration

An alternative to the steepest descent method and Newton's method for the minimization of (8.8) is the lagged diffusivity fixed point iteration [119]:

$$\mathbf{f}_{\nu+1} = [K^T K + \alpha\, L(\mathbf{f}_\nu)]^{-1} K^T \mathbf{d} \tag{8.30}$$

$$= \mathbf{f}_\nu - [K^T K + \alpha\, L(\mathbf{f}_\nu)]^{-1} \operatorname{grad} T(\mathbf{f}_\nu). \tag{8.31}$$

The fixed point form (8.30) can be derived by first setting $\operatorname{grad} T(\mathbf{f}) = \mathbf{0}$ to obtain $(K^T K + \alpha\, L(\mathbf{f}))\mathbf{f} = K^T \mathbf{d}$; see (8.21). The discretized diffusion coefficient $\psi'(\mathbf{f})$ is then evaluated

at $\mathbf{f}_\nu$ to obtain $L(\mathbf{f}_\nu)$; see expressions (8.17) and (8.26) and Remark 8.1. Hence the expression "lagged diffusivity." The equivalent quasi-Newton form (8.31) can also be derived by dropping the term $\alpha\, L'(\mathbf{f})\mathbf{f}$ from the Hessian; see (8.22).

The following algorithm is based on the quasi-Newton form (8.31). The quasi-Newton form tends to be less sensitive to roundoff error than the fixed point form (8.30).

Algorithm 8.2.3. Lagged Diffusivity Fixed Point Method for Total Variation–Penalized Least Squares.

$\nu := 0;$
$\mathbf{f}_0 :=$ initial guess;
begin fixed point iterations
$L_\nu := L(\mathbf{f}_\nu);$ *% discretized diffusion operator*
$\mathbf{g}_\nu := K^T(K\mathbf{f}_\nu - \mathbf{d}) + \alpha\, L_\nu \mathbf{f}_\nu;$ *% gradient*
$H = K^T K + \alpha\, L_\nu;$ *% approximate Hessian*
$\mathbf{s}_{\nu+1} := -H^{-1}\mathbf{g}_\nu;$ *% quasi–Newton step*
$\mathbf{f}_{\nu+1} := \mathbf{f}_\nu + \mathbf{s}_\nu;$ *% update approximate solution*
$\nu := \nu + 1;$
end fixed point iterations

Remark 8.3. If $K^T K$ is positive definite, one can rigorously prove that this fixed point iteration converges globally [4, 41, 13, 120, 19], so no line search is needed. The approximate Hessian differs from the true Hessian (8.22) by the term $\alpha L'(\mathbf{f}_\nu)\mathbf{f}_\nu$. This term does not typically vanish as the iteration proceeds, so the rate of convergence of the lagged diffusivity iteration should be expected to be linear.

8.2.5 A Primal-Dual Newton Method

We first recall some basic results from convex analysis. In this discussion, φ is a convex functional defined on a convex set $C \subset \mathbb{R}^d$. For our purposes, $d = 1$ or 2, $C = \mathbb{R}^d$, and

$$\varphi(\mathbf{x}) = \frac{1}{2}\psi(|\mathbf{x}|^2) \tag{8.32}$$

with $|\mathbf{x}|^2 = \mathbf{x}^T\mathbf{x} = \sum_{i=1}^d x_i^2$. Relevant examples of the function ψ include $\psi(t) = 2\sqrt{t}$, which yields $\varphi(\mathbf{x}) = |\mathbf{x}|$, and the approximations (8.12) and (8.13).

Definition 8.4. The conjugate set C^* is defined by

$$C^* = \left\{ \mathbf{y} \in \mathbb{R}^d \mid \sup_{\mathbf{x}\in C}[\mathbf{x}^T\mathbf{y} - \varphi(\mathbf{x})] < \infty \right\}, \tag{8.33}$$

and the corresponding conjugate functional to φ is

$$\varphi^*(\mathbf{y}) = \sup_{\mathbf{x}\in C}\{\mathbf{x}^T\mathbf{y} - \varphi(\mathbf{x})\}. \tag{8.34}$$

This functional, which is also known as the Fenchel transform of φ, has the conjugate set C^* as its domain.

One can show [79, Proposition 1, p. 196] that the conjugate set C^* and the conjugate functional φ^* are, respectively, a convex set and a convex functional. The corresponding

second conjugates are defined in the obvious manner:

$$\varphi^{**} = (\varphi^*)^* \quad \text{and} \quad \mathcal{C}^{**} = (\mathcal{C}^*)^*.$$

In our finite dimensional Hilbert space setting, one can show [79, Proposition 2, p. 198] that $\varphi^{**} = \varphi$ and $\mathcal{C}^{**} = \mathcal{C}$. Consequently, from (8.34) we obtain the dual representation

$$\varphi(\mathbf{x}) = \sup_{y \in \mathcal{C}^*} \{\mathbf{x}^T \mathbf{y} - \varphi^*(\mathbf{y})\}. \tag{8.35}$$

We now derive the dual representation of the Euclidean norm, $\varphi(\mathbf{x}) = |\mathbf{x}|$, on $\mathbb{R}^d$. This is used in section 8.4 to define the TV functional. By the Cauchy–Schwarz inequality,

$$\mathbf{x}^T \mathbf{y} - |\mathbf{x}| \le (|\mathbf{y}| - 1)|\mathbf{x}|, \tag{8.36}$$

with equality if and only if $\mathbf{y} = c\mathbf{x}$ for some $c \in \mathbb{R}$. If $|\mathbf{y}| > 1$, one can make (8.36) arbitrarily large by taking $\mathbf{y} = c\mathbf{x}$ and letting c increase. If $|\mathbf{y}| \le 1$, then (8.36) is zero or negative, and its maximum value of zero is attained for $\mathbf{x} = 0$. Hence

$$\sup_{\mathbf{x} \in \mathbb{R}^d} \{\mathbf{x}^T \mathbf{y} - |\mathbf{x}|\} = \begin{cases} 0, & |\mathbf{y}| \le 1, \\ +\infty, & |\mathbf{y}| \ge 1. \end{cases}$$

Thus the conjugate set is the unit ball,

$$\mathcal{C}^* = \{y \in \mathbb{R}^d \mid |y| \le 1\},$$

and the conjugate functional $\varphi^*(\mathbf{y}) = 0$ for each $\mathbf{y} \in \mathcal{C}^*$. The dual representation (8.35) then yields

$$|\mathbf{x}| = \sup_{|\mathbf{y}| \le 1} \mathbf{x}^T \mathbf{y}. \tag{8.37}$$

The following two examples give dual representations of convex approximations to the Euclidean norm derived from (8.12) and (8.13) via (8.32).

Example 8.5. Consider the convex functional

$$\varphi_\beta(\mathbf{x}) = \sqrt{|\mathbf{x}|^2 + \beta^2}, \qquad \beta > 0, \tag{8.38}$$

defined on $\mathcal{C} = \mathbb{R}^d$. One can show (see Exercise 8.7) that

$$\sup_{\mathbf{x} \in \mathbb{R}^d} \{\mathbf{x}^T \mathbf{y} - \varphi_\beta(\mathbf{x})\} = \begin{cases} -\beta\sqrt{1 - |\mathbf{y}|^2} & \text{if} \quad |\mathbf{y}| \le 1, \\ +\infty & \text{if} \quad |\mathbf{y}| > 1. \end{cases} \tag{8.39}$$

Hence, the conjugate set $\mathcal{C}^*$ is the unit ball in $\mathbb{R}^d$, $\varphi_\beta^*(\mathbf{y}) = -\beta\sqrt{1 - |\mathbf{y}|^2}$, and by (8.35),

$$\varphi_\beta(\mathbf{x}) = \sup_{|\mathbf{y}| \le 1} \left\{ \mathbf{x}^T \mathbf{y} + \beta\sqrt{1 - |\mathbf{y}|^2} \right\}. \tag{8.40}$$

Example 8.6. Consider the functional

$$\varphi_\epsilon(\mathbf{x}) = \begin{cases} \frac{|\mathbf{x}|^2}{2\epsilon} & \text{if} \quad |\mathbf{x}| \le \epsilon, \\ |\mathbf{x}| - \frac{\epsilon}{2} & \text{if} \quad |\mathbf{x}| > \epsilon, \end{cases} \tag{8.41}$$

defined on $C = \mathbb{R}^d$. The conjugate set C^* is again the unit ball, and the conjugate functional is given by

$$\varphi_\epsilon^*(\mathbf{y}) = \frac{\epsilon}{2}|\mathbf{y}|^2, \qquad \mathbf{y} \in C^*.$$

See Exercise 8.8. Consequently,

$$\varphi_\epsilon(\mathbf{x}) = \sup_{|\mathbf{y}| \le 1} \left\{ \mathbf{x}^T\mathbf{y} - \frac{\epsilon}{2}|\mathbf{y}|^2 \right\}. \tag{8.42}$$

The following theorem relates the gradient of a convex functional φ to the gradient of its conjugate φ^*. See [30, p. 290] for a proof.

