Discrete Inverse Problems

Fundamentals of Algorithms

Editor-in-Chief: Nicholas J. Higham, University of Manchester

The SIAM series on Fundamentals of Algorithms is a collection of short user-oriented books on state-of-the-art numerical methods. Written by experts, the books provide readers with sufficient knowledge to choose an appropriate method for an application and to understand the method's strengths and limitations. The books cover a range of topics drawn from numerical analysis and scientific computing. The intended audiences are researchers and practitioners using the methods and upper level undergraduates in mathematics, engineering, and computational science.

Books in this series not only provide the mathematical background for a method or class of methods used in solving a specific problem but also explain how the method can be developed into an algorithm and translated into software. The books describe the range of applicability of a method and give guidance on troubleshooting solvers and interpreting results. The theory is presented at a level accessible to the practitioner. MATLAB® software is the preferred language for codes presented since it can be used across a wide variety of platforms and is an excellent environment for prototyping, testing, and problem solving.

The series is intended to provide guides to numerical algorithms that are readily accessible, contain practical advice not easily found elsewhere, and include understandable codes that implement the algorithms.

Series Volumes

Hansen, P. C., *Discrete Inverse Problems: Insight and Algorithms*
Modersitzki, J., *FAIR: Flexible Algorithms for Image Registration*
Chan, R. H.-F. and Jin, X.-Q., *An Introduction to Iterative Toeplitz Solvers*
Eldén, L., *Matrix Methods in Data Mining and Pattern Recognition*
Hansen, P. C., Nagy, J. G., and O'Leary, D. P., *Deblurring Images: Matrices, Spectra, and Filtering*
Davis, T. A., *Direct Methods for Sparse Linear Systems*
Kelley, C. T., *Solving Nonlinear Equations with Newton's Method*

Per Christian Hansen
Technical University of Denmark
Lyngby, Denmark

Discrete Inverse Problems

Insight and Algorithms

siam®
Society for Industrial and Applied Mathematics
Philadelphia

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data:

Hansen, Per Christian.
Discrete inverse problems : insight and algorithms / Per Christian Hansen.
p. cm. -- (Fundamentals of algorithms series)
Includes bibliographical references and index.
ISBN 978-0-898716-96-2
1. Inverse problems (Differential equations) I. Title.
QA371.H365 2010
515'.357--dc22

2009047057

Contents

Preface

Inverse problems are mathematical problems that arise when our goal is to recover "interior" or "hidden" information from "outside"—or otherwise available—noisy data. For example, an inverse problem arises when we reconstruct a two-dimensional (2D) or three-dimensional (3D) medical image from tomography data, or when we reconstruct a sharper image from a blurred one. When we solve an inverse problem, we compute the source that gives rise to some observed data, using a mathematical model for the relation between the source and the data.

Inverse problems arise in many technical and scientific areas, such as medical and geophysical imaging, electromagnetic scattering, and nondestructive testing. Image deblurring arises, e.g., in astronomy or in biometric applications that involve fingerprint or iris recognition. The underlying mathematics is rich and well developed, and there are many books devoted to the subject of inverse (and ill-posed) problems.

So why yet another book? My experience from teaching this subject to engineering graduate students is that there is a need for a textbook that covers the basic subjects and also focuses on the computational aspects. Moreover, I believe that practical computational experience is important for understanding applied mathematics, and therefore the textbook should include a number of tutorial exercises to give the reader hands-on experience with the difficulties and challenges associated with the treatment of inverse problems.

The title of the book reflects this point of view: our *insight* about inverse problems must go hand-in-hand with our *algorithms* for solving these problems. Solving an inverse problem is rarely a matter of just picking an algorithm from a textbook, a research paper, or a software package. My experience is that each new inverse problem has its own features and peculiarities, which must be understood before one can decide on an algorithm (or, sometimes, develop a new one).

The present book is intended as a quite gentle introduction to a field characterized by advanced mathematics and sophisticated numerical methods. The book does not pretend to tell the whole story, to give all the details, or to survey all the important methods and techniques. The aim is to provide the reader with enough background in mathematics and numerical methods to understand the basic difficulties associated with linear inverse problems, to analyze the influence of measurement and approximation errors, and to design practical algorithms for computing regularized/stabilized solutions to these problems. Provided with this insight, the reader will be able to start reading the more advanced literature on the subject; indeed, anyone who wants to work in the area of linear inverse problems is advised to also consult some of the many

well-written books on the subject, such as [3], [8], [14], [23], [24], [32], [64], [74], [76].

The focus of the book is on linear inverse problems in the form of Fredholm integral equations of the first kind. The presentation starts with a summary of the most important properties of linear inverse problems in the continuous setting. Then we briefly discuss discretization methods and describe how many of the properties of the integral equation directly carry over to the discretized system—in the form of a linear (perhaps overdetermined) system of equations. The next chapter is devoted to simple regularization methods for computing regularized solutions in the form of filtered spectral expansions; this is an important class of methods which clearly illustrates the basic ideas of regularization. Since no regularization algorithm is complete without a method for choosing the regularization parameter, we also include a discussion of some state-of-the-art parameter choice methods. We conclude with a chapter on iterative methods for large-scale problems, a chapter with a some real-world problems, and a chapter on a more general class of regularization methods. Sections and exercises marked with a * denote more advanced material that can be skipped in a basic course.

At the end of each section we give a number of exercises, most of them involving numerical experiments with the MATLAB package *Regularization Tools* [31], [33], which further illustrate the concepts and methods discussed in the corresponding section. The package is available from Netlib at http://www.netlib.org/numeralgo and from the MATLAB Central File Exchange at http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=52. It must be emphasized that the package is mainly intended for teaching and experimenting with small-scale inverse problems, and therefore the package is not designed to be efficient for large problems.

Acknowledgments. This tutorial grew out of a series of lectures given at the Fifth Winter School in Computational Mathematics in Geilo, Norway, in February of 2005. The atmosphere there was very positive, and I enjoyed the chance to teach one of my favorite subjects to a dedicated audience. The presentation is based on many years of experience from numerous collaborations, too many to mention here, and I thank everyone I worked with for inspiration and motivation. In particular, I thank Zdeněk Strakoš for insightful discussions about regularizing iterations, Maurizio Fedi for sharing his insight in potential field inversion, Søren Holdt Jensen for introducing me to all kinds of noise, Jim Nagy for showing me that structure is everything in image deblurring, and Bill Lionheart for all kinds of thoughts on inverse problems. I also thank Ann Manning Allen, Elizabeth Greenspan, Nancy Griscom, and Sara Murphy from SIAM for their competent handling of this book.

Per Christian Hansen
Lyngby, 2009

List of Symbols

Symbol	Quantity	Dimension
A	coefficient matrix	$m \times n$
A_k	TSVD matrix	$m \times n$
$A_k^\dagger$	pseudoinverse of A_k	$n \times m$
b	right-hand side	m
B_k	lower bidiagonal matrix	$(k+1) \times k$
c_λ	curvature of Tikhonov L-curve in lin-lin scale	scalar
$\hat{c}_\lambda$	ditto in log-log scale	scalar
e	noise component in right-hand side	m
E	error in quadrature or expansion method	scalar
f	solution function (integral equation)	
g	right-hand side function (integral equation)	
G	GCV function	
h	grid spacing	scalar
I	identity matrix	
k	truncation parameter (TSVD)	integer
k_η	transition index	integer
K	kernel function (integral equation)	
$\mathcal{K}_k$	Krylov subspace of dimension k	
ℓ_i	eigenvalue of kernel	scalar
L	regularization matrix	$p \times n$
L_1	discrete 1. order derivative	$(n-1) \times n$
L_2	discrete 2. order derivative	$(n-2) \times n$
m, n	matrix dimensions, $m \geq n$	scalars
s, t	independent variables (integral equation)	
u_i	left singular function or vector	m
U	left singular matrix	$m \times n$
v_i	right singular function or vector	n
V	right singular matrix	$n \times n$
w_k	basis vectors of projection method; also CGLS and Lanczos vectors	n
W_k	matrix of basis vectors	$n \times k$

Symbol	Quantity	Dimension
x	"naive" solution	n
x_k, x_λ	TSVD and Tikhonov solutions	n
$x^{[k]}$	Landweber, Cimmino, ART iteration vector	n
$x^{(k)}$	solution computed via projection; also CGLS solution	n
$x_\lambda^{(k)}$	regularized solution via projection	n
$y^{(k)}$	solution to projected problem	k
α	relative decay of SVD coefficients	scalar
α_k, β_k	from bidiagonalization algorithm	scalars
$\bar{\alpha}_k$, $\bar{\beta}_k$	from CGLS algorithm	scalars
γ, γ_k, $\hat{\gamma}_k$, γ_λ	constants in perturbation bounds	scalars
Γ	Gamma function	
δ	upper bound on solution norm also delta function	scalar
ΔA	matrix perturbation	$m \times n$
Δb	right-hand side perturbation	m
ε	upper bound on residual norm	scalar
ε_k, ε_λ	TSVD and Tikhonov regularization errors	scalars
ζ_j	expansion coefficient	scalar
η	standard deviation for noise	scalar
κ_k, κ_λ	TSVD and Tikhonov condition numbers	scalars
λ	regularization parameter (Tikhonov)	scalar
μ_i	singular value of kernel	scalar
μ	safety factor (in various methods)	scalar
ξ, $\hat{\xi}$	solution norm squared, log of ditto	scalars
ρ, $\hat{\rho}$	residual norm squared, log of ditto	scalars
σ_i	singular value of matrix	scalar
Σ	diagonal matrix with singular values	$n \times n$
τ	threshold in SSVD method	scalar
ϕ_i, ψ_i	basis functions	
φ_i	generic filter factor	scalar
$\varphi_i^{[k]}$	filter factor for an iterative method	scalar
$\varphi_i^{[\lambda]}$	Tikhonov filter factor	scalar
$\Phi^{[\cdot]}$	diagonal matrix of filter factors	$n \times n$
Ψ	matrix used to generate colored noise	$m \times m$
χ_i	"top hat" $\sqcap\!\!\sqcup$ function	
ω_j	quadrature weight	scalar
$\langle \cdot, \cdot \rangle$	inner product	
$\| \cdot \|_2$, $\| \cdot \|_F$	2-norm and Frobenius norm	
$\mathrm{cond}(\cdot)$	condition number	
$\mathrm{Cov}(\cdot)$	covariance matrix	
$\mathcal{E}(\cdot)$	expected value	
$\mathrm{span}\{\cdots\}$	subspace spanned by vectors	
$\widetilde{\square}$	perturbed version of $\square$	

Chapter 1

Introduction and Motivation

If you have acquired this book, perhaps you do not need a motivation for studying the numerical treatment of inverse problems. Still, it is preferable to start with a few examples of the use of linear inverse problems. One example is that of computing the magnetization inside the volcano Mt. Vesuvius (near Naples in Italy) from measurements of the magnetic field above the volcano—a safe way to monitor the internal activities. Figure 1.1 below shows a computer simulation of this situation; the left figure shows the measured data on the surface of the volcano, and the right figure shows a reconstruction of the internal magnetization. Another example is the computation of a sharper image from a blurred one, using a mathematical model of the point spread function that describes the blurring process; see Figure 1.2 below.

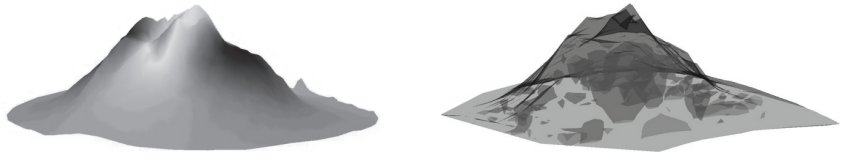

Figure 1.1. *Left: Simulated measurements of the magnetic field on the surface of Mt. Vesuvius. Right: Reconstruction of the magnetization inside the volcano.*

Figure 1.2. *Reconstruction of a sharper image from a blurred one.*

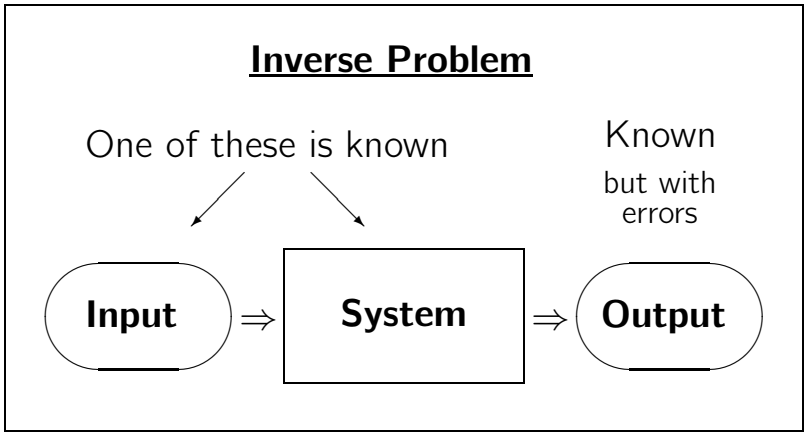

Figure 1.3. *The forward problem is to compute the output, given a system and the input to this system. The inverse problem is to compute either the input or the system, given the other two quantities. Note that in most situations we have imprecise (noisy) measurements of the output.*

Both are examples of a wide class of mathematical problems, referred to as *inverse problems*. These problems generally arise when we wish to compute information about internal or otherwise hidden data from outside (or otherwise accessible) measurements. See Figure 1.3 for a schematic illustration of an inverse problem.

Inverse problems, in turn, belong to the class of *ill-posed problems*. The term was coined in the early 20th century by Hadamard who worked on problems in mathematical physics, and he believed that ill-posed problems do not model real-world problems (he was wrong). Hadamard's definition says that a linear problem is well-posed if it satisfies the following three requirements:

- **Existence:** The problem must have a solution.
- **Uniqueness:** There must be only one solution to the problem.
- **Stability:** The solution must depend continuously on the data.

If the problem violates one or more of these requirements, it is said to be ill-posed.

The existence condition seems to be trivial—and yet we shall demonstrate that we can easily formulate problems that do not have a solution. Consider, for example, the overdetermined system

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2.2 \end{pmatrix}.$$

This problem does not have a solution; there is no x such that $x = 1$ and $2x = 2.2$. Violations of the existence criterion can often be fixed by a slight reformulation of the problem; for example, instead of the above system we can consider the associated

least squares problem

$$\min_x \left\| \begin{pmatrix} 1 \\ 2 \end{pmatrix} x - \begin{pmatrix} 1 \\ 2.2 \end{pmatrix} \right\|_2^2 = \min_x \Big((x-1)^2 + (2x-2.2)^2 \Big),$$

which has the unique solution $x = 1.08$.

The uniqueness condition can be more critical; but again it can often be fixed by a reformulation of the problem—typically by adding additional requirements to the solution. If the requirements are carefully chosen, the solution becomes unique. For example, the underdetermined problem

$$x_1 + x_2 = 1 \qquad \text{(the world's simplest ill-posed problem)}$$

has infinitely many solutions; if we also require that the 2-norm of x, given by $\|x\|_2 = (x_1^2 + x_2^2)^{1/2}$, is minimum, then there is a unique solution $x_1 = x_2 = 1/2$.