Theorem 8.7. *Suppose that φ is differentiable in a neighborhood of $\mathbf{x}_0 \in C \subset \mathbb{R}^d$, and the mapping $F = \operatorname{grad}\varphi : \mathbb{R}^d \to \mathbb{R}^d$ is invertible in that neighborhood. Then φ^* is Frechet differentiable in a neighborhood of $\mathbf{y}_0 = \varphi(\mathbf{x}_0)$ with*

$$\operatorname{grad}\varphi^*(y) = F^{-1}(\mathbf{y}). \tag{8.43}$$

We now apply convex analysis to obtain a dual formulation for the two-dimensional penalty functional (8.23). Setting

$$\varphi(x_1, x_2) = \frac{1}{2}\psi(x_1^2 + x_2^2) \tag{8.44}$$

and employing the dual representation (8.35) with $\mathbf{y} = (u, v)$, we obtain

$$J(f) = \sum_{i,j} \sup_{(u_{ij}, v_{ij}) \in C^*} \{(D_{ij}^x f)u_{ij} + (D_{ij}^y f)v_{ij} - \varphi^*(u_{ij}, v_{ij})\}.$$

As in section 8.2.2, we stack the array components f_{ij}, u_{ij}, and v_{ij} into column vectors $\mathbf{f}$, $\mathbf{u}$, and $\mathbf{v}$; we let D_x and D_y be matrix representers for the grid operators D_{ij}^x and D_{ij}^y; and we let $\langle \cdot, \cdot \rangle$ denote Euclidean inner product. Then the penalty functional can be rewritten as

$$\begin{aligned} J(\mathbf{f}) &= \sup_{(\mathbf{u},\mathbf{v}) \in C^*} \{\langle D_x\mathbf{f}, \mathbf{u}\rangle + \langle D_y\mathbf{f}, \mathbf{v}\rangle - \langle \varphi^*(\mathbf{u}, \mathbf{v}), \mathbf{1}\rangle\} \\ &= \sup_{(\mathbf{u},\mathbf{v}) \in C^*} \tilde{J}(\mathbf{u}, \mathbf{v}, \mathbf{f}), \end{aligned} \tag{8.45}$$

where

$$\tilde{J}(\mathbf{u}, \mathbf{v}, \mathbf{f}) = \langle \mathbf{f}, D_x^T\mathbf{u} + D_y^T\mathbf{v}\rangle - \langle \varphi^*(\mathbf{u}, \mathbf{v}), \mathbf{1}\rangle;$$

$\mathbf{1}$ denotes the vector of 1's; and by $(\mathbf{u}, \mathbf{v}) \in C^*$ we mean each of the component pairs (u_{ij}, v_{ij}) lies in C^*.

Minimization of the penalized least squares functional (8.8) is equivalent to computing the saddle point

$$(\mathbf{u}^*, \mathbf{v}^*, \mathbf{f}^*) = \arg\min_{\mathbf{f}} \max_{(\mathbf{u},\mathbf{v}) \in C^*} \tilde{T}(\mathbf{u}, \mathbf{v}, \mathbf{f}), \tag{8.46}$$

where

$$\tilde{T}(\mathbf{u}, \mathbf{v}, \mathbf{f}) = \frac{1}{2}||K\mathbf{f} - \mathbf{d}||^2 + \alpha\tilde{J}(\mathbf{u}, \mathbf{v}, \mathbf{f}).$$

We refer to $\mathbf{f}$ as the primal variable and to $\mathbf{u}$ and $\mathbf{v}$ as the dual variables.

Since (8.46) is unconstrained with respect to $\mathbf{f}$, a first order necessary condition for a saddle point is

$$\mathbf{0} = \operatorname{grad}_{\mathbf{f}} \tilde{T} = K^T(K\mathbf{f} - \mathbf{d}) + \alpha(D_x^T\mathbf{u} + D_y^T\mathbf{v}). \tag{8.47}$$

An additional necessary condition is that the duality gap in (8.35) must vanish, i.e., for each grid index i, j,

$$\varphi(D_{ij}^x\mathbf{f}, D_{ij}^y\mathbf{f}) = (D_{ij}^x\mathbf{f})u_{ij} + (D_{ij}^y\mathbf{f})v_{ij} - \varphi^*(u_{ij}, v_{ij}). \tag{8.48}$$

Finally, the dual variables must lie in the conjugate set; i.e.,

$$(u_{ij}, v_{ij}) \in \mathcal{C}^*. \tag{8.49}$$

We next examine the implications of (8.48). Suppose (8.48) holds for a point (u_{ij}, v_{ij}) in the interior of $\mathcal{C}^*$. This is the case in each of Examples 8.5 and 8.6; see Exercises 8.7 and 8.8. Then

$$\begin{aligned}(0,0) &= \operatorname{grad}_{u_{ij},v_{ij}}[(D_{ij}^x\mathbf{f})u_{ij} + (D_{ij}^y\mathbf{f})v_{ij} - \varphi^*(u_{ij}, v_{ij})] \\ &= (D_{ij}^x\mathbf{f}, D_{ij}^y\mathbf{f}) - \operatorname{grad}\varphi^*(u_{ij}, v_{ij}) \\ &= (D_{ij}^x\mathbf{f}, D_{ij}^y\mathbf{f}) - \frac{1}{\psi'((D_{ij}^x\mathbf{f})^2 + (D_{ij}^y\mathbf{f})^2)}(u_{ij}, v_{ij}). \end{aligned} \tag{8.50}$$

The last equality follows from the representation (8.44) and Theorem 8.7. Equation (8.50) is equivalent to

$$D_{ij}^x f = \frac{u_{ij}}{\psi'_{ij}}, \qquad D_{ij}^y f = \frac{v_{ij}}{\psi'_{ij}}, \tag{8.51}$$

where $\psi'_{ij} = \psi'_{ij}((D_{ij}^x\mathbf{f})^2 + (D_{ij}^y\mathbf{f})^2)$. Returning to matrix notation, we have

$$D_x\mathbf{f} = B(\mathbf{f})\,\mathbf{u}, \qquad D_y\mathbf{f} = B(\mathbf{f})\,\mathbf{v}, \tag{8.52}$$

where

$$B(\mathbf{f}) = \operatorname{diag}(1/\psi'(\mathbf{f})). \tag{8.53}$$

We can reformulate the first order necessary conditions (8.47), (8.52) as a nonlinear system

$$\mathbf{G}(\mathbf{u}, \mathbf{v}, \mathbf{f}) \stackrel{\text{def}}{=} \begin{bmatrix} B(\mathbf{f})\mathbf{u} - D_x\mathbf{f} \\ B(\mathbf{f})\mathbf{v} - D_y\mathbf{f} \\ \alpha D_x^T\mathbf{u} + \alpha D_y^T\mathbf{v} + K^T(K\mathbf{f} - \mathbf{d}) \end{bmatrix} = \mathbf{0}. \tag{8.54}$$

The derivative of G can be expressed as

$$\mathbf{G}'(\mathbf{u}, \mathbf{v}, \mathbf{f}) = \begin{bmatrix} B(\mathbf{f}) & \mathbf{0} & B'(\mathbf{f})\mathbf{u} - D_x \\ \mathbf{0} & B(\mathbf{f}) & B'(\mathbf{f})\mathbf{v} - D_y \\ \alpha D_x^T & \alpha D_y^T & K^TK \end{bmatrix}. \tag{8.55}$$

Here $B'(\mathbf{f})\mathbf{u}$ has component representation

$$[B'(\mathbf{f})\mathbf{u}]_{ij} = \frac{-\psi''_{ij}}{(\psi'_{ij})^2}\, 2\,((D_{ij}^x\mathbf{f})D_{ij}^x + (D_{ij}^y\mathbf{f})D_{ij}^y)\, u_{ij}.$$

In matrix form,

$$B'(\mathbf{f})\mathbf{u} - D_x = -E_{11}D_x - E_{12}D_y, \tag{8.56}$$

with

$$E_{11} = \operatorname{diag}\left(1 + \frac{2\,\psi''(\mathbf{f})\,(D_x\mathbf{f})\,\mathbf{u}}{\psi'(\mathbf{f})^2}\right), \qquad E_{12} = \operatorname{diag}\left(\frac{2\,\psi''(\mathbf{f})\,(D_y\mathbf{f})\,\mathbf{u}}{\psi'(\mathbf{f})^2}\right), \tag{8.57}$$

where the products and quotients are computed pointwise. Similarly,

$$B'(\mathbf{f})\mathbf{v} - D_y = -E_{21}D_x - E_{22}D_y, \tag{8.58}$$

with

$$E_{21} = \operatorname{diag}\left(\frac{2\,\psi''(\mathbf{f})\,(D_x\mathbf{f})\,\mathbf{v}}{\psi'(\mathbf{f})^2}\right), \qquad E_{22} = \operatorname{diag}\left(1 + \frac{2\,\psi''(\mathbf{f})\,(D_y\mathbf{f})\,\mathbf{v}}{\psi'(\mathbf{f})^2}\right). \tag{8.59}$$

Newton's method for the system (8.54) requires solutions of systems of the form $\mathbf{G}'(\mathbf{u},\mathbf{v},\mathbf{f})\ (\Delta\mathbf{u},\Delta\mathbf{v},\Delta\mathbf{f}) = -\mathbf{G}(\mathbf{u},\mathbf{v},\mathbf{f})$. Substituting (8.56)–(8.59) and applying block row reduction to convert $\mathbf{G}'$ to block upper triangular form, we obtain

$$\begin{bmatrix} B(\mathbf{f}) & \mathbf{0} & -E_{11}D_x - E_{12}D_y \\ \mathbf{0} & B(\mathbf{f}) & -E_{21}D_x - E_{22}D_y \\ \mathbf{0} & \mathbf{0} & K^TK + \alpha\overline{L} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{v} \\ \Delta\mathbf{f} \end{bmatrix} = \begin{bmatrix} -\mathbf{g}_1 \\ -\mathbf{g}_2 \\ \mathbf{r} \end{bmatrix}. \tag{8.60}$$

Here $\mathbf{g}_i$ denotes the ith component of $\mathbf{G}$ (see (8.54))

$$\begin{aligned} \overline{L} &= \begin{bmatrix} D_x^T & D_y^T \end{bmatrix} \begin{bmatrix} B(\mathbf{f})^{-1} & \mathbf{0} \\ \mathbf{0} & B(\mathbf{f})^{-1} \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} \begin{bmatrix} D_x \\ D_y \end{bmatrix} \\ &= D_x^T B(\mathbf{f})^{-1}E_{11}D_x + D_x^T B(\mathbf{f})^{-1}E_{12}D_y + D_y^T B(\mathbf{f})^{-1}E_{21}D_x \\ &\quad + D_y^T B(\mathbf{f})^{-1}E_{22}D_y \end{aligned} \tag{8.61}$$

and

$$\begin{aligned} \mathbf{r} &= -\mathbf{g}_3 + \alpha D_x^T B(\mathbf{f})^{-1}D_x\mathbf{g}_1 + \alpha D_y^T B(\mathbf{f})^{-1}D_y\mathbf{g}_2 \\ &= K^T(\mathbf{d} - K\mathbf{f}) - \alpha D_x^T B(\mathbf{f})^{-1}D_x\mathbf{f} - \alpha D_y^T B(\mathbf{f})^{-1}D_y\mathbf{f}. \end{aligned} \tag{8.62}$$

Consequently,

$$\Delta\mathbf{f} = [K^TK + \alpha\overline{L}]^{-1}\mathbf{r}, \tag{8.63}$$

$$\Delta\mathbf{u} = -\mathbf{u} + B(\mathbf{f})^{-1}[D_x\mathbf{f} + E_{11}D_x\Delta\mathbf{f} + E_{12}D_y\Delta\mathbf{f}], \tag{8.64}$$

$$\Delta\mathbf{v} = -\mathbf{v} + B(\mathbf{f})^{-1}[D_y\mathbf{f} + E_{21}D_x\Delta\mathbf{f} + E_{22}D_y\Delta\mathbf{f}]. \tag{8.65}$$

We employ backtracking to the boundary to maintain the constraint (8.49). In other words, we compute

$$\overline{\tau} = \max\{0 \le \tau \le 1 \mid (u_{ij} + \tau\Delta u_{ij}, v_{ij} + \tau\Delta v_{ij}) \in C^* \text{ for all } i, j\}.$$

We then update

$$\mathbf{u} := \mathbf{u} + \overline{\tau}\Delta\mathbf{u}, \qquad \mathbf{v} := \mathbf{v} + \overline{\tau}\Delta\mathbf{v}.$$

Practical experience [21] suggests that no globalization is needed in the update $\mathbf{f} := \mathbf{f} + \Delta\mathbf{f}$.

Algorithm 8.2.4. Primal-Dual Newton's Method for Total Variation–Penalized Least Squares Minimization in Two Space Dimensions.

$\nu := 0$;
$\mathbf{f}_0 :=$ initial guess for primal variable;
$\mathbf{u}_0, \mathbf{v}_0 :=$ initial guesses for dual variables;
begin primal-dual Newton iterations
$B_\nu^{-1} := \operatorname{diag}(\psi'(\mathbf{f}_\nu))$;
$\mathbf{w} := 2\psi'(\mathbf{f}_\nu)./\psi''(\mathbf{f}_\nu)$;
$E_{11} := \operatorname{diag}(\mathbf{w}. * (D_x\mathbf{f}_\nu). * \mathbf{u}_\nu)$;
$E_{12} := \operatorname{diag}(\mathbf{w}. * (D_y\mathbf{f}_\nu). * \mathbf{u}_\nu)$;
$E_{21} := \operatorname{diag}(\mathbf{w}. * (D_x\mathbf{f}_\nu). * \mathbf{v}_\nu)$;
$E_{22} := \operatorname{diag}(\mathbf{w}. * (D_y\mathbf{f}_\nu). * \mathbf{v}_\nu)$;
$\overline{L}_\nu := D_x^T B_\nu^{-1} E_{11} D_x + D_x^T B_\nu^{-1} E_{12} D_y + D_y^T B_\nu^{-1} E_{21} D_x$
$+ D_y^T B_\nu^{-1} E_{22} D_y$; % *discretized diffusion operator*
$\mathbf{r}_\nu := K^T(\mathbf{d} - K\mathbf{f}_\nu) - \alpha(D_x^T B_\nu^{-1} D_x + D_y^T B_\nu^{-1} D_y)\mathbf{f}_\nu$;
$\Delta\mathbf{f} := (K^T K + \alpha\,\overline{L}_\nu)^{-1}\mathbf{r}_\nu$; *% Newton step*
$\Delta\mathbf{u} := -\mathbf{u}_\nu + B_\nu^{-1}[D_x\mathbf{f}_\nu + (E_{11}D_x + E_{12}D_y)\Delta\mathbf{f}]$;
$\Delta\mathbf{v} := -\mathbf{u}_\nu + B_\nu^{-1}[D_y\mathbf{f}_\nu + (E_{21}D_x + E_{22}D_y)\Delta\mathbf{f}]$;
$\mathbf{f}_{\nu+1} := \mathbf{f}_\nu + \Delta\mathbf{f}$; *% update primal variable*
$\tau_\nu := \max\{0 \le \tau \le 1 \mid (\mathbf{u}_\nu + \tau\Delta\mathbf{u}, \mathbf{v}_\nu + \tau\Delta\mathbf{v}) \in C^*\}$;
$\mathbf{u}_{\nu+1} := \mathbf{u}_\nu + \tau_\nu\Delta\mathbf{u}$; *% update dual variables*
$\mathbf{v}_{\nu+1} := \mathbf{v}_\nu + \tau_\nu\Delta\mathbf{v}$;
$\nu := \nu + 1$;
end primal-dual Newton iterations

Remark 8.8. For large-scale systems where the matrix $K^T K$ does not have a sparse matrix representation, the most expensive part of the algorithm is the inversion of the matrix $A \stackrel{\text{def}}{=} K^T K + \alpha\overline{L}$ in the computation of the component $\Delta\mathbf{f}$ of the Newton step. If K has block Toeplitz structure, then the techniques of section 5.2.5 can be used to compute matrix-vector products $A\mathbf{v}$ at a cost of $n \log n$ (note that the matrix $\overline{L}$ is sparse; see (8.61)). This suggests the use of iterative linear solvers like the CG Algorithm 3.2. Use of CG is precluded by the fact that $\overline{L}$ need not be symmetric, but one can replace $\overline{L}$ by its symmetric part, $(\overline{L} + \overline{L}^T)/2$, and still retain quadratic convergence of the primal-dual Newton iteration [21]. CG iteration can then be applied as a linear solver. Chan, Chan, and Wong [15] provided preconditioners for CG in this setting.

8.2.6 Other Methods

The computational methods presented in the previous sections are based on smooth approximations to the Euclidean norm of the gradient. Ito and Kunisch [62] presented an alternative approach that is based on the representation (8.4).

One can also replace the Euclidean norm of the gradient by other norms. Li and Santosa [74] made use of the ℓ^1 norm. After discretization in two dimensions, their penalty functional took the form

$$J(f) = \sum_i \sum_j |D_{ij}^x f| + |D_{ij}^y f|.$$

They then applied an interior point method to solve their minimization problem. It should be noted that unlike the Euclidean norm, the ℓ^1 norm is not rotationally invariant. This may have the unfortunate consequence of making the reconstruction dependent on the orientation of the computational grid.

The time evolution approach outlined in Remark 8.2 provides one example of a broad class of nonlinear PDE-based techniques called nonlinear diffusion methods. These methods have found important applications in computer vision and image processing. See [124] for details and references.

Finally, we note that there exist several other mathematical expressions for variation that are closely related but not equivalent to (8.4). See [73] for details and references.

8.3 Numerical Comparisons

In this section, we compare the performance of some of the solution methods presented in the previous sections.