The stability condition is much harder to "deal with" because a violation implies that arbitrarily small perturbations of data can produce arbitrarily large perturbations of the solution. At least, this is true for infinite-dimensional problems; for finite-dimensional problems the perturbation is always finite, but this is quite irrelevant if the perturbation of the solution is, say, of the order 10^{12}.

Again the key is to reformulate the problem such that the solution to the new problem is less sensitive to the perturbations. We say that we *stabilize* or *regularize* the problem, such that the solution becomes more stable and regular. As an example, consider the least squares problem $\min_x \|Ax - b\|_2$ with coefficient matrix and right-hand side given by

$$A = \begin{pmatrix} 0.16 & 0.10 \\ 0.17 & 0.11 \\ 2.02 & 1.29 \end{pmatrix}, \qquad b = A \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0.01 \\ -0.03 \\ 0.02 \end{pmatrix} = \begin{pmatrix} 0.27 \\ 0.25 \\ 3.33 \end{pmatrix}.$$

Here, we can consider the vector $(0.01\,,\,-0.03\,,\,0.02)^T$ a perturbation of the exact right-hand side $(0.26\,,\,0.28\,,\,3.31)^T$. There is no vector x such that $Ax = b$, and the least squares solution is given by

$$x_{\mathrm{LS}} = \begin{pmatrix} 7.01 \\ -8.40 \end{pmatrix} \qquad \Rightarrow \qquad \|Ax_{\mathrm{LS}} - b\|_2 = 0.022\ .$$

Two other "solutions" with a small residual are

$$x' = \begin{pmatrix} 1.65 \\ 0 \end{pmatrix}, \qquad x'' = \begin{pmatrix} 0 \\ 2.58 \end{pmatrix} \qquad \Rightarrow$$

$$\|Ax' - b\|_2 = 0.031, \qquad \|Ax'' - b\|_2 = 0.036.$$

All three "solutions" x_{LS}, x', and x'' have small residuals, yet they are far from the exact solution $(\,1\,,\,1\,)^T$!

The reason for this behavior is that the matrix A is ill conditioned. When this is the case, it is well known from matrix computations that small perturbations of the right-hand side b can lead to large perturbations of the solution. It is also well known

that a small residual does not imply that the perturbed solution is close to the exact solution.

Ill-conditioned problems are *effectively underdetermined*. For example, for the above problem we have

$$A \begin{pmatrix} -1.00 \\ 1.57 \end{pmatrix} = \begin{pmatrix} -0.0030 \\ 0.0027 \\ 0.0053 \end{pmatrix},$$

showing that the vector $(-1.00\,,\,1.57)^T$ is "almost" a null vector for A. Hence we can add a large amount of this vector to the solution vector without changing the residual very much; the system behaves *almost* like an underdetermined system.

It turns out that we can modify the above problem such that the new solution is more stable, i.e., less sensitive to perturbations. For example, we can enforce an upper bound δ on the norm of the solution; i.e., we solve the modified problem:

$$\min_x \|Ax - b\|_2 \qquad \text{subject to} \qquad \|x\|_2 \leq \delta.$$

The solution x_δ depends in a unique but nonlinear way on δ; for example,

$$x_{0.1} = \begin{pmatrix} 0.08 \\ 0.05 \end{pmatrix}, \quad x_1 = \begin{pmatrix} 0.84 \\ 0.54 \end{pmatrix}, \quad x_{1.37} = \begin{pmatrix} 1.16 \\ 0.74 \end{pmatrix}, \quad x_{10} = \begin{pmatrix} 6.51 \\ -7.60 \end{pmatrix}.$$

The solution $x_{1.37}$ (for $\delta = 1.37$) is quite close to the exact solution. *By supplying the correct additional information we can compute a good approximate solution.* The main difficulty is how to choose the parameter δ when we have little knowledge about the exact solution.

Whenever we solve an inverse problem on a computer, we always face difficulties similar to the above, because the associated computational problem is ill conditioned. The purpose of this book is:

1. To discuss the inherent instability of inverse problems, in the form of first-kind Fredholm integral equations.

2. To explain why ill-conditioned systems of equations always arise when we discretize and solve these inverse problems.

3. To explain the fundamental "mechanisms" of this ill conditioning and how they reflect properties of the underlying problem.

4. To explain how we can modify the computational problem in order to stabilize the solution and make it less sensitive to errors.

5. To show how this can be done efficiently on a computer, using state-of-the-art methods from numerical analysis.

Regularization methods are at the heart of all this, and in the rest of this book we will develop these methods with a keen eye on the fundamental interplay between *insight* and *algorithms*.

Chapter 2

Meet the Fredholm Integral Equation of the First Kind

This book deals with one important class of linear inverse problems, namely, those that take the form of Fredholm integral equations of the first kind. These problems arise in many applications in science and technology, where they are used to describe the relationship between the source—the "hidden data"—and the measured data. Some examples are

- medical imaging (CT scanning, electro-cardiography, etc.),
- geophysical prospecting (search for oil, land-mines, etc.),
- image deblurring (astronomy, crime scene investigations, etc.),
- deconvolution of a measurement instrument's response.

If you want to work with linear inverse problems arising from first-kind Fredholm integral equations, you must make this integral equation your friend. In particular, you must understand the "psyche" of this beast and how it can play tricks on you if you are not careful. This chapter thus sets the stage for the remainder of the book by briefly surveying some important theoretical aspects and tools associated with first-kind Fredholm integral equations.

Readers unfamiliar with inner products, norms, etc. in function spaces may ask: How do I avoid reading this chapter? The answer is: Do not avoid it completely; read the first two sections. Readers more familiar with this kind of material are encouraged to read the first four sections, which provide important background material for the rest of the book.

2.1 A Model Problem from Geophysics

It is convenient to start with a simple model problem to illustrate our theory and algorithms. We will use a simplified problem from gravity surveying. An unknown mass distribution with density $f(t)$ is located at depth d below the surface, from 0 to 1 on the t axis shown in Figure 2.1. We assume there is no mass outside this source,

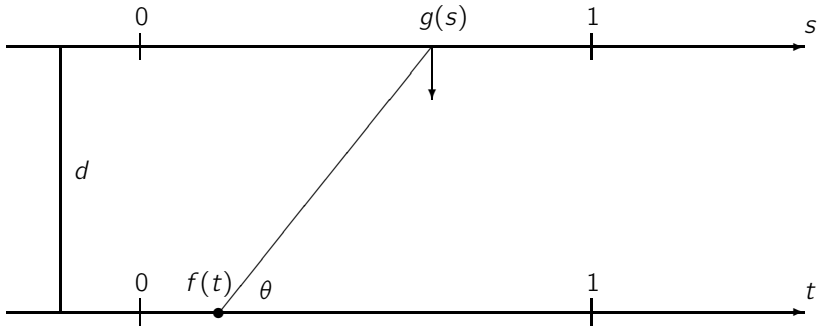

Figure 2.1. *The geometry of the gravity surveying model problem: $f(t)$ is the mass density at t, and $g(s)$ is the vertical component of the gravity field at s.*

which produces a gravity field everywhere. At the surface, along the s axis (see the figure) from 0 to 1, we measure the vertical component of the gravity field, which we refer to as $g(s)$.

The two functions f and g are (not surprisingly here) related via a Fredholm integral equation of the first kind. The gravity field from an infinitesimally small part of $f(t)$, of length dt, on the t axis is identical to the field from a point mass at t of strength $f(t)\,dt$. Hence, the magnitude of the gravity field along s is $f(t)\,dt\,/\,r^2$, where $r = \sqrt{d^2 + (s-t)^2}$ is the distance between the "source point" at t and the "field point" at s. The direction of the gravity field is from the "field point" to the "source point," and therefore the measured value of $g(s)$ is

$$dg = \frac{\sin\theta}{r^2}\, f(t)\, dt,$$

where θ is the angle shown in Figure 2.1. Using that $\sin\theta = d/r$, we obtain

$$\frac{\sin\theta}{r^2}\, f(t)\, dt = \frac{d}{(d^2 + (s-t)^2)^{3/2}}\, f(t)\, dt.$$

The total value of $g(s)$ for any $0 \le s \le 1$ consists of contributions from all mass along the t axis (from 0 to 1), and it is therefore given by the integral

$$g(s) = \int_0^1 \frac{d}{(d^2 + (s-t)^2)^{3/2}}\, f(t)\, dt.$$

This is the forward problem which describes how we can compute the measurable data g given the source f.

The associated inverse problem of gravity surveying is obtained by swapping the ingredients of the forward problem and writing it as

$$\int_0^1 K(s,t)\, f(t)\, dt = g(s), \qquad 0 \le s \le 1,$$

where the function K, which represents the model, is given by

$$K(s,t) = \frac{d}{(d^2 + (s-t)^2)^{3/2}}, \tag{2.1}$$

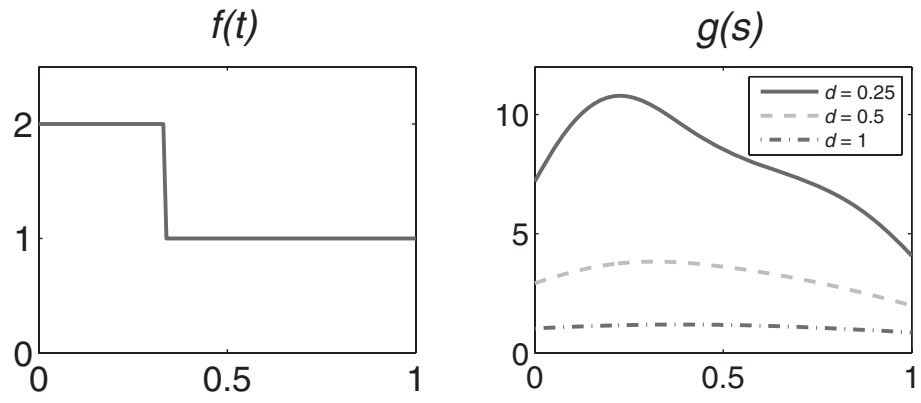

Figure 2.2. *The left figure shows the function f (the mass density distribution), and the right figure shows the measured signal g (the gravity field) for three different values of the depth d.*

and the right-hand side g is what we are able to measure. The function K is the vertical component of the gravity field, measured at s, from a unit point source located at t. From K and g we want to compute f, and this is the inverse problem.

Figure 2.2 shows an example of the computation of the measured signal $g(s)$, given the mass distribution f and three different values of the depth d. Obviously, the deeper the source, the weaker the signal. Moreover, we see that the observed signal g (the data) is much smoother than the source f, and in fact the discontinuity in f is not visible in g.

2.2 Properties of the Integral Equation

The Fredholm integral equation of the first kind takes the generic form

$$\boxed{\int_0^1 K(s,t)\, f(t)\, dt = g(s), \qquad 0 \le s \le 1.} \tag{2.2}$$

Here, both the *kernel* K and the *right-hand side* g are known functions, while f is the unknown function. This equation establishes a linear relationship between the two functions f and g, and the kernel K describes the precise relationship between the two quantities. Thus, the function K describes the underlying model. In Chapter 7 we will encounter formulations of (2.2) in multiple dimensions, but our discussion until then will focus on the one-dimensional case.

If f and K are known, then we can compute g by evaluating the integral; this is called the forward computation. The *inverse problem* consists of computing f given the right-hand side and the kernel. Two examples of these ingredients are listed in Table 2.1.

An important special case of (2.2) is when the kernel is a function of the difference between s and t, i.e., $K(s,t) = h(s-t)$, where h is some function. This version

Table 2.1. *The ingredients and mechanisms of two inverse problems.*

Problem	Image deblurring	Gravity surveying
Source f	Sharp image	Mass distribution
Data g	Blurred image	Gravity field component
Kernel K	Point spread function	Field from point mass
Forward problem	Compute blurred image	Compute gravity field
Inverse problem	Reconstruct sharp image	Reconstruct mass distrib.

of the integral equation is called a *deconvolution* problem, and it takes the form

$$\int_0^1 h(s-t)\,f(t)\,dt = g(s), \qquad 0 \le s \le 1$$

(and similarly in more dimensions). It follows from (2.1) that the gravity survey problem from the previous section is actually a convolution problem. Some numerical regularization methods for deconvolution problems are described in [35]; see also Sections 7.1 and 7.5 on barcode reading and image deblurring.

The smoothing that we observed in the above example, when going from the source f to the data g, is a universal phenomenon for integral equations. In the mapping from f to g, higher frequency components in f are damped compared to components with lower frequency. Thus, the integration with K in (2.2) has a smoothing effect on the function f, such that g will appear smoother than f.

The *Riemann–Lebesgue lemma* is a precise mathematical statement of the above. If we define the function f_p by

$$f_p(t) = \sin(2\pi\, p\, t), \qquad p = 1, 2, \ldots,$$

then for "arbitrary" kernels K we have

$$g_p(s) = \int_0^1 K(s,t)\,f_p(t)\,dt \to 0 \qquad \text{for} \qquad p \to \infty.$$

That is, as the frequency of f increases—as measured by the integer p—the amplitude of g_p decreases; Figure 2.3 illustrates this. See, e.g., Theorem 12.5C in [19] for a precise formulation of the Riemann–Lebesgue lemma.

In other words, higher frequencies are damped in the mapping of f to g, and therefore g will be smoother than f. The inverse problem, i.e., that of computing f from g, is therefore a process that amplifies high frequencies, and the higher the frequency, the more the amplification. Figure 2.4 illustrates this: the functions δf_p and δg_p are related via the same integral equation as before, but now all four δg_p functions are scaled such that $\|\delta g_p\|_2 = 0.01$. The amplitude of δf_p increases as the frequency p increases.

Clearly, even a small random perturbation of g can lead to a very large perturbation of f if the perturbation has a high-frequency component. Think of the function δg_p in Figure 2.4 as a perturbation of g and the function δf_p as the corresponding perturbation of f. As a matter of fact, no matter how small the perturbation of g,

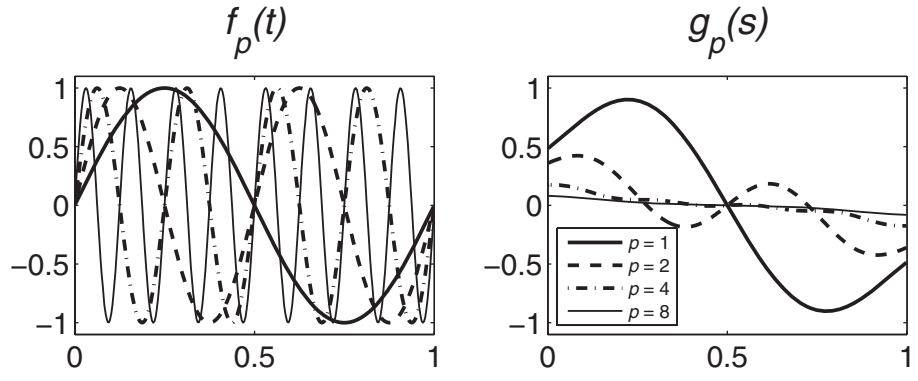

Figure 2.3. *Illustration of the Riemann–Lebesgue lemma with the function* $f_p(t) = \sin(2\pi\, p\, t)$. *Clearly, the amplitude of* g_p *decreases as the frequency* p *increases.*

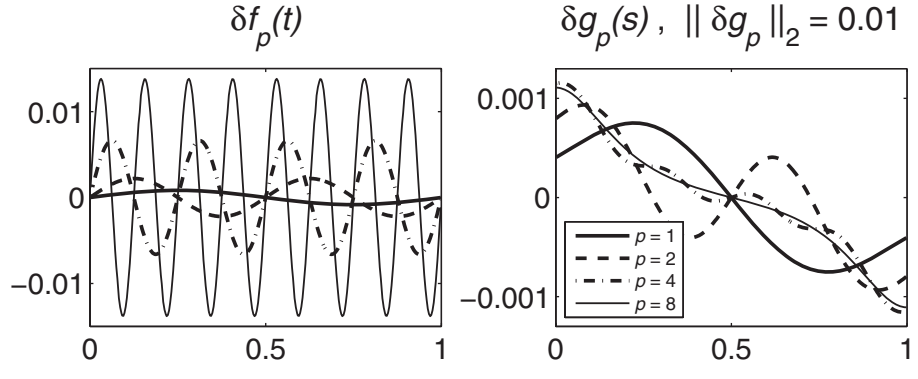

Figure 2.4. *The consequence of the Riemann–Lebesgue lemma is that high frequencies are amplified in the inversion. Here the functions* δg_p *are normalized such that* $\|\delta g_p\|_2 = 0.01$. *Clearly, the amplitude of* δf_p *increases as the frequency* p *increases.*

the corresponding perturbation of f can be arbitrarily large (just set the frequency p of the perturbation high enough). This illustrates the fundamental problem of solving inverse problems, namely, that the solution may not depend continuously on the data.