8.3.1 Results for a One-Dimensional Test Problem

The one-dimensional test problem is described in section 1.1, and the data used are presented in Figure 1.1. To solve this problem, we minimized the discrete regularized least squares functional (8.8)–(8.10) with

$$\frac{1}{2}\psi((D_i\mathbf{f})^2) = \sqrt{\left(\frac{f_i - f_{i-1}}{\Delta x}\right)^2 + \beta^2} \tag{8.66}$$

using the four iterative solution methods presented in sections 8.2.3–8.2.5.

The matrix K in (8.8) is Toeplitz and not sparse. In the primal-dual Newton implementation, the conjugate set C^* is the interval $-1 \leq u \leq 1$; see Example 8.5.

The reconstruction obtained using a one-dimensional version of the primal-dual Newton Algorithm 8.2.5 is shown in Figure 1.5. The functional (8.8) is strictly convex for this test example, so the other methods yield essentially the same reconstruction provided the minimizer is computed to sufficient accuracy.

One of our primary measures of numerical performance is the relative iterative solution error norm,

$$e_\alpha^\nu = \frac{||\mathbf{f}_\alpha^\nu - \mathbf{f}_\alpha||}{||\mathbf{f}_\alpha||}, \tag{8.67}$$

where $\mathbf{f}_\alpha$ represents the minimizer of (8.8) and $\mathbf{f}_\alpha^\nu$ represents the numerical approximation to $\mathbf{f}_\alpha$ at iteration ν. In place of the exact $\mathbf{f}_\alpha$ we used an extremely accurate approximation obtained with the primal-dual Newton method.

The performances of the steepest descent method (Algorithm 8.2.3) and Newton's method with a line search (Algorithm 8.2.3) are compared in Figure 8.3. Initially the steepest descent method exhibits a rapid decrease in the iterative solution error, but almost no change in the reconstructions is observed after the first five steepest descent iterations. This is consistent with Theorem 3.5, since the Hessian is quite ill-conditioned.

With Newton's method very little progress occurs until about iteration 50. During the earlier iterations, the line search restricts the step size. In the last six iterations, there is a dramatic decrease in solution error, as the local quadratic convergence rate characteristic of Newton's method is finally attained. This behavior is consistent with the theory presented in

section 3.3. Note that the convergence constant c_* in (3.18) becomes large as the minimum eigenvalue of the Hessian becomes small and the Lipschitz constant γ becomes large. The former event occurs when the regularization parameter α is small; the latter occurs when the parameter β in (8.66) becomes small. When c_* is large, $\mathbf{f}_\alpha^\nu$ must be quite close to $\mathbf{f}_\alpha$ before the iteration will converge without a line search.

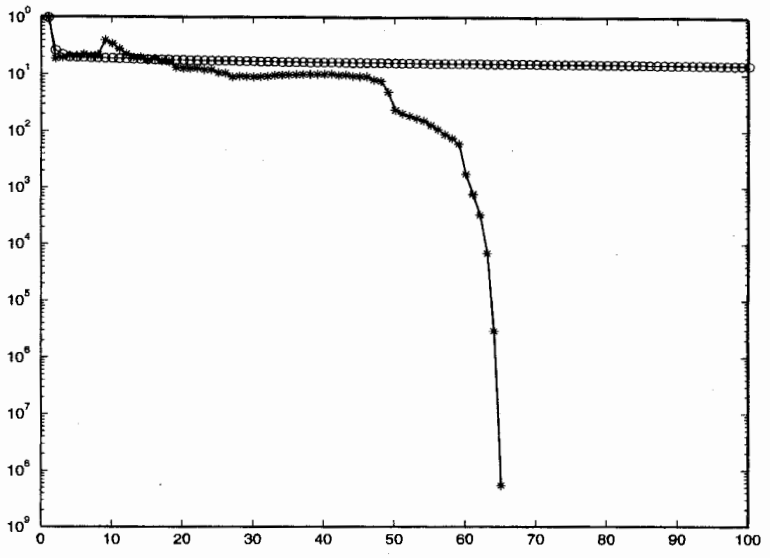


Figure 8.3. *Numerical performance of the steepest descent method and Newton's method on a one-dimensional test problem. The relative iterative solution error norm* (8.67) *is plotted against the iteration count. Circles represent the results for the steepest descent method (Algorithm* 8.2.3*), and asterisks represent the results for the primal Newton method (Algorithm* 8.2.3*).*

In Figure 8.4 the performance of the lagged diffusivity fixed point method (Algorithm 8.2.4) and the primal-dual Newton method (Algorithm 8.2.5) are compared. The lagged diffusivity fixed point method displays rapid decrease in the solution error during the first few iterations. Convergence then slows to a steady linear rate. The primal-dual Newton method also displays fairly fast initial convergence. After about eight iterations, a more rapid quadratic convergence rate can be seen.

Note that the steepest descent method requires a nonsparse matrix-vector multiplication at each iteration, since K is a full matrix. The other three methods require the inversion of nonsparse linear systems at each iteration. (We used Gaussian elimination to solve these systems.) Hence, the cost per iteration of the steepest descent method is significantly less than that of the other three methods. However, for this particular test problem, the extremely slow convergence rate of steepest descent negates the advantage of low computational cost per iteration. The other three methods all have roughly the same cost per iteration. Due to its more rapid convergence rate, the primal-dual Newton method is the most efficient method for this problem.

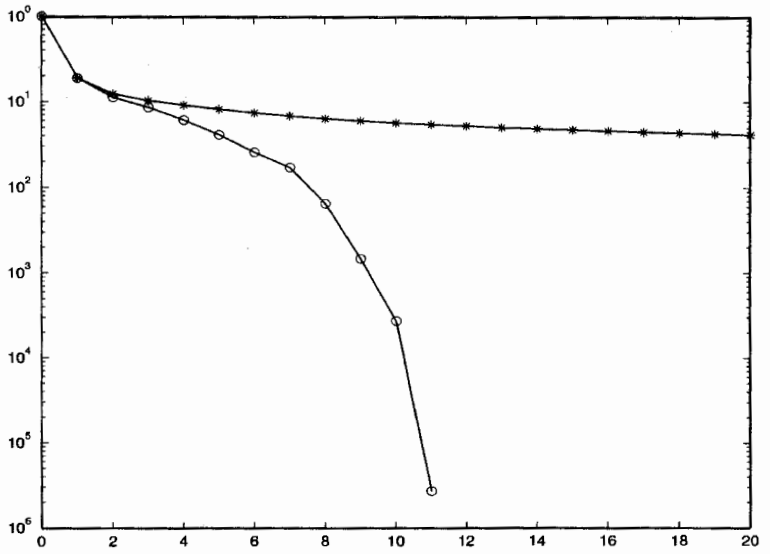


Figure 8.4. *Numerical performance of the lagged diffusivity fixed point iteration and the primal-dual Newton method on a one-dimensional test problem. Asterisks represent the relative iterative solution error norm for the fixed point method (Algorithm* 8.2.4*), and circles represent the results for the primal-dual Newton method (Algorithm* 8.2.5*).*

8.3.2 Two-Dimensional Test Results

The image deblurring test problem in this section is described in section 5.1.1, the test data are similar to that shown in Figure 5.2, and the reconstructions are similar to that shown in Figure 8.1. To obtain the reconstructions, we minimized a two-dimensional version of the penalized least squares functional (8.8)–(8.10) with

$$\frac{1}{2}\psi((D_{ij}^x \mathbf{f})^2 + (D_{ij}^y \mathbf{f})^2) = \sqrt{\left(\frac{f_{i,j} - f_{i-1,j}}{\Delta x}\right)^2 + \left(\frac{f_{i,j} - f_{i,j-1}}{\Delta x}\right)^2 + \beta^2}.$$

We present numerical performance results (see Figure 8.5) only for the lagged diffusivity fixed point Algorithm 8.2.4 and for the primal-dual Newton Algorithm 8.2.5. Comparison of the other methods is left to Exercise 8.14.

In the primal-dual Newton implementation, the conjugate set C^* is the unit ball in $\mathbb{R}^2$; see Example 8.5. The matrix K in (8.8) is block Toeplitz with Toeplitz blocks; see section 5.2.5. As in our one-dimensional test problem, K is not sparse.

As in one dimension, the lagged diffusivity fixed point convergence is rapid at first but slows to a steady linear rate after a few iterations. Primal-dual Newton displays fairly rapid initial convergence, with an increase in the convergence rate at later iterations. For this test problem, primal-dual Newton clearly converges at a much more rapid rate.

Both lagged diffusivity and primal-dual Newton require the solution of nonsparse linear systems at each iteration. In this two-dimensional application, these systems are large enough to discourage the use of direct matrix decomposition methods. Instead we applied the CG Algorithm 3.2 with no preconditioning. (See Remark 8.8 for primal-dual Newton

implementation details.) We found that we needed a very small (residual) CG stopping tolerance to maintain rapid convergence of the primal-dual Newton iterations. A much more relaxed CG stopping tolerance could be used without degrading the convergence of the lagged diffusivity iteration. Consequently, the cost per iteration of primal-dual Newton was significantly larger than the cost per iteration of lagged diffusivity fixed point. This may no longer be the case if preconditioning is applied; see [15] and Exercise 8.15. An overall cost comparison is difficult to carry out, since it depends on factors like stopping tolerances; values of parameters like α, β, and the system size; and the effectiveness of the preconditioner.