Why bother about these issues associated with ill-posed problems? Fredholm integral equations of the first kind are used to model a variety of real applications. We can only hope to compute useful solutions to these problems if we fully understand their *inherent* difficulties. Moreover, we must understand how these difficulties carry over to the discretized problems involved in a computer solution, such that we can deal with them (mathematically and numerically) in a satisfactory way. This is precisely what the rest of this book is about.

2.3 The Singular Value Expansion and the Picard Condition

The singular value expansion (SVE) is a mathematical tool that gives us a "handle" on the discussion of the smoothing effect and the existence of solutions to first-kind Fredholm integral equations. First we need to introduce a bit of notation. Given two functions ϕ and ψ defined on the interval from 0 to 1, their *inner product* is defined as

$$\langle \phi, \psi \rangle \equiv \int_0^1 \phi(t)\, \psi(t)\, dt. \tag{2.3}$$

Moreover, the 2-norm of the function ϕ is defined by

$$\|\phi\|_2 \equiv \langle \phi, \phi \rangle^{1/2} = \left(\int_0^1 \phi(t)^2\, dt \right)^{1/2}. \tag{2.4}$$

The kernel K in our generic integral equation (2.2) is square integrable if the integral $\int_0^1 \int_0^1 K(s,t)^2\, ds\, dt$ is finite. For any square integrable kernel K the *singular value expansion* (SVE) takes the form

$$\boxed{K(s,t) = \sum_{i=1}^{\infty} \mu_i\, u_i(s)\, v_i(t).} \tag{2.5}$$

The functions u_i and v_i are called the left and right singular functions. All the u_i-functions are orthonormal with respect to the usual inner product (2.3), and the same is true for all the v_i-functions, i.e.,

$$\langle u_i, u_j \rangle = \langle v_i, v_j \rangle = \delta_{ij}, \qquad i = 1, 2, \ldots.$$

The quantities μ_i are called the singular values, and they form a nonincreasing sequence:

$$\mu_1 \geq \mu_2 \geq \mu_3 \geq \cdots \geq 0.$$

If there is only a finite number of nonzero singular values, then the kernel is called degenerate. The singular values and functions satisfy a number of relations, and the most important is the "fundamental relation"

$$\int_0^1 K(s,t)\, v_i(t)\, dt = \mu_i\, u_i(s), \qquad i = 1, 2, \ldots. \tag{2.6}$$

We note that it is rare that we can determine the SVE analytically; later we shall demonstrate how we can compute it numerically. For a more rigorous discussion of the SVE, see, e.g., Section 15.4 in [50].

2.3.1 The Role of the SVE

The singular values μ_i always decay to zero, and it can be shown that the "smoother" the kernel K, the faster the decay. Specifically, if the derivatives of K of order

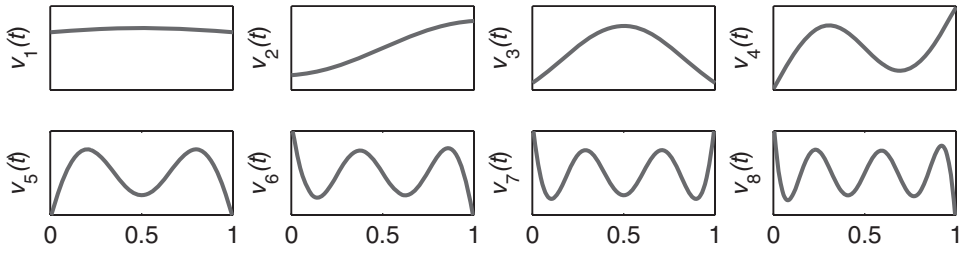

Figure 2.5. *The number of oscillations, or zero-crossings, in u_i and v_i increases with i (as the corresponding singular values decay). Here we used the gravity surveying model problem.*

$0, \ldots, q$ exist and are continuous, then the singular values μ_i decay approximately as $\mathcal{O}(i^{-q-1/2})$. If K is infinitely many times differentiable, then the singular values decay even faster: the μ_i decay exponentially, i.e., as $\mathcal{O}(\varrho^i)$ for some positive ϱ strictly smaller than one.

The singular functions resemble spectral bases in the sense that the smaller the μ_i, the more oscillations (or zero-crossings) in the corresponding singular functions u_i and v_i. Figure 2.5 illustrates this.

The left and right singular functions form bases of the function space $L_2([0,1])$ of square integrable functions in the interval $[0,1]$. Hence, we can expand both f and g in terms of these functions:

$$f(t) = \sum_{i=1}^{\infty} \langle v_i, f \rangle \, v_i(t) \qquad \text{and} \qquad g(s) = \sum_{i=1}^{\infty} \langle u_i, g \rangle \, u_i(s).$$

If we insert the expansion for f into the integral equation and make use of the fundamental relation, then we see that g can also be written as

$$g(s) = \sum_{i=1}^{\infty} \mu_i \, \langle v_i, f \rangle \, u_i(s).$$

Since v_i is mapped to $\mu_i \, u_i(s)$, this explains why the higher frequencies are damped more than the lower frequencies in the forward problem, i.e., why the integration with K has a smoothing effect.

When we insert the two expansions for f and g into the integral equation and make use of the fundamental relation again, then it is easy to obtain the relation

$$\sum_{i=1}^{\infty} \mu_i \, \langle v_i, f \rangle \, u_i(s) = \sum_{i=1}^{\infty} \langle u_i, g \rangle \, u_i(s). \tag{2.7}$$

The first observation to make from this relation is that if there are zero singular values μ_i, then we can only have a solution $f(t)$ that satisfies (2.2) if the corresponding components $\langle u_i, g \rangle \, u_i(s)$ are zero.

For convenience let us now assume that all $\mu_i \neq 0$. It turns out that there is still a condition on the right-hand side $g(s)$. By equating each of the terms in the expansion

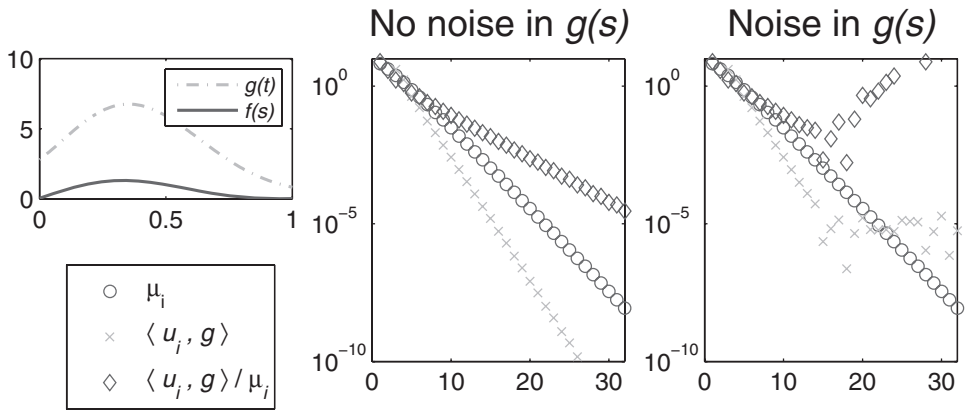

Figure 2.6. *Illustration of the Picard condition. The left figure shows the functions f and g used in the example. The middle figure shows the singular values μ_i and the SVE expansion coefficients $\langle u_i, g\rangle$ and $\langle u_i, g\rangle / \mu_i$ (for g and f, respectively) for a noise-free problem. The right figure shows how these coefficients change when we add noise to the right-hand side.*

(2.7), we see that the expansion coefficients for the solution are $\langle v_i, f\rangle = \langle u_i, g\rangle / \mu_i$ for $i = 1, 2, \ldots$, and this leads to the following expression for the solution:

$$f(t) = \sum_{i=1}^{\infty} \frac{\langle u_i, g\rangle}{\mu_i} v_i(t). \tag{2.8}$$

While this relation is not suited for numerical computations, it sheds important light on the existence of a solution to the integral equation. More precisely, we see that for a square integrable solution f to exist, the 2-norm $\|f\|_2$ must be bounded, which leads to the following condition:

The Picard condition. The solution is square integrable if

$$\|f\|_2^2 = \int_0^1 f(t)^2\, dt = \sum_{i=1}^{\infty} \langle v_i, f\rangle^2 = \sum_{i=1}^{\infty} \left(\frac{\langle u_i, g\rangle}{\mu_i}\right)^2 < \infty. \tag{2.9}$$

In words, the Picard condition says that the right-hand side coefficients $\langle u_i, g\rangle$ must decay to zero somewhat faster than the singular values μ_i. We recognize this as a condition on the smoothness of the right-hand side.

The trouble with first-kind Fredholm integral equations is that, even if the exact data satisfies the Picard condition, the measured and noisy data g usually violates the condition. Figure 2.6 illustrates this. The middle figure shows the singular values μ_i and the expansion coefficients $\langle u_i, g\rangle$ and $\langle u_i, g\rangle / \mu_i$ for the ideal situation with no noise, and here the Picard condition is clearly satisfied. The right figure shows the behavior of these quantities when we add noise to the right-hand side; now the right-hand side coefficients $\langle u_i, g\rangle$ level off at the noise level, and the Picard condition

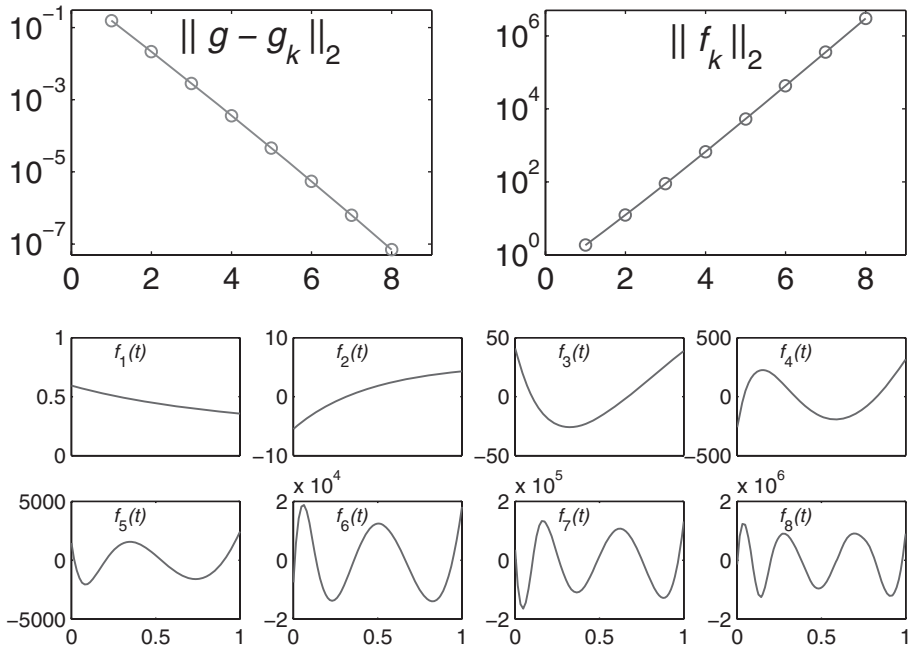

Figure 2.7. *Ursell's problem* (2.10). *Top left: The norm of the error* $g - g_k$ *in the approximation* (2.11) *for the right-hand side. Top right: The norm of the "approximate solution"* f_k (2.12). *Bottom: "Approximate solutions"* f_k *for* $k = 1, \ldots, 8$.

is violated. As a consequence, the solution coefficients $\langle u_i, g \rangle / \mu_i$ increase, and the norm of the "solution" becomes unbounded.

The violation of the Picard condition is the simple explanation of the instability of linear inverse problems in the form of first-kind Fredholm integral equations. At the same time, the insight we gain from a study of the quantities associated with the SVE gives a hint on how to deal with noisy problems that violate the Picard condition; simply put, we want to filter the noisy SVE coefficients.

2.3.2 Nonexistence of a Solution

Ursell [70] has provided an example of a first-kind Fredholm integral equation that does not have a square integrable solution; i.e., there is no solution whose 2-norm is finite. His example takes the form

$$\int_0^1 \frac{1}{s+t+1} f(t)\, dt = 1, \qquad 0 \le s \le 1. \tag{2.10}$$

Hence, the kernel and the right-hand side are given by $K(s,t) = (s+t+1)^{-1}$ and $g(s) = 1$.

Let us expand the right-hand side in terms of the left singular functions. We consider the approximation

$$g_k(s) = \sum_{i=1}^{k} \langle u_i, g \rangle \, u_i(s), \tag{2.11}$$

and the top left plot in Figure 2.7 shows that the error in g_k decreases as k increases; we have

$$\|g - g_k\|_2 \to 0 \qquad \text{for} \qquad k \to \infty.$$

Next we consider the integral equation $\int_0^1 (s+t+1)^{-1} f_k(t)\, dt = g_k(s)$, whose solution f_k is given by the expansion

$$f_k(t) = \sum_{i=1}^{k} \frac{\langle u_i, g \rangle}{\mu_i} \, v_i(t). \tag{2.12}$$

Clearly $\|f_k\|_2$ is finite for all k; but the top right plot in Figure 2.7 shows that the norm of f_k increases with k:

$$\|f_k\|_2 \to \infty \qquad \text{for} \qquad k \to \infty.$$

The bottom plots in Figure 2.7 corroborate this: clearly, the amplitude of the function f_k increases with k, and these functions do not seem to converge to a square integrable solution.

2.3.3 Nitty-Gritty Details of the SVE*

We conclude this section with some nitty-gritty details that some readers may wish to skip. If the kernel K is continuous, then for all equations that involve infinite sums of K's singular functions, such as (2.5), (2.6), and (2.8), the equality sign means that the sum converges uniformly to the left-hand side. If K is discontinuous, then we have convergence in the mean square. For example, for (2.8), uniform convergence means that for every ε there exists an integer N such that for all $k > N$ we have

$$\left| f(t) - \textstyle\sum_{i=1}^{k} \frac{\langle u_i, g \rangle}{\mu_i} \, v_i(t) \right| < \varepsilon \qquad \forall\, t \in [0, 1].$$

For convergence in the mean square, the inequality takes the form

$$\left\| f - \textstyle\sum_{i=1}^{k} \frac{\langle u_i, g \rangle}{\mu_i} \, v_i \right\|_2 < \varepsilon.$$

The expansion in terms of an orthonormal basis,

$$f(t) = \sum_{i=1}^{\infty} \langle \phi_i, f \rangle \, \phi_i(t),$$

i.e., with expansion coefficients $\zeta_i = \langle \phi_i, f \rangle$, is a standard result in functional analysis; see, e.g., [50]. The SVE is "essentially unique," and by this we mean the following.

- When μ_i is isolated (i.e., different from its neighbors), then we can always replace the functions u_i and v_i with $-u_i$ and $-v_i$, because

$$\int_0^1 K(s,t)\ (-v_i(t))\ dt = -\int_0^1 K(s,t)\, v_i(t)\, dt = -\mu_i\, u_i(s) = \mu_i\, (-u_i(t))\,.$$

- When μ_i has multiplicity larger than one, i.e., when

$$\mu_{i-1} > \mu_i = \cdots = \mu_{i+\nu} > \mu_{i+\nu+1},$$

then the corresponding singular functions are not unique, but the subspaces span$\{u_i$, $\ldots, u_{i+\nu}\}$ and span$\{v_i, \ldots, v_{i+\nu}\}$ are unique. For example, if $\mu_1 = \mu_2$, then

$$\begin{aligned}\int_0^1 K(s,t)\ (\alpha\, v_1(t) + \beta\, v_2(t)) &= \alpha\, \mu_1\, u_1(s) + \beta\, \mu_2\, u_2(s) \\ &= \mu_1\, (\alpha\, u_1(s) + \beta\, u_2(s))\,,\end{aligned}$$

showing that we can always replace u_1 and v_1 with $\alpha\, u_1 + \beta\, u_2$ and $\alpha\, v_1 + \beta\, v_2$ as long as $\alpha^2 + \beta^2 = 1$ (to ensure that the new singular functions have unit 2-norm).

2.4 Ambiguity in Inverse Problems

As we mentioned in Chapter 1, inverse problems such as the first-kind Fredholm integral equation (2.2) are ill-posed. We have already seen that they may not have a solution, and that the solution is extremely sensitive to perturbations of the right-hand side g, thus illustrating the "existence" and "stability" issues of the Hadamard requirements for a well-posed problem.

The "uniqueness" issue of the Hadamard requirement is relevant because, as a matter of fact, not all first-kind Fredholm integral equations (2.2) have a unique solution. The nonuniqueness of a solution to an inverse problem is sometimes referred to as ambiguity, e.g., in the geophysical community, and we will adopt this terminology here. To clarify things, we need to distinguish between two kinds of ambiguity, as described below.

The first type of ambiguity can be referred to as *null-space ambiguity* or nonuniqueness of the solution. We encounter this kind of ambiguity when the integral operator has a null space, i.e., when there exist functions $f_{\text{null}} \neq 0$ such that, in the generic case,

$$\int_0^1 K(s,t)\, f_{\text{null}}(t)\, dt = 0.$$

Such functions are referred to as annihilators, and we note that if f_{null} is an annihilator, so is also any scalar multiple of this function. Annihilators are undesired, in the sense that if we are not careful, then we may have no control of their contribution to the computed solution.

The null space associated with the kernel $K(s,t)$ is the space spanned by all the annihilators. In the ideal case, the null space consists of the null function only; when this is the case we need not pay attention to it (similar to the treatment of linear algebraic equations with a nonsingular coefficient matrix). Occasionally, however, we encounter problems with a nontrivial null space, and it follows from (2.6) that this null space is spanned by the right singular functions $v_i(t)$ corresponding to the zero singular values $\mu_i = 0$. The dimension of the null space is finite if only a finite number of singular values are zero; otherwise the dimension is infinite, which happens when there is a finite number of nonzero singular values. In the latter case we say that the kernel is degenerate.

It is easy to construct an example of an integral operator that has a null space; if $K(s,t) = s + 2t$, then simple evaluation shows that

$$\int_{-1}^{1} (s + 2t)\,(3t^2 - 1)\,dt = 0,$$

i.e., $f_{\text{null}}(t) = 3t^2 - 1$ is an annihilator. In fact, this kernel is degenerate, and the null space is spanned by the Legendre polynomials of degree ≥ 2; see Exercise 2.2.

Annihilators are not just theoretical conceptions; they arise in many applications, such as potential theory in geophysics, acoustics, and electromagnetic theory. It is therefore good to know that (depending on the discretization) we are often able to detect and compute annihilators numerically, and in this way control (or remove) their contribution to the solution. Exercise 3.9 in the next chapter deals with the numerical computation of the one-dimensional null space associated with an integral equation that arises in potential theory.

Unfortunately, numerical computation of an annihilator may occasionally be a tricky business due to the discretization errors that arise when turning the integral equation into a matrix problem (Chapter 3) and due to rounding errors on the computer. Exercise 4.9 in Chapter 4 illustrates this aspect.

The second kind of ambiguity can be referred to as *formulation ambiguity*, and while it has another cause than the null-space ambiguity, and the two types are occasionally mixed up. The formulation ambiguity typically arises in applications where two different integral formulations, with two different kernels, lead to the same right-hand side function g associated with a physical quantity. A classical example comes from potential theory where we consider the field g outside a 3D domain Ω. Then Green's third identity states that any field g outside Ω can be produced by both a source distribution inside Ω and an infinitely thin layer of sources on the surface of Ω. Which formulation to prefer depends on which model best describes the physical problem under consideration.

How to deal with a formulation ambiguity is thus, strictly speaking, purely a modeling issue— and not a computational issue, as in the case of a null-space ambiguity. Unfortunately, as we shall see in Section 7.9, discretized problems and their regularized solutions may show a behavior that clearly resembles a formulation ambiguity. For this reason, it is important to be aware of both kinds of ambiguity.

2.5 Spectral Properties of the Singular Functions*

This is the only section of the book that requires knowledge about functional analysis. The section summarizes the analysis from [38], which provides a spectral characterization of the singular functions and the Fourier functions $e^{\hat{\imath}ks}/\sqrt{2\pi}$, where $\hat{\imath}$ denotes the imaginary unit. If you decide to skip this section, you should still pay attention to the spectral characterization as stated in (2.14), because this characterization is important for the analysis and algorithms developed throughout the book.

In contrast to existing asymptotic results, the point of this analysis is to describe behavior that is essentially *nonasymptotic*, since this is what is always captured upon discretization of a problem. The material is not a rigorous analysis; rather, it uses some tools from functional analysis to gain insight into nonasymptotic relations between the Fourier and singular functions of the integral operator. We define the integral operator $\mathcal{K}$ such that

$$[\mathcal{K}f](s) = \int_{-\pi}^{\pi} K(s,t)\, f(t)\, dt,$$

and we make the following assumptions.

1. We assume that the kernel K is real and $C^1([-\pi,\pi]\times[-\pi,\pi])$ (this can be weakened to piecewise C^1, carrying out the argument below on each piece), and that it is nondegenerate (i.e., it has infinitely many nonzero singular values).
2. For simplicity, we also assume that $\|K(\pi,t) - K(-\pi,t)\|_2 = 0$; this assumption can be weakened so that this quantity is of the order of the largest singular value of the integral operator.

Since the operator $\mathcal{K}$ is square integrable, it has left and right singular functions $u_j(x)$ and $v_j(x)$ and singular values $\mu_j > 0$, such that

$$[\mathcal{K}v_j](s) = \mu_j u_j(s), \qquad [\mathcal{K}^* u_j](t) = \mu_j v_j(t), \qquad j = 1, 2, \ldots.$$

Now, define the infinite matrix $\mathbf{B}$ with rows indexed from $k = -\infty, \ldots, \infty$ and columns indexed by $j = 1, \ldots, \infty$, and with entries

$$\mathbf{B}_{k,j} = \left| \left\langle u_j, e^{\hat{\imath}ks}/\sqrt{2\pi} \right\rangle \right|. \tag{2.13}$$

A similar analysis with v_j replacing u_j is omitted. We want to show that the largest entries in the matrix $\mathbf{B}$ have the generic shape $\boxed{<}$, i.e., that of a "V" lying on the side and the tip of the "V" located at indices $k = 0$, $j = 1$.

This means, for example, that the function u_1 is well represented by just the smallest values of k (i.e., the lowest frequencies). Moreover, for larger values of j the function u_j is well represented by just a small number of the Fourier functions $e^{\hat{\imath}ks}/\sqrt{2\pi}$ for some $|k|$ in a band of contiguous integers depending on j. This leads to the following general characterization (see Figure 2.5 for an example):

Spectral Characterization of the Singular Functions. The singular functions are similar to the Fourier functions in the sense that large singular values (for small j) and their corresponding singular functions correspond to low frequencies, while small singular values (for larger j) correspond to high frequencies. (2.14)

In order to show the above result, we first derive an estimate for the elements in the matrix $\mathbf{B}$. We have, for $k \neq 0$,

$$\begin{aligned}
\left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle &= \left\langle \frac{1}{\mu_j} \mathcal{K} v_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \\
&= \frac{1}{\mu_j} \left\langle v_j, \mathcal{K}^* \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \\
&= \frac{1}{\mu_j} \int_{-\pi}^{\pi} v_j(t) \int_{-\pi}^{\pi} K(s,t) \frac{e^{-\hat{\imath}ks}}{\sqrt{2\pi}} \, ds \, dt \\
&= \frac{1}{k\,\mu_j} \int_{-\pi}^{\pi} v_j(t) \left((-1)^{k+1} \big[K(\pi,t) - K(-\pi,t) \big] \right. \\
&\qquad\qquad \left. + \int_{-\pi}^{\pi} \frac{\partial K}{\partial s} \frac{e^{-\hat{\imath}ks}}{\sqrt{2\pi}} \, ds \right) dt,
\end{aligned}$$

where the last line is obtained through integration by parts. Therefore, using the second assumption,

$$\left| \left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \right| \leq \frac{1}{k\,\mu_j} \left\| \frac{\partial K}{\partial s} \frac{e^{-\hat{\imath}ks}}{\sqrt{2\pi}} \right\|_2 . \tag{2.15}$$

It is instructive to show that the term in the norm on the right hand side is roughly bounded above by μ_1. By definition of induced operator norms, we know that

$$\mu_1 = \max_{\|f\|_2 = 1} \|\mathcal{K} f\|_2,$$

and so in particular for $k = \pm 1$ we have $\mu_1 \geq \left\| \mathcal{K} e^{\pm \hat{\imath} s} / \sqrt{2\pi} \right\|_2$. Using integration by parts again, we find

$$\mu_1 \geq \left\| \frac{1}{\sqrt{2\pi}} \frac{\partial K}{\partial s} e^{\pm \hat{\imath} s} \right\|_2 .$$

From the Riemann–Lebesque lemma (see, e.g., Theorem 12.5C in [19]) it follows that

$$\left\| \frac{\partial K}{\partial s} \, e^{-\hat{\imath}ks} / \sqrt{2\pi} \right\|_2 \to 0 \qquad \text{for} \qquad k \to \infty.$$

Therefore, we also expect for $|k| > 1$ that

$$\mu_1 \gtrsim \left\| \frac{\partial K}{\partial s} \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\|_2 ,$$

so

$$\left| \left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \right| \lesssim \frac{\mu_1}{k \mu_j}. \tag{2.16}$$

The characterization of the matrix $\mathbf{B}$ is a consequence of (2.15), (2.16), and Bessel's and Parseval's inequalities and is argued by induction on j. The elements in any column of $\mathbf{B}$ satisfy

$$\sum_{k=-\infty}^{\infty} |\mathbf{B}_{kj}|^2 = \sum_{k=-\infty}^{\infty} \left| \left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \right|^2 = 1 \tag{2.17}$$

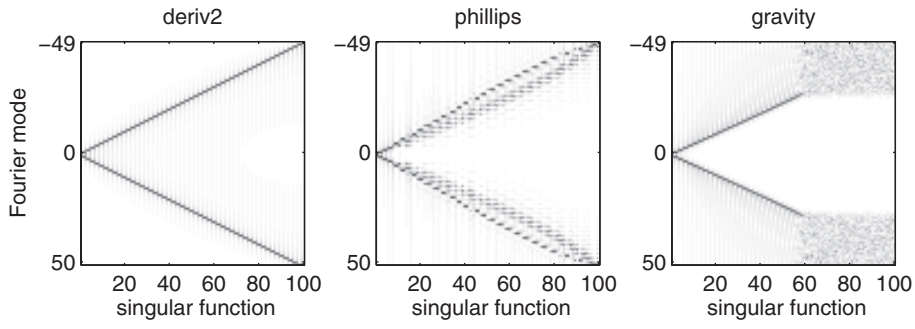

Figure 2.8. *Characterization of the singular functions. The three figures show sections of the infinite matrix* $\mathbf{B}$ (2.13) *for* $j = 1, \ldots, 100$ *and* $k = -49, \ldots, 50$ *for three test problems.*

by Parseval's equality, since $e^{\hat{\imath}ks}/\sqrt{2\pi}$ is a basis for $L^2([-\pi, \pi])$ and $u_j \in L^2([-\pi, \pi])$. Note that by orthonormality and Cauchy–Schwarz $\left|\left\langle u_j, e^{\hat{\imath}ks}/\sqrt{2\pi}\right\rangle\right| \leq 1$. Also, since the u_j form an orthonormal set in $L^2([-\pi, \pi])$, we have Bessel's inequality

$$\sum_{j=1}^{\infty} |\mathbf{B}_{kj}|^2 = \sum_{j=1}^{\infty} \left| \left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \right|^2 \leq 1, \tag{2.18}$$

showing that for the elements in each row in $\mathbf{B}$, the sum of squares is finite.

Consider first $j = 1$. Because of (2.15) and (2.16) the terms closest to 1 in magnitude in the first column of $\mathbf{B}$ occur only for k very close to $k = 0$. A little calculus on (2.18) using (2.16) shows that for any integer $\ell > 0$,

$$1 - \frac{2}{\ell} \lesssim \sum_{k=-\ell}^{\ell} \left| \left\langle u_j, \frac{e^{\hat{\imath}ks}}{\sqrt{2\pi}} \right\rangle \right|^2 .$$

Thus there must be some relatively large terms for k close to 0. So the desired behavior is observed for the base case, $j = 1$.

Now consider the jth column of $\mathbf{B}$ for any j such that $\mu_j \ll \mu_1$. By (2.16) we know that when k is large enough, say, $k > k^*$, then the entries $\mathbf{B}_{kj}$ will be small. But inequality (2.16) does not tell us about the behavior for k close to zero since $\mu_1/\mu_j \gg 1$. At this point, we know that the largest entries occur for indices $|k| \leq k^*$. However, large entries in the jth column cannot occur for rows where $|k| < j$ because of (2.18); that is, entries large in magnitude have already occurred in those rows in previous columns, and too many large entries in a row would make the sum larger than one. Therefore, the band of indices for which large entries can occur moves out from the origin as j increases, and we observe the pattern of a "tilted V" shape, as desired.