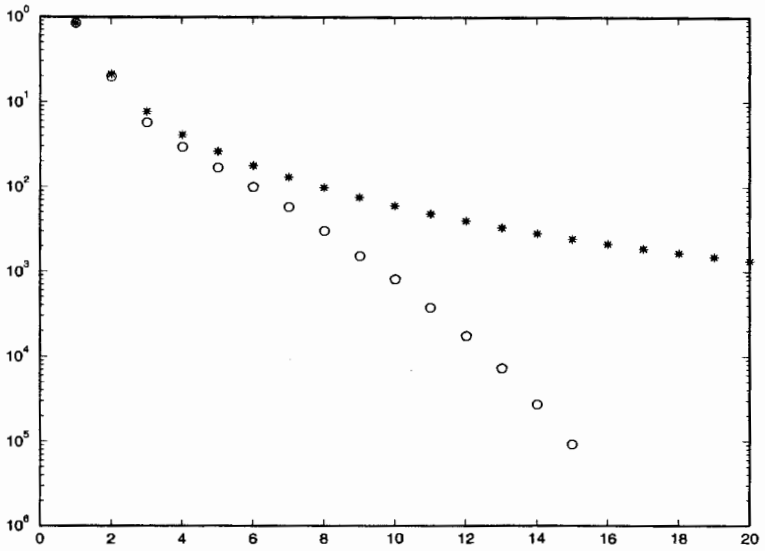


Figure 8.5. *Comparison of lagged diffusivity fixed point iteration and primal-dual Newton iteration for a two-dimensional image reconstruction problem. Asterisks denote relative iterative solution error for the fixed point iteration, and circles denote error for primal-dual Newton.*

Total variation methods have been applied to more general inverse problems. See [33] for an application to distributed parameter identification.

8.4 Mathematical Analysis of Total Variation

In this section, Ω denotes a simply connected, nonempty, open subset of $\mathbb{R}^d$, $d = 1, 2, \ldots$, with Lipschitz continuous boundary. In imaging applications, Ω is typically the unit square in $\mathbb{R}^2$. We use the symbol ∇ to denote the gradient of a smooth function $f : \mathbb{R}^d \to \mathbb{R}^1$, i.e., $\nabla f = (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d})$. $C_0^1(\Omega; \mathbb{R}^d)$ denotes the space of vector-valued functions $\vec{v} = (v_1, \ldots, v_d)$ whose component functions v_i are each continuously differentiable and compactly supported on Ω, i.e., each v_i vanishes outside some compact subset of Ω. The divergence of $\vec{v}$ is given by

$$\operatorname{div} \vec{v} = \sum_{i=1}^{d} \frac{\partial v_i}{\partial x_i}.$$

The Euclidean norm is denoted by $|\cdot|$. In particular, $|\vec{v}(x)| = [\sum_{i=1}^{d} v_i(x)^2]^{1/2}$. The Sobolev space $W^{1,1}(\Omega)$ denotes the closure of $C_0^1(\Omega)$ with respect to the norm

$$\|f\|_{1,1} = \int_\Omega \left[|f| + \sum_{i=1}^{d} \left| \frac{\partial f}{\partial x_i} \right| \right].$$

The following definition is taken from Giusti [45].

Definition 8.9. The total variation of a function $f \in L^1(\Omega)$ is defined by

$$\mathrm{TV}(f) = \sup_{\vec{v} \in \mathcal{V}} \int_\Omega f \ \mathrm{div}\ \vec{v}\, dx, \tag{8.68}$$

where the space of test functions

$$\mathcal{V} = \{\vec{v} \in C_0^1(\Omega; \mathbb{R}^d) \mid |\vec{v}(x)| \le 1 \ \text{ for all } \ x \in \Omega\}. \tag{8.69}$$

Remark 8.10. Equation (8.68) can be viewed as a weak form of

$$\mathrm{TV}(f) = \int_\Omega |\nabla f|\, dx.$$

Using the dual representation (8.37) for the Euclidean norm and formally applying integration by parts,

$$\begin{aligned} \int_\Omega |\nabla f|\, dx &= \int_\Omega \sup_{|\vec{v}| \le 1} \nabla f^T \vec{v}\, dx \\ &= \sup_{|\vec{v}| \le 1} \left[\int_{\partial\Omega} f\, \vec{v}^T \hat{n}\, dS - \int_\Omega f \ \mathrm{div}\ \vec{v}\, dx \right], \end{aligned}$$

where $\partial\Omega$ denotes the boundary of Ω and $\hat{n}$ denotes the outward unit normal to $\partial\Omega$. If $\vec{v}$ is compactly supported in Ω, then the boundary integral term vanishes. Note that $|\vec{v}| \le 1$ if and only if $|-\vec{v}| \le 1$, so we can drop the minus sign to obtain (8.68)–(8.69).

Example 8.11. Let $\Omega = [0, 1] \subset \mathbb{R}^1$, and define

$$f(x) = \begin{cases} f_0, & x < 1/2, \\ f_1, & x > 1/2, \end{cases}$$

where f_0, f_1 are constants. For any $v \in C_0^1[0, 1]$,

$$\int_0^1 f(x)v'(x)\, dx = \int_0^{1/2} f(x)v'(x)\, dx + \int_{1/2}^1 f(x)v'(x)\, dx = (f_0 - f_1)\, v(1/2).$$

This quantity is maximized over all $v \in \mathcal{V}$ when $v(1/2) = \mathrm{sign}(f_0 - f_1)$. This yields $\mathrm{TV}(f) = |f_1 - f_0|$, which agrees with (8.1).

Example 8.12. Let E be a set contained in $\Omega \subset \mathbb{R}^d$, with $d \ge 2$, and assume its boundary ∂E is C^2. Let $f(x) = f_0$ if $x \in E$ and $f(x) = 0$ otherwise. In this case,

$$\mathrm{TV}(f) = f_0\, \mathrm{Area}(\partial E),$$

where Area$(\cdot)$ denotes surface area. (This reduces to arc length when the dimension $d = 2$.) To verify, for any $\vec{v} \in C_0^1(\Omega; \mathbb{R}^d)$ the divergence theorem yields

$$\int_\Omega f \operatorname{div} \vec{v}\, dx = f_0 \int_E \operatorname{div} \vec{v}\, dx = f_0 \int_{\partial E} \vec{v}^T \hat{n}\, dS, \tag{8.70}$$

where $\hat{n}(x)$ denotes the outward unit normal to ∂E at x and dS denotes surface integration. Imposing $|\vec{v}(x)| \leq 1$, we obtain from (8.68) that $\mathrm{TV}(f) \leq f_0 \mathrm{Area}(\partial S)$. Since E has C^2 boundary, its outward unit normal $\hat{n}(x)$ will be a C^1 vector-valued function, which can be extended to a function $\vec{v} \in C_0^1(\Omega; \mathbb{R}^d)$ for which $|\vec{v}(x)| \leq 1$. Then by (8.68) and (8.70), $\mathrm{TV}(f) \geq f_0 \int_{\partial E} |\hat{n}(x)|^2\, dS = f_0 \mathrm{Area}(\partial S)$.

Proposition 8.13. *If $f \in W^{1,1}(\Omega)$, then*

$$\mathrm{TV}(f) = \int_\Omega |\nabla f|. \tag{8.71}$$

Proof. If $f \in C^1(\Omega)$ and $\vec{v} \in C_0^1(\Omega; \mathbb{R}^d)$, then integration by parts yields

$$\int_\Omega f \operatorname{div} \vec{v}\, dx = -\int_\Omega \nabla f^T \vec{v}\, dx.$$

Take

$$\vec{w}(x) = \begin{cases} -\frac{\nabla f}{|\nabla f|} & \text{if} \quad \nabla f(x) \neq 0, \\ 0 & \text{if} \quad \nabla f(x) = 0. \end{cases}$$

One can pick $\vec{v} \in C_0^1(\Omega; \mathbb{R}^d)$ with components arbitrarily close to those of $\vec{w}$ with respect to the L^2 norm, and, hence, (8.71) holds for any $f \in C^1(\Omega)$. By a standard denseness argument, this also holds for $f \in W^{1,1}(\Omega)$. □

Definition 8.14. The space of functions of bounded variation, denoted by $BV(\Omega)$, consists of functions $f \in L^1(\Omega)$ for which

$$\|f\|_{BV} \stackrel{\text{def}}{=} \|f\|_{L^1(\Omega)} + \mathrm{TV}(f) < \infty. \tag{8.72}$$

Theorem 8.15. *$\|\cdot\|_{BV}$ is a norm, and $BV(\Omega)$ is a Banach space under this norm. The TV functional is a seminorm on this space.*

See Giusti [45] for a proof of this theorem. Proposition 8.13 and Examples 8.11 and 8.12 show that $W^{1,1}(\Omega)$ is a proper subspace of $BV(\Omega)$.

The following three theorems pertain to the important properties of compactness, convexity, and semicontinuity. Proofs can be found in [2].

Theorem 8.16. *Let $\mathcal{S}$ be a BV-bounded set of functions. For $\Omega \subset \mathbb{R}^d$, $\mathcal{S}$ is a relatively compact subset of $L^p(\Omega)$ for $1 \leq p < d/(d-1)$ and is weakly relatively compact in $L^{d/(d-1)}(\Omega)$. In case the dimension $d = 1$, we set $d/(d-1) = +\infty$.*

Theorem 8.17. *The* TV *functional* (8.68)*, defined on the space $BV(\Omega)$, is convex but not strictly convex. The restriction of this functional to $W^{1,1}(\Omega)$ is strictly convex.*

Theorem 8.18. *The* TV *functional is weakly lower semicontinuous with respect to the L^p norm topology for $1 \leq p < \infty$.*

We next examine the existence, uniqueness, and stability of minimizers of the BV-penalized least squares functional

$$T(f) = ||Kf - d||^2_{L^2(\Omega)} + \alpha ||f||_{BV}, \qquad \alpha > 0. \tag{8.73}$$

Theorem 8.19. *Let $1 \leq p < d/(d-1)$, and let C be a closed, convex subset of $L^p(\Omega)$. Assume $K : L^p(\Omega) \to L^2(\Omega)$ is linear, bounded, and* Null$(K) = \{0\}$. *Then, for any fixed $d \in L^2(\Omega)$, the functional in* (8.73) *has a unique constrained minimizer,*

$$f_* = \arg\min_{f \in C} T(f).$$

Proof. Existence follows arguments similar to those of Theorem 2.30. See [2] for details. Note that since K is linear with a trivial null space and the squared Hilbert space norm is strictly convex, the mapping $f \mapsto ||Kf - d||^2_{L^2(\Omega)}$ is strictly convex. Uniqueness follows from strict convexity. □

The following stability result is proved in [2].

Theorem 8.20. *Suppose the hypotheses of Theorem* 8.19 *hold. Then the minimizer f_* is stable with respect to*

(i) *perturbations d_n of the data d for which $||d_n - d||_{L^2(\Omega)} \to 0$;*

(ii) *perturbations K_n of the operator K for which $||K_n(f) - K(f)||_{L^2(\Omega)} \to 0$ uniformly on compact subsets in $L^p(\Omega)$;*

(iii) *perturbations α_n of the regularization parameter $\alpha > 0$.*

Similar existence-uniqueness-stability results can be obtained when the BV norm (8.72) is replaced by the TV functional (8.68), yielding

$$T(f) = ||Kf - d||^2_{L^2(\Omega)} + \alpha \mathrm{TV}(f). \tag{8.74}$$

The condition that K has a trivial null space can also be weakened somewhat. The following result is an example. See [2] for a proof.

Theorem 8.21. *Let C be a closed, convex subset of $L^p(\Omega)$ with $1 \leq p < d/(d-1)$. Let $K : L^p(\Omega) \to L^2(\Omega)$ be linear and bounded. Assume that $K1 \neq 0$, where 1 denotes the function $1(x) = 1$ for all $x \in \Omega$. Then the functional in* (8.74) *has a unique constrained minimizer over C.*

8.4.1 Approximations to the TV Functional

As in section 8.2.5, we replace the Euclidean norm $|\cdot|$ by a smooth, convex approximation φ. For smooth f one can define a corresponding approximation to the TV functional,

$$J(f) = \int_\Omega \varphi(\nabla f)\, dx, \tag{8.75}$$

which is analogous to the representation (8.71).

To obtain an extension of the functional in (8.75) that is valid for nonsmooth f in a manner analogous to (8.68), we make use of the dual representation

$$J(f) = \sup_{\vec{v}\in\mathcal{V}} \int_\Omega [-f \text{ div } \vec{v} - \varphi^*(\vec{v}(x))]\, dx, \tag{8.76}$$

where

$$\mathcal{V} = \{\vec{v} \in C_0^1(\Omega; \mathbb{R}^d) \mid \vec{v}(x) \in C^* \text{ for all } x \in \Omega\}.$$

Equation (8.76) is obtained from (8.35) by replacing x with ∇f, replacing y with $\vec{v}(x)$, and integrating by parts; see Remark 8.10. Motivated by Examples 8.5 and 8.6, we define

$$J_\beta(f) = \sup_{\vec{v}\in\mathcal{V}} \int_\Omega \left[-f \text{ div } \vec{v} + \beta\sqrt{1 - |\vec{v}(x)|^2}\right] dx \tag{8.77}$$

and

$$J_\epsilon(f) = \sup_{\vec{v}\in\mathcal{V}} \int_\Omega \left[-f \text{ div } \vec{v} - \frac{\epsilon}{2}|\vec{v}(x)|^2\right] dx, \tag{8.78}$$

where $\mathcal{V}$ is given in (8.69).

The following results establish stability of total variation regularized solutions with respect to perturbations (8.77) and (8.78) of the TV functional as the parameters β and ϵ tend to zero. See [2] for proofs.

Proposition 8.22. *Both J_β and J_ϵ are convex and weakly lower semicontinuous. Moreover, $J_\beta(f) \to \text{TV}(f)$ as $\beta \to 0$ and $J_\epsilon(f) \to \text{TV}(f)$ as $\epsilon \to 0$, uniformly on BV-bounded sets.*

Theorem 8.23. *Total variation regularized solutions are stable with respect to certain perturbations in the penalty functional. In particular, if* $\text{TV}(f)$ *in* (8.74) *is replaced by either $J_\beta(f)$ or $J_\epsilon(f)$ and $\alpha > 0$ is fixed, then the corresponding regularized solutions $f_{\alpha,\beta}$ and $f_{\alpha,\epsilon}$ converge to the total variation regularized solution in L^p norm, $1 \le p < d/(d-1)$, as $\beta \to 0$ and $\epsilon \to 0$.*

Exercises

8.1. Prove that $L(\mathbf{f})$ in (8.17) is a positive semidefinite matrix. What is the null space of $L(\mathbf{f})$?

8.2. Let the functional J be as in (8.28). Recall from Remark 2.35 that its Gateaux, or directional, derivative at f in the direction h is given by

$$\delta J(f; h) = \frac{d}{d\tau} J(f + \tau h)|_{\tau=0}.$$

Show that for smooth f and h,

$$\delta J(f; h) = \int_0^1 \int_0^1 \psi'(|\nabla f|^2)\, \nabla f^T \nabla h \, dx\, dy.$$

Then show that $\delta J(f; h) = \langle \mathcal{L}(f) f, h\rangle$, provided that the normal derivative of f vanishes on the boundary of the unit square. Here $\mathcal{L}(f)$ is given in (8.27) and $\langle \cdot, \cdot \rangle$ denotes the L^2 inner product on the unit square.

8.3. Derive the two-dimensional representation for $L'(\mathbf{f})\mathbf{f}$ in (8.29).

8.4. Suppose that explicit time marching, or the forward Euler method, is applied to the system of ODEs:

$$\frac{d\mathbf{f}}{dt} = -\text{grad}\, T(f),$$

where the functional T is given in (8.8). Show that the resulting iteration is equivalent to that of the steepest descent Algorithm 8.2.3, except that the line search parameter $\tau_v = \Delta t$ is fixed.

8.5. Show that the right-hand side of (8.30) is equivalent to (8.31).

8.6. Verify (8.35) directly. *Hint:* Use the Cauchy–Schwarz inequality to show that the left-hand side is bounded by the right-hand side. Then show that the bound is attained.

8.7. Verify equation (8.39).

8.8. With φ_ϵ given in Example 8.6, verify that

$$\sup_{\mathbf{x}\in\mathbb{R}^d} \{\mathbf{x}^T\mathbf{y} - \varphi_\epsilon(\mathbf{x})\} = \begin{cases} \frac{\epsilon}{2}|\mathbf{y}|^2 & \text{if} \quad |\mathbf{y}| \le 1, \\ +\infty & \text{if} \quad |\mathbf{y}| > 1. \end{cases}$$

8.9. By applying block Gaussian elimination to the right-hand side of (8.55), derive the expression for $\overline{L}$ in (8.61). Also, derive (8.62).

8.10. For the matrix $\overline{L}$ in (8.61), prove that the symmetric part $(\overline{L} + \overline{L}^T)/2$ is positive semidefinite.

8.11. For the one-dimensional test problem of section 8.3.1, conduct a numerical study of the effects of varying the parameters α and β on the performance of each of the four algorithms applied in that section. In particular, what are the effects on numerical performance of making β very small?

8.12. What is the qualitative effect on the reconstructions in the one-dimensional test problem of *increasing* the parameter β?

8.13. For the one-dimensional test problem, replace the approximation (8.38) to the absolute value by (8.41). Explain why one cannot then implement either the primal Newton method or the primal-dual Newton method. Implement and compare results for the remaining two methods, the steepest descent method and the lagged diffusivity fixed point method.

8.14. For the two-dimensional test problem of section 8.3.2, implement both the steepest descent Algorithm 8.2.3 and the Newton Algorithm 8.2.3. How do these methods compare in terms of convergence rates and computational cost?