An analogous argument shows that the matrix involving inner products with v_j and the Fourier functions have a similar pattern.

Figure 2.8 shows sections of the infinite matrix $\mathbf{B}$ (2.13) for $j = 1, \ldots, 100$ and $k = -49, \ldots, 50$ for three test problems (to be introduced later). The computations

were done using techniques from Chapter 3, and both discretization effects and finite-precision effects are visible. The left plot, with the perfect "tilted V" shape, confirms that the singular functions of the second derivative problem (see Exercise 2.3) are indeed trigonometric functions. The middle and left plots show the same overall behavior: each singular function u_j is dominated by very few Fourier functions with $k \approx \pm j/2$. In the rightmost plot, the results for $j > 60$ are dominated by rounding errors, but note that the computed singular functions contain solely high-frequency Fourier modes.

2.6 The Story So Far

In this chapter we saw how the first-kind Fredholm integral equation (2.2) provides a linear model for the relationship between the source and the data, as motivated by a gravity surveying problem in geophysics. Through the Riemann–Lebesgue lemma we showed that for such models the solution can be arbitrarily sensitive to perturbations of the data, and we gave an example where a square integrable solution does not exist.

We then introduced the singular value expansion (SVE) as a powerful mathematical tool for analysis of first-kind Fredholm integral equations, and we used the SVE to discuss the existence and the instability of the solution. An important outcome of this analysis is the Picard condition (2.9) for existence of a square integrable solution. We also briefly discussed the ambiguity problem associated with zero singular values μ_i of first-kind Fredholm integral equations.

Finally, we gave a rigorous discussion of the spectral properties of the singular functions u_i and v_i and showed that these functions provide a spectral basis with an increasing number of oscillations as the index i increases. This result imbues the rest of the discussion and analysis in this book.

The main conclusion of our analysis in this chapter is that the right-hand side $g(s)$ must be sufficiently "smooth" (as measured by its SVE coefficients) and that small perturbations in the right-hand side are likely to produce "solutions" that are highly contaminated by undesired high-frequency oscillations.

Exercises

2.1. **Derivation of SVE Solution Expression**
Give the details of the derivation of the expression (2.8) for the solution $f(t)$ to the integral equation (2.2).

2.2. **The SVE of a Degenerate Kernel**
The purpose of this exercise is to illustrate how the singular value expansion (SVE) can give valuable insight into the problem. We consider the simple integral equation

$$\int_{-1}^{1} (s + 2t)\, f(t)\, dt = g(s), \quad -1 \leq s \leq 1; \tag{2.19}$$

i.e., the kernel is given by

$$K(s,t) = s + 2t.$$

This kernel is chosen solely for its simplicity. Show (e.g., by insertion) that the SVE of the kernel K is given by

$$\mu_1 = 4/\sqrt{3}, \quad \mu_2 = 2/\sqrt{3}, \quad \mu_3 = \mu_4 = \cdots = 0$$

and

$$u_1(s) = 1/\sqrt{2}, \quad u_2(s) = \sqrt{3/2}\,s, \quad v_1(t) = \sqrt{3/2}\,t, \quad v_2(t) = 1/\sqrt{2}.$$

Show that this integral equation has a solution only if the right-hand side is a linear function, i.e., it has the form $g(s) = \alpha s + \beta$, where α and β are constants. Hint: evaluate $\int_{-1}^{1} K(s,t)\,f(t)\,dt$ for an arbitrary function $f(t)$.

2.3. **The SVE of the Second Derivative Problem**

We can occasionally calculate the SVE analytically. Consider the first-kind Fredholm integral equation $\int_0^1 K(s,t)\,f(t)\,dt = g(s)$, $0 \le s \le 1$, with the kernel

$$K(s,t) = \begin{cases} s\,(t-1), & s < t, \\ t\,(s-1), & s \ge t. \end{cases} \tag{2.20}$$

This equation is associated with the computation of the second derivative of a function: given a function g, the solution f is the second derivative of g, i.e., $f(t) = g''(t)$ for $0 \le t \le 1$. It is available in *Regularization Tools* as the test problem `deriv2`. An alternative expression for the kernel is given by

$$K(s,t) = -\frac{2}{\pi^2} \sum_{i=1}^{\infty} \frac{\sin(i\,\pi\,s)\,\sin(i\,\pi\,t)}{i^2}.$$

Use this relation to derive the expressions

$$\mu_i = \frac{1}{(i\pi)^2}, \quad u_i(s) = \sqrt{2}\sin(i\pi s), \quad v_i(t) = -\sqrt{2}\sin(i\pi t), \qquad i = 1, 2, \ldots,$$

for the singular values and singular functions. Verify that the singular functions are orthonormal.

2.4. **The Picard Condition**

We continue the study of the second-derivative problem from the previous exercise, for which we know the SVE analytically. Consider the two right-hand sides and their Fourier series:

$$\begin{aligned} g_{\text{linear}}(s) &= s \\ &= \frac{2}{\pi} \sum_{i=1}^{\infty} (-1)^{i+1} \frac{\sin(i\,\pi s)}{i}, \\ g_{\text{cubic}}(s) &= s\,(1+s)\,(1-s) \\ &= \frac{12}{\pi^3} \sum_{i=1}^{\infty} (-1)^{i+1} \frac{\sin(i\,\pi s)}{i^3}. \end{aligned}$$

Which of these two functions satisfies the Picard condition?

Chapter 3

Getting to Business: Discretizations of Linear Inverse Problems

After our encounter with some of the fundamental theoretical aspects of the first-kind Fredholm integral equation, it is now time to study how we can turn this beast into something that can be represented and manipulated on a computer. We will discuss basic aspects of discretization methods, and we will introduce the matrix-version of the SVE, namely, the singular value decomposition (SVD). We finish with a brief look at noisy discrete problems.

All the matrix problems that we will encounter in this book will involve an $m \times n$ matrix A, a right-hand side b with m elements, and a solution vector x of length n. When $m = n$ the matrix is square, and the problem takes the form of a system of linear equations,

$$A\,x = b, \qquad A \in \mathbb{R}^{n\times n}, \quad x, b \in \mathbb{R}^n.$$

When $m > n$ the system is overdetermined, and our problem takes the form of a linear least squares problem,

$$\min_x \|A\,x - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m.$$

We will not cover the case of underdetermined systems (when $m < n$), mainly because of technicalities in the presentation and in the methods.

3.1 Quadrature and Expansion Methods

In order to handle continuous problems, such as integral equations, on a computer (which is designed to work with numbers) we must replace the problem of computing the unknown function f in the integral equation (2.2) with a discrete and finite-dimensional problem that we can solve on the computer. Since the underlying integral equation is assumed to be linear, we want to arrive at a system of linear algebraic equations, which we shall refer to as the *discrete inverse problem*. There are two basically different approaches to do this, and we summarize both methods below; a rigorous discussion of numerical methods for integral equations can be found in [4].

3.1.1 Quadrature Methods

These methods compute approximations $\tilde{f}_j$ to the solution f solely at selected abscissas $t_1, t_2, \ldots, t_n$, i.e.,

$$\tilde{f}_j = \tilde{f}(t_j), \qquad j = 1, 2, \ldots, n.$$

Notice the tilde: we compute sampled values of some function $\tilde{f}$ that approximates the exact solution f, but we do not obtain samples of f itself (unless it happens to be perfectly represented by the simple function underlying the quadrature rule).

Quadrature methods—also called Nyström methods—take their basis in the general quadrature rule of the form

$$\int_0^1 \varphi(t)\, dt = \sum_{j=1}^{n} \omega_j\, \varphi(t_j) + E_n,$$

where φ is the function whose integral we want to evaluate, E_n is the quadrature error, $t_1, t_2, \ldots, t_n$ are the abscissas for the quadrature rule, and $\omega_1, \omega_2, \ldots, \omega_n$ are the corresponding weights. For example, for the midpoint rule in the interval $[0, 1]$ we have

$$t_j = \frac{j - \frac{1}{2}}{n}, \qquad \omega_j = \frac{1}{n}, \qquad j = 1, 2, \ldots, n. \tag{3.1}$$

The crucial step in the derivation of a quadrature method is to apply this rule *formally* to the integral, pretending that we know the values $f(t_j)$. Notice that the result is still a function of s:

$$\psi(s) = \int_0^1 K(s, t)\, f(t)\, dt = \sum_{j=1}^{n} \omega_j\, K(s, t_j)\, f(t_j) + E_n(s);$$

here the error term E_n is a function of s. We now enforce the *collocation* requirement that the function ψ must be equal to the right-hand side g at n selected points:

$$\psi(s_i) = g(s_i), \qquad i = 1, \ldots, n,$$

where $g(s_i)$ are sampled or measured values of the function g. This leads to the following system of relations:

$$\sum_{j=1}^{n} \omega_j\, K(s_i, t_j)\, f(x_j) = g(s_i) - E_n(s_i), \quad i = 1, \ldots, n.$$

Finally we must, of course, neglect the unknown error term $E_n(s_i)$ in each relation, and in doing so we introduce an error in the computed solution. Therefore, we must replace the exact values $f(t_j)$ with approximate values, denoted by $\tilde{f}_j$, and we arrive at the relations

$$\sum_{j=1}^{n} \omega_j\, K(s_i, t_j)\, \tilde{f}_j = g(s_i), \quad i = 1, \ldots, n. \tag{3.2}$$

We note in passing that we could have used $m > n$ collocation points, in which case we would obtain an overdetermined system. In this presentation, for simplicity we always use $m = n$.

The relations in (3.2) are just a linear system of linear equations in the unknowns

$$\tilde{f}_j = \tilde{f}(t_j), \qquad j = 1, 2, \ldots, n.$$

To see this, we can write the n equations in (3.2) in the form

$$\begin{pmatrix} \omega_1 K(s_1,t_1) & \omega_2 K(s_1,t_2) & \cdots & \omega_n K(s_1,t_n) \\ \omega_1 K(s_2,t_1) & \omega_2 K(s_2,t_2) & \cdots & \omega_n K(s_2,t_n) \\ \vdots & \vdots & & \vdots \\ \omega_1 K(s_n,t_1) & \omega_2 K(s_n,t_2) & \cdots & \omega_n K(s_n,t_n) \end{pmatrix} \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_n \end{pmatrix} = \begin{pmatrix} g(s_1) \\ g(s_2) \\ \vdots \\ g(s_n) \end{pmatrix}$$

or simply $A\,x = b$, where A is an $n \times n$ matrix. The elements of the matrix A, the right-hand side b, and the solution vector x are given by

$$\left.\begin{aligned} a_{ij} &= \omega_j\, K(s_i, t_j) \\ x_j &= \tilde{f}(t_j) \\ b_i &= g(s_i) \end{aligned}\right\} \qquad i, j = 1, \ldots, n. \tag{3.3}$$

3.1.2 Expansion Methods

These methods compute an approximation of the form

$$f^{(n)}(t) = \sum_{j=1}^{n} \zeta_j\, \phi_j(t), \tag{3.4}$$

where $\phi_1(t), \ldots, \phi_n(t)$ are expansion functions or basis functions chosen by the user. They should be chosen in such a way that they provide a good description of the solution; but no matter how good they are, we still compute an approximation $f^{(n)}$ living in the span of these expansion functions.

Expansion methods come in several flavors, and we will focus on the Galerkin (or Petrov–Galerkin) method, in which we must choose two sets of functions ϕ_i and ψ_j for the solution and the right-hand side, respectively. Then we can write

$$\begin{aligned} f(t) &= f^{(n)}(t) + E_f(t), & f^{(n)} &\in \operatorname{span}\{\phi_1, \ldots, \phi_n\}, \\ g(s) &= g^{(n)}(s) + E_g(s), & g^{(n)} &\in \operatorname{span}\{\psi_1, \ldots, \psi_n\}, \end{aligned}$$

where E_f and E_g are the expansion errors for f and g due to the use of a finite set of basis vectors. To be more specific, we write the approximate solution $f^{(n)}$ in the form (3.4), and we want to determine the expansion coefficients ζ_j such that $f^{(n)}$ is an approximate solution to the integral equation. We therefore introduce the function

$$\vartheta(s) = \int_0^1 K(s,t)\, f^{(n)}(t)\, dt = \sum_{j=1}^{n} \zeta_j \int_0^1 K(s,t)\, \phi_j(t)\, dt$$

and, similarly to f and g, we write this function in the form

$$\vartheta(s) = \vartheta^{(n)}(s) + E_\vartheta(s), \qquad \vartheta^{(n)} \in \operatorname{span}\{\psi_1, \ldots, \psi_n\}.$$

How do we choose the functions ϕ_i and ψ_j? The basis functions ϕ_i for the solution should preferably reflect the knowledge we may have about the solution f (such as smoothness, monotonicity, periodicity, etc.). Similarly, the basis functions ψ_j should capture the main information in the right-hand side. Sometimes, for convenience, the two sets of functions are chosen to be identical.

The key step in the Galerkin method is to recognize that, in general, the function ϑ is not identical to g, nor does ϑ lie in the subspace $\text{span}\{\psi_1, \ldots, \psi_n\}$ in which $g^{(n)}$ lies (leading to the error E_ϑ). Let $\vartheta^{(n)}$ and $g^{(n)}$ in the above equations be the orthogonal projections of ϑ and g on the subspace $\text{span}\{\psi_1, \ldots, \psi_n\}$ (this ensures that they are unique). The Galerkin approach is then to determine the expansion coefficients ζ_j such that the two functions $\vartheta^{(n)}$ and $g^{(n)}$ are identical:

$$\begin{aligned} \vartheta^{(n)}(s) &= g^{(n)}(s) && \Leftrightarrow \\ \vartheta(s) - E_\vartheta(s) &= g(s) - E_g(s) && \Leftrightarrow \\ \vartheta(s) - g(s) &= E_\vartheta(s) - E_g(s). \end{aligned}$$

Here, we recognize the function $\vartheta(s) - g(s)$ as the residual. Since both functions $E_\vartheta(s)$ and $E_g(s)$ are orthogonal to the subspace $\text{span}\{\psi_1, \ldots, \psi_n\}$, the Galerkin condition can be stated as the requirement that the residual $\vartheta(s) - g(s)$ is orthogonal to each of the functions $\psi_1, \ldots, \psi_n$.

In order to arrive at a system of equations for computing the unknowns $\zeta_1, \ldots, \zeta_n$, we now enforce the orthogonality of the residual and the ψ_i-functions: That is, we require that

$$\langle \psi_i \, , \, \vartheta - g \rangle = 0 \qquad \text{for} \qquad i = 1, \ldots, n.$$

This is equivalent to requiring that

$$\langle \psi_i \, , \, g \rangle = \langle \psi_i \, , \, \vartheta \rangle = \left\langle \psi_i \, , \, \int_0^1 K(s,t)\, f^{(n)}(t)\, dt \right\rangle, \qquad i = 1, \ldots, n.$$

Inserting the expansion for $f^{(n)}$, we obtain the relation

$$\langle \psi_i \, , \, g \rangle = \sum_{j=1}^{n} \zeta_j \left\langle \psi_i \, , \, \int_0^1 K(s,t)\, \phi_j(t)\, dt \right\rangle, \qquad i = 1, \ldots, n,$$

which, once again, is just an $n \times n$ system of linear algebraic equations $Ax = b$ whose solution coefficients are the unknown expansion coefficients, i.e., $x_i = \zeta_i$ for $i = 1, \ldots, n$, and the elements of A and b are given by

$$a_{ij} = \int_0^1 \int_0^1 \psi_i(s)\, K(s,t)\, \phi_j(t)\, ds\, dt, \tag{3.5}$$

$$b_i = \int_0^1 \psi_i(s)\, g(s)\, ds. \tag{3.6}$$

If these integrals are too complicated to be expressed in closed form, they must be evaluated numerically (by means of a suited quadrature formula). Notice that if K is

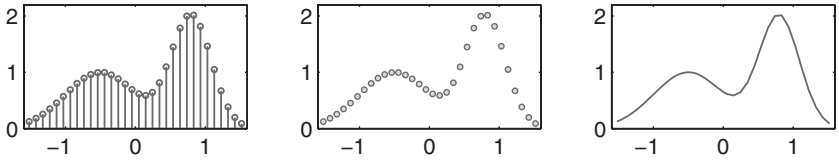

Figure 3.1. *Three different ways to plot the solution computed by means of a quadrature method (using the test problem* `shaw` *from Exercise* 3.6*). In principle, only the left and middle plots are correct.*

symmetric and $\psi_i = \phi_i$, then the matrix A is symmetric (and the approach is called the Rayleigh–Ritz method).

To illustrate the expansion method, let us consider a very simple choice of orthonormal basis functions, namely, the "top hat" ⊓⊔ functions (or scaled indicator functions) on an equidistant grid with interspacing $h = 1/n$:

$$\chi_i(t) = \begin{cases} h^{-1/2}, & t \in [\,(i-1)h\,,\; ih\,], \\ 0 & \text{elsewhere,} \end{cases} \qquad i = 1, 2, \ldots, n. \tag{3.7}$$

These functions are clearly orthogonal (because their supports do not overlap), and they are scaled to have unit 2-norm. Hence, by choosing

$$\phi_i(t) = \chi_i(t) \qquad \text{and} \qquad \psi_i(s) = \chi_i(s) \qquad \text{for} \qquad i = 1, 2, \ldots, n,$$

we obtain two sets of orthonormal basis functions for the Galerkin method. The matrix and right-hand side elements are then given by

$$a_{ij} = h^{-1} \int_{(i-1)h}^{ih} \int_{(j-1)h}^{jh} K(s,t)\, ds\, dt,$$

$$b_i = h^{-1/2} \int_{(i-1)h}^{ih} g(s)\, ds.$$

Again, it may be necessary to use quadrature rules to evaluate these integrals.

3.1.3 Which Method to Choose?

This is a natural question for a given application. The answer depends on what is needed, what is given, and what can be computed.

Quadrature methods are simpler to use and implement than the expansion methods, because the matrix elements q_{ij} are merely "samples" of the kernel K at distinct points. Also, the particular quadrature method can be chosen to reflect the properties of the functions K and f; for example, we can take into account if K is singular. Similarly, if the integration interval is infinite, then we can choose a quadrature method that takes this into account.

One important issue to remember is that we compute approximate function values $\tilde{f}_j$ only at the quadrature points; we have information about the solution between

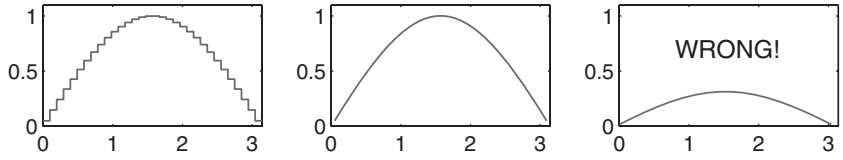

Figure 3.2. *Three different ways to plot the solution computed by means of an expansion method using "top hat" basis functions (using the test problem* `baart`*). In principle, only the left plot is correct. The right plot is wrong for two reasons: The scaling by* $h^{-1/2}$ *is missing, and abscissa values are not located symmetrically in the* t*-interval* $[0, \pi]$.

these points. Hence, one can argue that the solutions computed by quadrate methods should be plotted as shown in the left or middle plots of Figure 3.1; but as n increases the difference is marginal.

Expansion methods can be more difficult to derive and implement, because the matrix elements are double integrals, and sometimes numerical quadrature or other approximations must be used. The advantage, on the other hand, is that if the basis functions are orthonormal, then we have a well-understood relationship between the SVE and the SVD of the matrix. Another advantage is that the basis functions can be tailored to the application (e.g., we can use spline functions, thin plate smoothing splines, etc.).

Expansion methods produce an approximate solution that is known everywhere on the t-interval, because $f^{(n)}$ is expressed in terms of the basis functions $\phi_j(t)$. Thus, one can argue that plots of the solution should always reflect this (see Figure 3.2 for an example using the "top hat" basis functions); again, as n increases the difference is marginal. What is important, however, is that any scaling in the basis functions should be incorporated when plotting the solution.

In applications, information about the right-hand side g is almost always available solely in the form of (noisy) samples $b_i = g(s_i)$ of this function. This fits very well with the quadrature methods using collocation in the sampling points s_i. For Galerkin methods there are two natural choices of the basis functions ψ_i for the right-hand side. The choice of delta functions $\psi_i(s) = \delta(s - s_i)$ located at the sampling points represents a perfect sampling of g because, by definition,

$$\int_0^1 \delta(s - s_i)\, g(s)\, ds = g(s_i).$$

The choice of the "top-hat" functions $\psi_i(s) = \chi_i(s)$ (3.7) corresponds better to how real data are collected, namely, by basically integrating a signal over a short interval.

3.2 The Singular Value Decomposition

From the discussion in the previous chapter it should be clear that the SVE is a very powerful tool for analyzing first-kind Fredholm integral equations. In the discrete setting, with finite-dimensional matrices, there is a similarly powerful tool known as

the *singular value decomposition* (SVD). See, e.g., [5] or [67] for introductions to the SVD.

While the matrices that we encountered in the previous section are square, we have already foreshadowed that we will treat the more general case of rectangular matrices. Hence, we assume that the matrix is either square or has more rows than columns. Then, for any matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, the SVD takes the form

$$\boxed{A = U\,\Sigma\,V^T = \sum_{i=1}^{n} u_i\,\sigma_i\,v_i^T.} \tag{3.8}$$

Here, $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix with the *singular values*, satisfying

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n), \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

Two important matrix norms are easily expressed in terms of the singular values:

$$\|A\|_{\mathrm{F}} \equiv \left(\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right)^{1/2} = \big(\mathrm{trace}(A^TA)\big)^{1/2} = \big(\mathrm{trace}(V\,\Sigma^2V^T)\big)^{1/2}$$

$$= \big(\mathrm{trace}(\Sigma^2)\big)^{1/2} = \big(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2\big)^{1/2}$$

$$\|A\|_2 \equiv \max_{\|x\|_2=1} \|Ax\|_2 = \sigma_1.$$

(The derivation of the latter expression is omitted here.) The matrices $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ consist of the left and right *singular vectors*

$$U = (u_1, \ldots, u_n), \qquad V = (v_1, \ldots, v_n),$$

and both matrices have orthonormal columns:

$$u_i^T u_j = v_i^T v_j = \delta_{ij}, \qquad i, j = 1, \ldots, n,$$

or simply

$$U^TU = V^TV = I.$$

It is straightforward to show that the inverse of A (if it exists) is given by

$$A^{-1} = V\,\Sigma^{-1}U^T.$$

Thus we have $\|A^{-1}\|_2 = \sigma_n^{-1}$, and it follows immediately that the 2-norm condition number of A is given by

$$\mathrm{cond}(A) = \|A\|_2\,\|A^{-1}\|_2 = \sigma_1/\sigma_n.$$

The same expression holds if A is rectangular and has full rank.

Software for computing the SVD is included in all serious program libraries, and Table 3.1 summarizes some of the most important implementations. The complexity of all these SVD algorithms is $\mathcal{O}(m\,n^2)$ flops, as long as $m \geq n$. There is also software, based on Krylov subspace methods, for computing selected singular values of large sparse or structured matrices; for example, MATLAB's `svds` function belongs in this class.

Table 3.1. *Some available software for computing the SVD.*

Software package	Subroutine
ACM TOMS	`HYBSVD`
EISPACK	`SVD`
GNU Sci. Lib.	`gsl_linalg_SV_decomp`
IMSL	`LSVRR`
LAPACK	`_GESVD`
LINPACK	`_SVDC`
NAG	`F02WEF`
Numerical Recipes	`SVDCMP`
MATLAB	`svd`

3.2.1 The Role of the SVD

The singular values and vectors obey a number of relations which are similar to the SVE. Perhaps the most important relations are

$$A\,v_i = \sigma_i\,u_i, \qquad \|A\,v_i\|_2 = \sigma_i, \qquad i = 1, \ldots, n, \tag{3.9}$$

and, if A is square and nonsingular,

$$A^{-1}u_i = \sigma_i^{-1}\,v_i, \qquad \|A^{-1}u_i\|_2 = \sigma_i^{-1}, \qquad i = 1, \ldots, n. \tag{3.10}$$

We can use some of these relations to derive an expression for the solution $x = A^{-1}b$. First note that since the matrix V is orthogonal, we can always write the vector x in the form

$$x = V\,V^T x = V \begin{pmatrix} v_1^T x \\ \vdots \\ v_n^T x \end{pmatrix} = \sum_{i=1}^{n} (v_i^T x)\,v_i,$$

and similarly for b we have

$$b = \sum_{i=1}^{n} (u_i^T b)\,u_i.$$

When we use the expression for x, together with the SVD, we obtain

$$A\,x = \sum_{i=1}^{n} \sigma_i\,(v_i^T x)\,u_i.$$

By equating the expressions for $A\,x$ and b, and comparing the coefficients in the expansions, we immediately see that the "naive" solution is given by

$$x = A^{-1}b = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i}\,v_i. \tag{3.11}$$

This expression can also be derived via the SVD relation for A^{-1}, but the above derivation parallels that of the SVE. For the same reasons as in the previous chapter

(small singular values σ_i and noisy coefficients $u_i^T b$), we do not want to compute this solution to discretizations of ill-posed problems.

While the SVD is now a standard tool in signal processing and many other fields of applied mathematics, and one which many students will encounter during their studies, it is interesting to note than the SVD was developed much later than the SVE; cf. [63].

3.2.2 Symmetries*

The kernel in the integral equation (2.2) is symmetric if $K(s,t) = K(t,s)$. Whether the corresponding matrix A inherits this symmetry, such that $A^T = A$, depends on the discretization method used to obtain A; but this is recommended because we want the linear system of equations to reflect the integral equation.

Symmetric matrices have real eigenvalues and orthonormal eigenvectors: $A = U\,\mathrm{diag}(\lambda_1, \ldots, \lambda_n)\,U^T$ with U orthogonal. It follows immediately that the SVD of A is related to the eigenvalue decomposition as follows: U is the matrix of eigenvectors, and

$$\Sigma = \mathrm{diag}(|\lambda_1|, \ldots, |\lambda_n|), \qquad V = \big(\,\mathrm{sign}(\lambda_1)\,u_1, \ldots, \mathrm{sign}(\lambda_n)\,u_n\,\big).$$

In particular, $v_i = \mathrm{sign}(\lambda_i)\,u_i$ for $i = 1, \ldots, n$.

Another kind of symmetry sometimes arises: we say that a matrix is persymmetric if it is symmetric across the antidagonal. In other words, if J is the exchange matrix,

$$J = \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix},$$

then persymmetry is expressed as

$$A\,J = (A\,J)^T = J\,A^T \qquad \Leftrightarrow \qquad A = J\,A^T\,J.$$

Persymmetry of A dictates symmetry relations between the singular vectors. To derive these relations, let the symmetric matrix $A\,J$ have the eigenvalue decomposition $A\,J = U\,\Gamma\,U^T$, where U is orthogonal; then

$$A = U\,\Gamma\,U^T J = U\,\Gamma\,(J\,U)^T = \sum_{i=1}^{n} u_i\,|\gamma_i|\,(\mathrm{sign}(\gamma_i)\,J\,u_i)^T,$$

and it follows that

$$v_i = \mathrm{sign}(\gamma_i)\,J\,u_i, \qquad i = 1, \ldots, n.$$

This means that, except perhaps for a sign change, the right singular vector v_i is identical to the left singular vector u_i with its elements in reverse order.

If A is both symmetric and persymmetric, then additional symmetries occur because $u_i = \mathrm{sign}(\lambda_i)\,v_i = \mathrm{sign}(\lambda_i)\,\mathrm{sign}(\gamma_i)\,J\,u_i$ for $i = 1, \ldots, n$. In words, the left and right singular vectors of a symmetric and persymmetric matrix are identical except perhaps for a sign change, and the sequence of elements in each vector is symmetric around the middle elements except perhaps for a sign change.

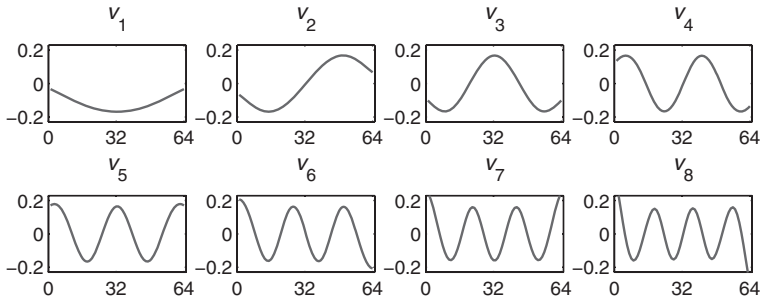

Figure 3.3. *The right singular vectors for the* `phillips` *test problem with* $n = 64$. *Note that* $v_i = J\,v_i$ *for* $i = 1, 3, 5, 7$ *and* $v_i = -J\,v_i$ *for* $i = 2, 4, 6, 8$.

We illustrate the above with the `phillips` problem from *Regularization Tools*, which is one of the classical test problems from the literature. It is stated as Example 1 in [58] with several misprints; the equations below are correct. Define the function

$$\varphi(x) = \begin{cases} 1 + \cos\left(\frac{\pi x}{3}\right), & |x| < 3, \\ 0, & |x| \geq 3. \end{cases}$$

Then the kernel $K(s,t)$, the solution $f(t)$, and the right-hand side $g(s)$ are given by

$$\begin{aligned} K(s,t) &= \varphi(s-t), \\ f(t) &= \varphi(t), \\ g(s) &= \left(6 - |s|\right)\left(1 + \frac{1}{2}\cos\left(\frac{\pi\, s}{3}\right)\right) + \frac{9}{2\,\pi}\,\sin\left(\frac{\pi\,|s|}{3}\right). \end{aligned} \tag{3.12}$$

Both integration intervals are $[-6, 6]$. The matrix A produced by `phillips` is symmetric and persymmetric, and Figure 3.3 shows some of its singular vectors for $n = 64$; notice the symmetries around the middle elements. Since the solution is symmetric, $x = J\,x$, only the singular vectors $v_1, v_3, v_5, v_7, \ldots$ contribute to x.