8.15. In the implementation of the lagged diffusivity fixed point method and the primal-dual Newton method for two-dimensional test problem, replace the CG linear solver with preconditioned CG. Use the level 2 block circulant preconditioner of section 5.3.3.

8.16. Prove Proposition 8.22. Use the facts that $\text{TV}(f) \le J_\beta(f) \le \text{TV}(f) + \beta\ \text{Vol}(\Omega)$ and $\text{TV}(f) - \frac{\epsilon}{2}\ \text{Vol}(\Omega) \le J_\epsilon(f) \le \text{TV}(f)$. Here $\text{Vol}(\Omega) = \int_\Omega dx$ denotes the volume of the set Ω.

Bibliography

[1] M. Abramowitz and I. Stegun, eds., *Handbook of Mathematical Functions*, Dover, New York, 1965.

[2] R. Acar and C. R. Vogel, *Analysis of total variation penalty methods*, Inverse Problems, **10** (1994), pp. 1217–1229.

[3] K. E. Atkinson, *An Introduction to Numerical Analysis,* 2nd Edition, Wiley, New York, 1989.

[4] G. Aubert, M. Barlaud, L. Blanc-Feraud, and P. Charbonnier, *Deterministic edge-preserving regularization in computed imaging*, Tech. report 94-01, Informatique, Signaux et Systemes de Sophia Antipolis, France, 1994.

[5] O. Axelsson and V. A. Barker, *Finite Element Solution of Boundary Value Problems*, Academic Press, New York, 1984.

[6] H. T. Banks and K. Kunisch, *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser Boston, Boston, 1989.

[7] M. Bertero and P. Boccacci, *Introduction to Inverse Problems in Imaging*, IOP Publishing, Bristol, UK, 1998.

[8] D. Bertsekas, *Projected Newton methods for optimization problems with simple constraints*, SIAM Journal on Control and Optimization, **20** (1982), pp. 221–246.

[9] A. Bjork, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[10] W. E. Boyce and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems,* 7th Edition, Wiley, New York, 2000.

[11] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on Scientific Computing, **16** (1995), pp. 1190–1208.

[12] G. Casella and R. L. Berger, *Statistical Inference*, Brooks/Cole, Pacific Grove, CA, 1990.

[13] A. Chambolle and P.-L. Lions, *Image recovery via total variation minimization and related problems*, Numerische Mathematik, **76** (1997), pp. 167–188.

[14] R. H. Chan and M. K. Ng, *Conjugate gradient method for Toeplitz systems*, SIAM Review, **38** (1996), pp. 427–482.

[15] R. H. Chan, T. F. Chan, and C. K. Wong, *Cosine transform based preconditioners for total variation deblurring*, IEEE Transactions on Image Processing, **8** (1999), pp. 1472–1478.

[16] R. H. Chan, M. K. Ng, and C. Wong, *Sine transform based preconditioners for symmetric Toeplitz systems*, Linear Algebra and Its Applications, **232** (1996), pp. 237–260.

[17] R. H. Chan, J. G. Nagy, and R. J. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM Journal on Numerical Analysis, **30** (1993), pp. 1740–1768.

[18] R. H. Chan, M. K. Ng, and R. J. Plemmons, *Generalization of Strang's preconditioner with applications to Toeplitz least squares problems*, Numerical Linear Algebra and Applications, **3** (1996), pp. 45–64.

[19] T. Chan and P. Mulet, *On the convergence of the lagged diffusivity fixed point method in total variation image restoration*, SIAM Journal on Numerical Analysis, **36** (1999), pp. 354–367.

[20] T. Chan and J. Olkin, *Circulant preconditioners for Toeplitz-block matrices*, Numerical Algorithms, **6** (1994), pp. 89–101.

[21] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for TV-based image restoration*, SIAM Journal on Scientific Computing, **20** (1999), pp. 1964–1977.

[22] G. Chavent and P. Lemonnier, *Identification de la Non-Linéarité D'Une Équation Parabolique Quasilinéare*, Applied Mathematics and Optimization, **1** (1974), pp. 121–162.

[23] T. F. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM Journal on Optimization, **6** (1996), pp. 418–445.

[24] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM Journal on Numerical Analysis, **25** (1988), pp. 433–460.

[25] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, SIAM, Philadelphia, 2000.

[26] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation, **19** (1965), pp. 297–301.

[27] J. W. Daniel, *The conjugate gradient method for linear and nonlinear operator equations*, SIAM Journal on Numerical Analysis, **4** (1967), pp. 10–26.

[28] A. R. Davies and R. S. Anderssen, *Optimization in the regularization of ill-posed problems*, Journal of the Australian Mathematical Society Series B, **28** (1986), pp. 114–133.

[29] P. J. Davis, *Circulant Matrices*, Wiley, New York, 1979.

[30] K. Deimling, *Nonlinear Functional Analysis*, Springer-Verlag, Berlin, 1985.

[31] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, **16** (1982), pp. 400–408.

[32] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[33] D. Dobson and F. Santosa, *An image enhancement technique for electrical impedance tomography*, Inverse Problems, **10** (1994), pp. 317–334.

[34] D. Dobson and F. Santosa, *Recovery of blocky images from noisy and blurred data*, SIAM Journal on Applied Mathematics, **56** (1996), pp. 1181–1198.

[35] H. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer, Dordrecht, 1996.

[36] R. Fletcher, *Practical Methods of Optimization*, 2nd Edition, Wiley, New York, 1987.

[37] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, Computer Journal, **7** (1964), pp. 149–154.

[38] J. N. Franklin, *Well-posed stochastic extensions of ill-posed linear problems*, Journal of Mathematical Analysis and Applications, **31** (1970), pp. 682–716.

[39] A. Friedlander and J. M. Martinez, *On the maximization of a concave quadratic function with box constraints*, SIAM Journal on Optimization, **4** (1994), pp. 177–192.

[40] A. Friedlander, J. M. Martinez, and S. A. Santos, *A new trust region algorithm for bound constrained minimization*, Applied Mathematics and Optimization, **30** (1994), pp. 235–266.

[41] D. Geman and C. Yang, *Nonlinear image recovery with half-quadratic regularization*, IEEE Transactions on Image Processing, **4** (1995), pp. 932–945.

[42] H. Gfrerer, *An a posteriori parameter choice for ordinary and iterated Tikhonov regularization of ill-posed problems leading to optimal convergence rates*, Mathematics of Computation, **49** (1987), pp. 507–522.

[43] D. Gilliam, J. Lund, and C. R. Vogel, *Quantifying information content for ill-posed problems*, Inverse Problems, **6** (1990), pp. 205–217.

[44] D. A. Girard, *The fast Monte-Carlo cross-validation and C_L procedures: Comments, new results, and application to image recovery problems*, Computational Statistics, **10** (1995), pp. 205–231.

[45] E. Giusti, *Minimal Surfaces and Functions of Bounded Variation*, Birkhäuser Boston, Boston, 1984.

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

[47] G. H. Golub and U. von Matt, *Generalized cross-validation for large-scale problems*, Journal of Computational and Graphical Statistics, **6** (1997), pp. 1–34.

[48] C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pitman, Boston, 1984.

[49] C. W. Groetsch, *Inverse Problems in the Mathematical Sciences*, Vieweg, Braunschweig, 1993.

[50] M. Hanke, *Conjugate Gradient Type Methods for Ill-Posed Problems*, Longman Scientific & Technical, Essex, UK, 1995.

[51] M. Hanke, *Limitations of the L-curve method in ill-posed problems*, BIT, **36** (1996), pp. 287–301.

[52] M. Hanke and P. C. Hansen, *Regularization methods for large-scale problems*, Surveys on Mathematics for Industry, **3** (1993), pp. 253–315.

[53] M. Hanke and J. G. Nagy, *Toeplitz approximate inverse preconditioner for banded Toeplitz matrices*, Numerical Algorithms, **7** (1994), pp. 183–199.

[54] M. Hanke, J. Nagy, and C. R. Vogel, *Quasi-Newton approach to nonnegative image restoration*, Linear Algebra and Its Applications, **316** (2000), pp. 223–236.

[55] M. Hanke and C. R. Vogel, *Two-level preconditioners for regularized inverse problems* I: *Theory*, Numerische Mathematik, **83** (1999), pp. 385–402.

[56] P. C. Hansen, *Numerical tools for analysis and solution of Fredholm integral equations of the first kind*, Inverse Problems, **8** (1992), pp. 956–972.

[57] P. C. Hansen, *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numerical Algorithms, **6** (1994), pp. 1–35.

[58] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.

[59] P. C. Hansen, *Regularization tools version* 3.0 *for Matlab* 5.2, Numerical Algorithms, **20** (1999), pp. 195–196.

[60] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM Journal on Scientific Computing, **14** (1993), pp. 1487–1503.

[61] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Communications in Statistics, Simulation, and Computation, **19** (1990), pp. 433–450.

[62] K. Ito and K. Kunisch, *An active set strategy based on the augmented Lagrangian formulation for image reconstruction*, RAIRO, Mathematical Modeling and Numerical Analysis, **33** (1999), pp. 1–21.