3.2.3 Nitty-Gritty Details of the SVD*

Similar to the SVE, the SVD of a given real matrix A is "essentially unique" in the following sense.

- When σ_i is an isolated singular value, i.e., it differs from its neighbors, then the corresponding singular vectors are unique up to a sign change; we may replace u_i and v_i with $-u_i$ and $-v_i$ (both pairs satisfy$A\,v_i = \sigma_i\,u_i$).
- For multiple singular values $\sigma_{i-1} > \sigma_i = \cdots = \sigma_{i+\nu} > \sigma_{i+\nu+1}$, the two subspaces $\mathrm{span}\{u_i, \ldots, u_{i+\nu}\}$ and $\mathrm{span}\{v_i, \ldots, v_{i+\nu}\}$ are uniquely determined, but the individual singular vectors are not (except that they must always satisfy $A\,v_i = \sigma_i\,u_i$).

For this reason, different computer programs for computing the SVD may give different results, and this is not a flaw in the software.

If the matrix A has more rows than columns, i.e., if $m > n$, then the SVD comes in two variants. The one used in (3.8) is often referred to as the "thin SVD" since the matrix $U \in \mathbb{R}^{m \times n}$ is rectangular. There is also a "full SVD" of the form

$$A = (U , U_\perp) \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T,$$

where the left singular matrix is $m \times m$ and orthogonal, and the middle matrix is $m \times n$. Both versions provide the same information for inverse problems.

If the matrix A has fewer rows than columns, i.e., if $m < n$, then the SVD takes the form $A = U \Sigma V^T = \sum_{i=1}^{m} u_i \, \sigma_i \, v_i^T$, where now $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times m}$, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_m) \in \mathbb{R}^{m \times m}$, and $\mathrm{cond}(A) = \sigma_1 / \sigma_m$.

3.3 SVD Analysis and the Discrete Picard Condition

The linear systems obtained from discretization of first-kind Fredholm integral equations always exhibit certain characteristics in terms of the SVD, which we will discuss in this section. These characteristics are closely connected to the properties of the integral equation, as expressed through the SVE (see Section 2.3). If you did not read that section, please skip the first part of this section.

The close relationship between the SVD and the SVE allows us to use the SVD, together with the Galerkin discretization technique, to compute approximations to the SVE (see [25]). Specifically, if we compute the matrix A according to (3.5), then the singular values σ_i of A are approximations to the singular values μ_i of K. More precisely, if the Galerkin basis functions are orthonormal, if we define the positive quantity Δ_n such that

$$\Delta_n^2 = \|K\|_2^2 - \|A\|_{\mathrm{F}}^2, \qquad \|K\|_2^2 \equiv \int_0^1 \int_0^1 |K(s,t)|^2 \, ds \, dt,$$

and if we use $\sigma_i^{(n)}$ to denote the singular values of the $n \times n$ matrix A, then it can be shown that

$$0 \leq \mu_i - \sigma_i^{(n)} \leq \Delta_n, \qquad i = 1, \ldots, n, \tag{3.13}$$

and

$$\sigma_i^{(n)} \leq \sigma_i^{(n+1)} \leq \mu_i, \qquad i = 1, \ldots, n. \tag{3.14}$$

That is, the singular values of A are increasingly better approximations to those of K, and the error is bounded by the quantity Δ_n.

We can also compute approximations to the left and right singular functions via the SVD. Define the functions

$$u_j^{(n)}(s) = \sum_{i=1}^{n} u_{ij} \, \psi_i(s), \quad v_j^{(n)}(t) = \sum_{i=1}^{n} v_{ij} \, \phi_i(t), \qquad j = 1, \ldots, n. \tag{3.15}$$

Then these functions converge to the singular functions, i.e., $u_j^{(n)}(s) \to u_j(s)$ and $v_j^{(n)}(t) \to v_j(t)$, as $n \to \infty$. For example, it can be shown that

$$\max\left\{\|u_1 - u_1^{(n)}\|_2\,,\ \|v_1 - v_1^{(n)}\|_2\right\} \leq \left(\frac{2\,\Delta_n}{\mu_1 - \mu_2}\right)^{1/2}.$$

Note that this bound depends on the separation between the first and the second singular values. Similar, but more complicated, bounds exist for the other singular functions.

A nice, and important, consequence of this property of the singular functions is that the SVD coefficients $u_i^T b$ are approximations to the corresponding inner products (u_i, g) in the SVE basis. To see this, we first note that the inner products $\langle u_j^{(n)}, g^{(n)}\rangle$ become increasingly better approximations to $\langle u_i, g\rangle$. We now insert the expressions for $u_j^{(n)}$ and $g^{(n)}$ into the definition of their inner product:

$$\begin{aligned}\left\langle u_j^{(n)}, g^{(n)}\right\rangle &= \int_0^1 \sum_{i=1}^{n} u_{ij}\,\psi_i(s)\ \sum_{k=1}^{n} b_k\,\psi_k(s)\,ds \\ &= \sum_{i=1}^{n} u_{ij} \sum_{k=1}^{n} b_k\,\langle \psi_i, \psi_k\rangle = \sum_{i=1}^{n} u_{ij}\,b_i = u_j^T b.\end{aligned}$$

Hence we can study all the important SVE relations by computing the SVD of the matrix A, and in particular we can use the numerical quantities to investigate whether the Picard condition seems to be satisfied.

To illustrate the characteristics of discrete inverse problems, we consider a discretization of the gravity surveying model problem (from Section 2.1) by means of the midpoint quadrature rule; see Exercise 3.5 for details. Figure 3.4 shows the singular values for a discretization with $n = 64$. Clearly, the singular values decay to zero with no gap anywhere in the spectrum. For $i \geq 55$ they tend to level off, which is due to the influence of the finite-precision arithmetic (these tiny singular values are smaller than the machine precision times the largest singular value σ_1, and they cannot be trusted). The condition number of A is of the order 10^{18}, which, for all practical purposes, is infinite.

Figure 3.4 also shows the singular values of the matrix obtained by discretization of the second derivative problem in (2.20) in Exercise 2.3. In accordance with the theory mentioned in the previous chapter, these singular values decay slower because the kernel K for this problem is less smooth than the kernel for the gravity problem. The kernel K in (2.20) is not differentiable at $s = t$.

Figure 3.5 shows the first nine left singular vectors u_i for the gravity surveying problem; the right singular functions are identical, because the matrix is symmetric. We see that the singular vectors have more oscillations as the index i increases, i.e., as the corresponding σ_i decrease (for this problem, the number of sign changes in the ith singular vector is $i - 1$). We can safely conclude that the singular functions behave in the same manner. We note, however, that while all 64 singular vectors are orthonormal, by construction, the last 10 vectors do not carry reliable information

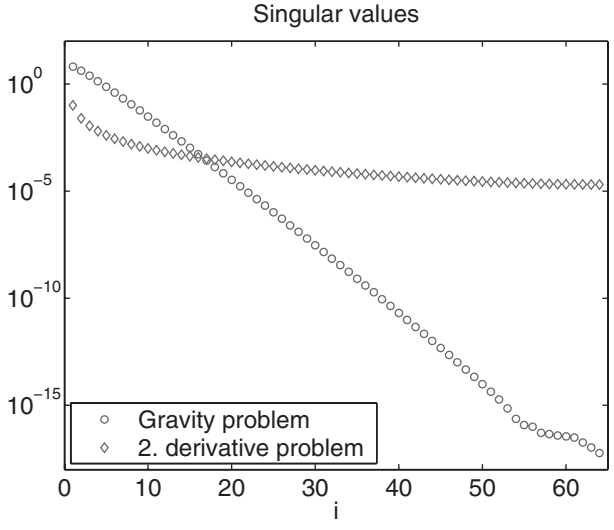

Figure 3.4. *The singular values σ_i of two 64×64 matrices A obtained from discretization of the gravity surveying problem (circles) and the second derivative problem from Exercise* 2.3 *(diamonds).*

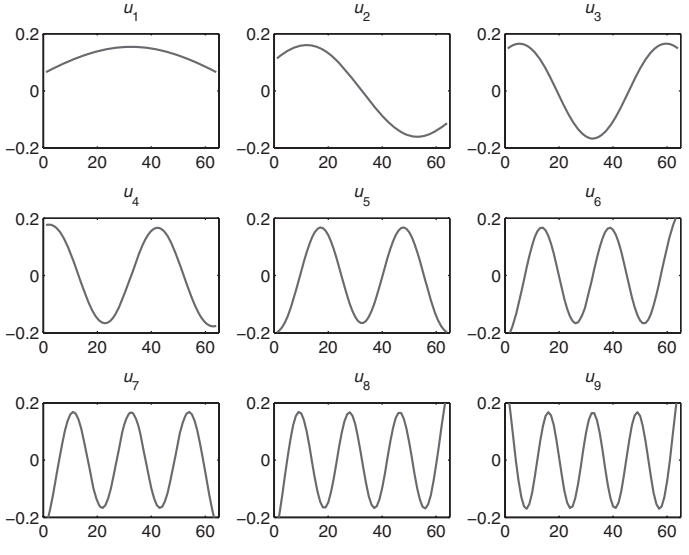

Figure 3.5. *The first* 9 *left singular vectors u_i of the same matrix as in Figure* 3.4.

about the singular functions (because the corresponding singular values are so tiny, as seen in Figure 3.4).

Finally, let us investigate the behavior of the SVD coefficients $u_i^T b$ (of the right-hand side) and $u_i^T b/\sigma_i$ (of the solution). A plot of these coefficients, together with

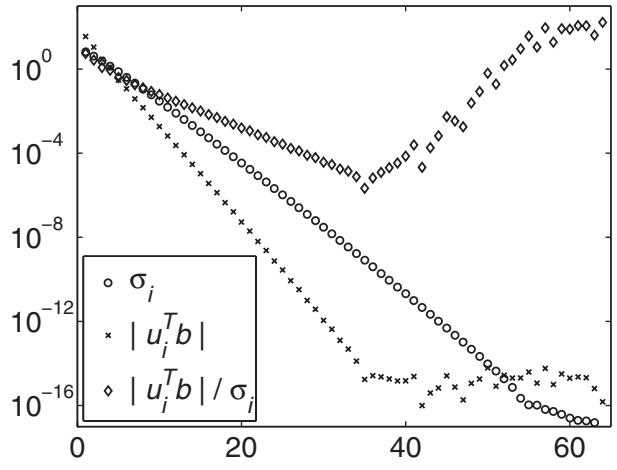

Figure 3.6. *The Picard plot for the discretized gravity surveying problem with no noise in the right-hand side.*

the singular values, is often referred to as a *Picard plot*, and Figure 3.6 shows such a plot for the discretized gravity surveying problem for a noise-free right-hand side. Clearly, the coefficients $|u_i^T b|$ decay faster than the singular values, until they level off for $i \geq 35$ at a plateau determined by the machine precision, and these tiny coefficients are less than the machine precision times $\|b\|_2$. The solution coefficients $u_i^T b/\sigma_i$ also decay for $i < 35$, but for $i \geq 35$ they start to increase again due to the inaccurate values of $u_i^T b$.

This illustrates that even without errors in the data, we cannot always expect to compute a meaningful solution to the discrete inverse problem. The influence of rounding errors in the computation of the SVD is enough to destroy the computed solution. We note that the same is true if the solution $x = A^{-1}b$ is computed via Gaussian elimination; only the influence of rounding errors leads to a different, but still wrong, solution.

Figure 3.7 shows two Picard plots for the gravity problem with two noisy right-hand sides, obtained by adding Gaussian white noise with zero mean to the vector Ax. The standard deviation of the noise is larger in the right plot. Again, we see an initial decay of the SVD coefficients $|u_i^T b|$, until they level off at a plateau determined by the noise. The solution coefficients also decrease initially, but for larger values of the index i they start to increase again. The computed solutions are completely dominated by the SVD coefficients corresponding to the smallest singular values.

This kind of analysis of the SVD coefficients, together with an understanding of their relation to the SVE coefficients, leads to the introduction of a discrete version of the Picard condition. While the solution of a finite-dimensional problem can never become unbounded, its norm can become so large that it is useless in practice. Since the SVD coefficients are so closely related to their corresponding SVE coefficients, it is relevant and necessary to introduce the following condition for the discretized

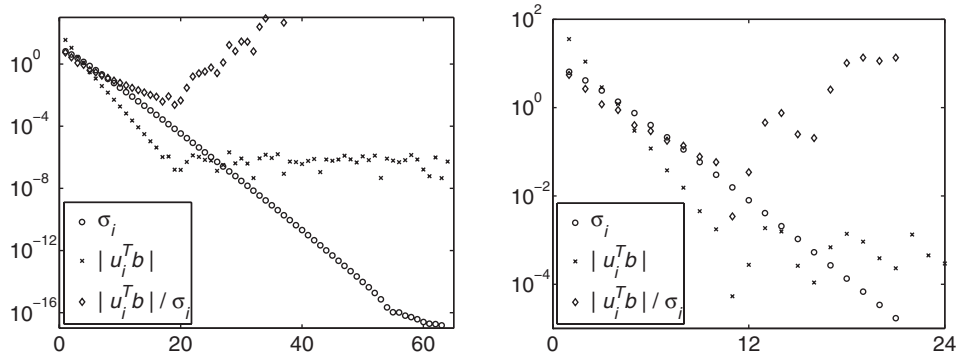

Figure 3.7. *The Picard plots for the discretized gravity surveying problem with a noisy right-hand side. The two figures correspond to two different noise levels. The larger the noise level, the fewer SVD coefficients $u_i^T b$ remain above the noise level.*

problem (originally proposed and studied in [27]).

The Discrete Picard Condition. Let τ denote the level at which the computed singular values σ_i level off due to rounding errors. The discrete Picard condition is satisfied if, for all singular values larger than τ, the corresponding coefficients $|u_i^T b|$, on average, decay faster than the σ_i. (3.16)

We emphasize that it is the *decay* of the σ_i and $|u_i^T b|$ that matters here—not the size of these quantities. For example, in Figure 3.7 we see that $|u_i^T b| > \sigma_i$ for $i = 1, 2, 3, 4$, but the discrete Picard condition is still satisfied for $i = 1, \ldots, 10$.

A visual inspection of the Picard plot is often enough to verify if the discrete Picard condition is satisfied. One should always ignore the part of the Picard plot that corresponds to tiny singular values at the machine precision level, and when noise is present in the right-hand side then one should also ignore all those coefficients for which $|u_i^T b|$ level off at some noise plateau.

To further illustrate the power of the analysis that can be carried out by means of the Picard plot, we return to Ursell's test problem (2.10). Figure 3.8 shows the Picard plot for a discretization of this problem (computed with $n = 16$). Clearly, the discrete Picard condition is not satisfied; the coefficients $|u_i^T b|$ decay more slowly than the singular values σ_i for all singular values larger than the machine precision level. This is a very strong indication that the underlying continuous problem does not satisfy the Picard condition. And this is, indeed, the case for this problem.