[63] M. Jacobsen, *Two-Grid Iterative Methods for Ill-Posed Problems*, MSc. Thesis, Report IMM-EKS-2000-27, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark, 2000.

[64] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, New York, 1989.

[65] T. Kailath and A. H. Sayed, *Fast Reliable Algorithms for Matrices with Structure*, SIAM, Philadelphia, 1999.

[66] L. Kaufman, *Maximum likelihood, least squares, and penalized least squares for PET*, IEEE Transactions on Medical Imaging, **12** (1993), pp. 200–214.

[67] J. Kay, *Asymptotic comparison factors for smoothing parameter choices in regression problems*, Statistics and Probability Letters, **15** (1992), pp. 329–335.

[68] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, Philadelphia, 1999.

[69] M. E. Kilmer and D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM Journal on Matrix Analysis and Applications, **22** (2001), pp. 1204–1221.

[70] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems*, Springer-Verlag, New York, 1996.

[71] E. Kreyszig, *Introduction to Functional Analysis with Applications*, Wiley, New York, 1978.

[72] L. Landweber, *An iteration formula for Fredholm integral equations of the first kind*, American Journal of Mathematics, **73** (1951), pp. 615–624.

[73] A. S. Leonov, *Numerical piecewise-uniform regularization for two-dimensional ill-posed problems*, Inverse Problems, **15** (1999), pp. 1165–1176.

[74] Y. Li and F. Santasa, *A computational algorithm for minimizing total variation in image restoration*, IEEE Transactions on Image Processing, **5** (1996), pp. 987–995.

[75] C. C. Lin and L. A. Segel, *Mathematics Applied to Deterministic Problems in the Natural Sciences*, SIAM, Philadelphia, 1988.

[76] C.-J. Lin and J. J. Moré, *Newton's method for large bound-constrained optimization problems*, SIAM Journal on Optimization, **9** (1999), pp. 1100–1127.

[77] J. Llacer and E. Veklerov, *Feasible images and practical stopping rules for iterative algorithms in emission tomography*, IEEE Transactions on Medical Imaging, **8** (1989), pp. 186–193.

[78] B. Lucy, *An iterative method for the rectification of observed distributions*, Astronomical Journal, **79** (1974), pp. 745–754.

[79] D. G. Luenberger, *Optimization by Vector Space Methods*, Wiley, New York, 1969.

[80] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[81] M. A. Lukas, *Asymptotic behavior of the minimum bound method for choosing the regularization parameter*, Inverse Problems, **14** (1998), pp. 149–159.

[82] M. A. Lukas, *Comparisons of parameter choice methods for regularization with discrete noisy data*, Inverse Problems, **14** (1998), pp. 161–184.

[83] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, Wiley, New York, 1997.

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

[85] J. J. Moré and G. Toraldo, *On the solution of large quadratic programming problems with bound constraints*, SIAM Journal on Optimization, **1** (1991), pp. 93–113.

[86] V. A. Morozov, *On the solution of functional equations by the method of regularization*, Soviet Mathematics Doklady, **7** (1966), pp. 414–417.

[87] V. A. Morozov, *Regularization Methods for Ill-Posed Problems*, CRC Press, Boca Raton, FL, 1993.

[88] J. Nagy and Z. Strakos, *Enforcing nonnegativity in image reconstruction algorithms*, in Proceedings of the SPIE International Conference on Mathematical Modeling, Estimation, and Imaging **4121**, David C. Wilson, et. al., eds., SPIE, Bellingham, WA, 2000, pp. 182–190.

[89] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, 1996.

[90] F. Natterer, *The Mathematics of Computerized Tomography*, Wiley, New York, 1986.

[91] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

[92] F. O'Sullivan and G. Wahba, *A cross validated Bayesian retrieval algorithm for nonlinear remote sensing*, Journal of Computational Physics, **59** (1985), pp. 441–455.

[93] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, Journal of the Association for Computing Machinery, **9** (1962), pp. 84–97.

[94] E. Polak, *Optimization. Algorithms and Consistent Approximations*, Springer-Verlag, New York, 1997.

[95] T. Raus, *On the discrepancy principle for the solution of ill-posed problems*, Acta et Commentationes Universitatis Tartuensis de Mathematica, **672** (1984), pp. 16–26 (in Russian).

[96] W. H. Richardson, *Bayesian-based iterative methods for image restoration*, Journal of the Optical Society of America, **62** (1972), pp. 55–59.

[97] A. Rieder, *A wavelet multilevel method for ill-posed problems stabilized by Tikhonov regularization*, Numerische Mathematik, **75** (1997), pp. 501–522.

[98] K. Riley and C. R. Vogel, *Preconditioners for linear systems arising in image reconstruction*, in Advanced Signal Processing Algorithms, Architectures, and Implementations VIII, Proceedings of SPIE **3461–37**, 1998, pp. 372–380.

[99] M. C. Roggemann and B. Welsh, *Imaging Through Turbulence*, CRC Press, Boca Raton, FL, 1996.

[100] R. L. Royden, *Real Analysis,* 2nd Edition, MacMillan, New York, 1968.

[101] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, **60** (1992), pp. 259–268.

[102] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.

[103] A. Spence, *Error bounds and estimates for eigenvalues of integral equations*, Numerische Mathematik, **29** (1978), pp. 133–147.

[104] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM Journal on Numerical Analysis, **20** (1983), pp. 626–637.

[105] G. Strang, *A proposal for Toeplitz matrix calculations*, Studies in Applied Mathematics, **74** (1986), pp. 171–176.

[106] A. N. Tikhonov, *Regularization of incorrectly posed problems*, Soviet Mathematics Doklady, **4** (1963), pp. 1624–1627.

[107] A. N. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*, Wiley, New York, 1977.

[108] A. N. Tikhonov, A. S. Leonov, and A. G. Yagola, *Nonlinear Ill-Posed Problems, Volumes* I *and* II, Chapman and Hall, London, 1998.

[109] A. M. Thompson, J. C. Brown, J. W. Kay, and D. M. Titterington, *A comparison of methods of choosing the smoothing parameter in image restoration by regularization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **13** (1991), pp. 326–339.

[110] C. F. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.

[111] Y. Vardi and D. Lee, *From image deblurring to optimal investments: Maximum likelihood solutions for positive linear inverse problems*, Journal of the Royal Statistical Society B, **55** (1993), pp. 569–612.

[112] E. Veklerov and J. Llacer, *Stopping rule for the MLE algorithm based on statistical hypothesis testing*, IEEE Transactions on Medical Imaging, **6** (1987), pp. 313–319.

[113] C. R. Vogel, *Optimal choice of a truncation level for the truncated SVD solution of linear first kind integral equations when the data are noisy*, SIAM Journal on Numerical Analysis, **23** (1986), pp. 109–117.

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

[115] C. R. Vogel, *Sparse matrix equations arising in distributed parameter identification*, SIAM Journal on Matrix Analysis, **20** (1999), pp. 1027–1037.

[116] C. R. Vogel, *A limited memory BFGS method for an inverse problem in atmospheric imaging*, in Methods and Applications of Inversion, P.C. Hansen, B.H. Jacobsen, and K. Mosegaard, eds., Lecture Notes in Earth Sciences **92**, 2000, Springer-Verlag, New York, pp. 292–304.

[117] C. R. Vogel, *Negative results for multilevel preconditioners in image deblurring*, in Scale-Space Theories in Computer Vision, M. Nielsen, P. Johansen, O.F. Olsen, and J. Weickert, eds., Springer-Verlag, New York, 1999.

[118] C. R. Vogel and M. Hanke, *Two-Level Preconditioners for Regularized Inverse Problems* II: *Implementation and Numerical Results*, technical report, Department of Mathematical Sciences, Montana State University, Bozeman, MT, 1999.

[119] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM Journal on Scientific Computing, **17** (1996), pp. 227–238.

[120] C. R. Vogel and M. E. Oman, *A fast, robust algorithm for total variation based reconstruction of noisy, blurred images*, IEEE Transactions on Image Processing, **7** (1998), pp. 813–824.

[121] C. R. Vogel and J. G. Wade, *Analysis of costate discretizations in parameter identification for linear evolution equations*, SIAM Journal on Control and Optimization, **33** (1995), pp. 227–254.

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

[123] G. Wahba, *Spline Models for Observational Data*, SIAM, Philadelphia, 1990.

[124] J. Weickert, *Anisotropic Diffusion in Image Processing*, Teubner, Stuttgart, 1998.

[125] E. Zeidler, *Nonlinear Functional Analysis and Its Applications.* I. *Fixed-Point Theorems*, Springer-Verlag, New York, 1986.

[126] E. Zeidler, *Nonlinear Functional Analysis and Its Applications.* II. *Variational Methods and Optimization*, Springer-Verlag, New York, 1985.

[127] E. Zeidler, *Applied Functional Analysis: Applications to Mathematical Physics*, Springer-Verlag, New York, 1995.

[128] E. Zeidler, *Applied Functional Analysis: Main Principles and their Applications*, Springer-Verlag, New York, 1995.