3.4 Convergence and Nonconvergence of SVE Approximation*

We mentioned in Section 2.3 that for the singular value expansion (SVE) to exist, the kernel K must be square integrable, that is, for our generic integral equation (2.2) we

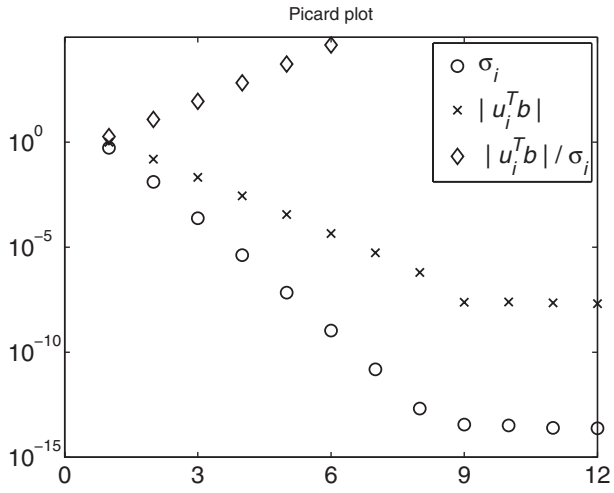

Figure 3.8. *The Picard plots for the discretized version of Ursell's model problem* (2.10)*. Clearly, the Picard condition is not satisfied.*

must have

$$\int_0^1 \int_0^1 K(x,t)^2 \, ds \, dt < \infty.$$

In this section we illustrate the importance of this requirement with a numerical example, where we use the SVD to compute approximations to the SVE.

We consider the Laplace transform of a function f, which is defined as the function $g(s) = \int_0^\infty e^{-st} f(t) \, dt$. This transform plays an important role in the study of dynamical systems. The *inverse Laplace transform* is then the task of computing the function f from a given function g, i.e., to solve the first-kind Fredholm integral equation

$$\int_0^\infty e^{-st} f(t) \, dt = g(s), \quad 0 \le s \le \infty. \tag{3.17}$$

This kernel in not square integrable, and hence it does not have an SVE. To see this, we note that

$$\int_0^a (e^{-st})^2 \, ds = \int_0^a e^{-2st} \, ds = \frac{1 - e^{-2ta}}{2t} \to \frac{1}{2t} \quad \text{for} \quad a \to \infty.$$

Moreover, it is known that the integral $\int_0^\infty t^{-1} \, dt$ is infinite. Hence we conclude that $\int_0^\infty \int_0^\infty (e^{-st})^2 \, ds \, dt$ is also infinite.

However, if $f(t)$ stays bounded for $t \to \infty$, then the integral in (3.17) exists for all $s \ge 0$, and we introduce only a small error by truncating the integration at some (large) number a. Moreover, the function $g(s)$ will decay for $s \to \infty$, and again we introduce only a small error by limiting s to the interval $[0, a]$. The so-obtained integral equation

$$\int_0^a e^{-st} f(t) \, dt = g(s), \quad 0 \le s \le a,$$

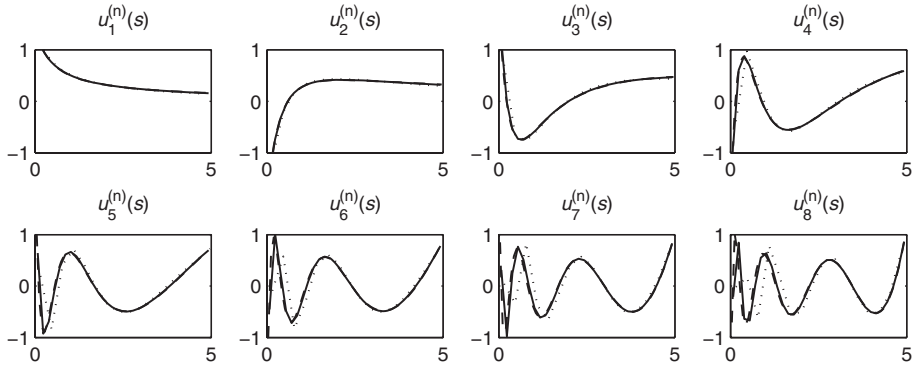

Figure 3.9. *Approximations $u_j^{(n)}$ to the left singular functions for the inverse Laplace transform restricted to the domain $[0, 5] \times [0, 5]$, such that the kernel is square integrable and the SVE exists. We use $n = 16$ (dotted lines), $n = 32$ (solid lines), and $n = 64$ (dashed lines), and the approximations converge to the singular functions.*

has a square integrable kernel (because $K(s, t) = e^{-st}$ is bounded and both integration intervals are finite).

For illustration, we discretize this integral equation with the "top hat" basis functions (3.7) with $h = a/n$, and for the integrals in (3.5) we use the approximations

$$a_{ij} = h\, K\big((i - \tfrac{1}{2})h, (j - \tfrac{1}{2})h\big).$$

Figure 3.9 shows the approximate singular functions $u_j^{(n)}$, given by (3.15), for $a = 5$ and $n = 16$, 32, and 64. It is clear that when the inverse Laplace transform is restricted to the finite domain $[0, 5] \times [0, 5]$, such that the kernel is square integrable and the SVE exists, then the approximations $u_j^{(n)}$ indeed converge to the singular functions for this kernel. The convergence is faster for small values of s.

We now consider the behavior of the approximations $u_j^{(n)}$ when we fix $n = 128$ and increase the size a of the domain $[0, a] \times [0, a]$, still using the Galerkin approach with the "top hat" basis functions. The results are shown in Figure 3.10, and we see no sign of convergence as we increase a. Of course, we cannot expect any convergence since the SVE fails to exist as $a \to \infty$, and our experiment is in accordance with this fact.

We note in passing that the test problem `i_laplace` in *Regularization Tools* uses a different discretization of the inverse Laplace transform (3.17), as described in [72]. Due to the nonexistence of the SVE for this problem, the SVD of the matrix produced by `i_laplace` does not exhibit any convergence as the order n increases.

3.5 A Closer Look at Data with White Noise

As we have illustrated above, when solving problems with real, measured data, the trouble-maker is the noise component in the right-hand side. The best way to deal with trouble-makers is to study them carefully in order to be able to limit their damage.

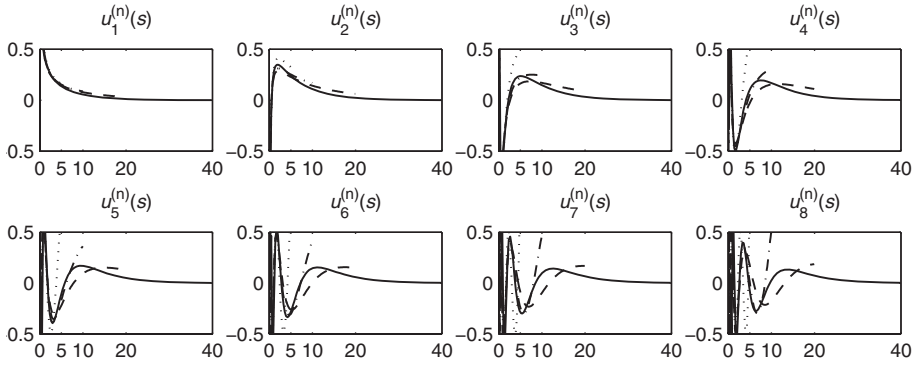

Figure 3.10. *Functions $u_j^{(n)}$ for increasing size of the domain $[0, a] \times [0, a]$. We use $a = 5$ (dotted line), $a = 10$ (dash-dotted line), $a = 20$ (dashed line), and $n = 40$ (solid line), and the functions do not show any tendency to converge.*

Therefore, at this stage it is useful to take a closer look at discretized problems with noisy data.

A few words about terminology are in order here. In mathematics and physics, the term "perturbation" is used to denote a secondary influence on a system that causes it to deviate. If we change the right-hand side in the underlying integral equation (2.2) from $g(s)$ to $\tilde{g}(s) = g(s) + \delta g(s)$, then the function $\delta g(s)$ is a perturbation that causes the solution $f(t)$ to change. In this setting we could study the effect of the perturbation, similar to our analysis in Section 2.2.

When dealing with a discretized problem $Ax = b$ and real data, the perturbation of the right-hand side b usually takes the form of "noise," i.e., more or less random variations caused by measurement errors, etc. Throughout the book, the focus is on such errors in the right-hand side—how they influence the solution, and how we can use regularization techniques to dampen their influence.

We restrict ourselves to the case where the errors are confined to the right-hand side b. This is clearly an oversimplification; often the matrix A is influenced by model errors or quadrature errors in the evaluation of its coefficients. However, these errors tend to be more systematic in nature than those in the right-hand side, and the analysis is more complicated. But our analysis here still illustrates well the basic problems with the practical solution of ill-posed problems.

Assume, therefore, that there exists an exact solution x^{exact}, and that there is a corresponding exact right-hand side given by $b^{\text{exact}} = Ax^{\text{exact}}$. Note that this requirement is not necessarily satisfied if A, b^{exact} and x^{exact} are obtained by discretization of K, g, and f, respectively, in the Fredholm integral equation (because of the quadrature and truncation errors); but it is a very convenient assumption for our analysis. Thus, our model for the right-hand side is

$$\boxed{b = b^{\text{exact}} + e, \qquad \text{where} \qquad b^{\text{exact}} = Ax^{\text{exact}}} \tag{3.18}$$

and where the vector e represents the noise in the data. This noise typically arises from measurement interference and other error sources in the recorded signal, as well as noise and inaccuracies in the measuring device.

3.5.1 Gaussian White Noise

Throughout most of this presentation, we will assume that the errors are Gaussian white noise, i.e., that all the elements of $e \in \mathbb{R}^m$ come from the same Gaussian distribution with zero mean and standard deviation η (we use the nonstandard notation η here, instead of the standard symbol σ, to avoid any confusion with the singular values). This is equivalent to saying that the covariance matrix for e is a scaled identity,

$$\mathrm{Cov}(e) \equiv \mathcal{E}(e\, e^T) = \eta^2\, I,$$

where $\mathcal{E}(\cdot)$ denotes the expected value, and η^2 is the variance. In particular, it follows that the elements e_i satisfy

$$\mathcal{E}(e_i) = 0, \qquad \mathcal{E}(e_i^2) = \eta^2, \qquad \mathcal{E}(|e_i|) = \eta\sqrt{2/\pi} \approx \eta \cdot 0.8, \tag{3.19}$$

while the noise vector $e \in \mathbb{R}^m$ satisfies

$$\mathcal{E}(e) = 0, \qquad \mathcal{E}(\|e\|_2^2) = \eta^2\, m, \qquad \mathcal{E}(\|e\|_2) = \eta\, \frac{\sqrt{2}\,\Gamma((m+1)/2)}{\Gamma(m/2)}, \tag{3.20}$$

where Γ is the gamma function. The factor $\sqrt{2}\,\Gamma((m+1)/2)/\Gamma(m/2)$ approaches $\sqrt{m}$ rapidly; for $m = 100$ the factor is 9.975. For more details, see, e.g., Chapter 18 in [47].

Throughout, the elements of b^{exact} and e are assumed to be uncorrelated. Consequently we have $\mathcal{E}(b) = \mathcal{E}(b^{\text{exact}})$, and it follows that the covariance for the right-hand side b is

$$\mathrm{Cov}(b) \equiv \mathcal{E}\Big((b - \mathcal{E}(b))\,(b - \mathcal{E}(b))^T\Big) = \mathcal{E}(e\, e^T) = \eta^2\, I.$$

Then it follows from standard results in multivariate statistics that the covariance matrix for the "naive" solution

$$x = A^{-1}b = A^{-1}(b^{\text{exact}} + e) = x^{\text{exact}} + A^{-1}e$$

is given by

$$\mathrm{Cov}(x) = A^{-1}\mathrm{Cov}(b)\, A^{-T} = \eta^2\, (A^T A)^{-1}.$$

The norm of this matrix is

$$\|\mathrm{Cov}(x)\|_2 = \eta^2 \,/\, \sigma_n^2,$$

in which σ_n is the smallest singular value of A. Clearly, if A is very ill conditioned, $\mathrm{cond}(A) \gg 1$, then this covariance matrix is likely to have very large elements, indicating that $x = A^{-1}b$ is very sensitive to data errors.

Now let us consider the influence of the noise on the SVD coefficients, i.e., the elements of the vector $U^T b = U^T b^{\text{exact}} + U^T e$ which are given by

$$u_i^T b = u_i^T (b^{\text{exact}} + e) = u_i^T b^{\text{exact}} + u_i^T e, \qquad i = 1, \ldots, n.$$

Using again elementary relations from statistics, we obtain

$$\mathrm{Cov}(U^T e) = U^T \mathrm{Cov}(e)\, U = \eta^2 U^T U = \eta^2 I,$$

showing that the transformed noise vector $U^T e$ also behaves as Gaussian white noise, and therefore all the results in (3.19)–(3.20) carry over to $U^T e$.

We assume now that the exact right-hand side b^{exact} satisfies the discrete Picard condition; i.e., the coefficients $|u_i^T b^{\text{exact}}|$ decay faster than the singular values. We also make the reasonable assumption that the norm of the perturbation e is smaller than that of b^{exact} (otherwise all information from b^{exact} is lost in b).

From these assumptions it follows that some of the coefficients $u_i^T b^{\text{exact}}$ must be larger in magnitude than the "noise coefficients" $u_i^T e$, while others may be smaller. In other words, the noisy coefficients satisfy

$$\boxed{u_i^T b = u_i^T b^{\text{exact}} + u_i^T e \approx \begin{cases} u_i^T b^{\text{exact}}, & |u_i^T b^{\text{exact}}| > |u_i^T e|, \\ u_i^T e, & |u_i^T b^{\text{exact}}| < |u_i^T e|. \end{cases}} \tag{3.21}$$

In other words, there are two different kinds of right-hand side coefficients:

- There are "reliable" coefficients, characterized by $|u_i^T b| \gg \eta$, that mainly carry information about the exact data b^{exact}. Due to the discrete Picard condition, these coefficients correspond to the larger singular values σ_i.
- There are "noisy" SVD components, namely, those for which $|u_i^T b^{\text{exact}}|$ is smaller than η and for which, therefore, $|u_i^T b| \approx \eta$. These coefficients correspond to the smaller singular values σ_i, also due to the discrete Picard condition.

We have thus explained the overall behavior of the Picard plot, as observed in the figures shown in this chapter.

3.5.2 Uniformly Distributed White Noise

We emphasize that a white-noise vector e is not required to have elements from a Gaussian distribution; the requirement is that $\mathrm{Cov}(e) = \eta^2 I$. This is also true if all the elements e_i of e are uncorrelated and from the same uniform distribution in the interval $[-\sqrt{3}\eta, \sqrt{3}\eta]$. Then

$$\mathcal{E}(e_i) = 0, \qquad \mathcal{E}(e_i^2) = \eta^2, \qquad \mathcal{E}(|e_i|) = \tfrac{\sqrt{3}}{2}\,\eta,$$

and the noise vector satisfies

$$\mathcal{E}(e) = 0, \qquad \mathcal{E}(\|e\|_2^2) = m\,\eta^2, \qquad \mathcal{E}(\|e\|_2) = \Upsilon_m \sqrt{m}\,\eta.$$

The factor Υ_m satisfies $\frac{\sqrt{3}}{2} < \Upsilon_m < 1$, and it approaches 1 rapidly as m increases; cf. [2]. Uniformly distributed white noise arises, e.g., in the form of quantization noise when a measurement is represented by a finite number of digits; for example, if a signal is represented by an integer, then the representation errors are uniformly distributed in the interval $[-0.5, 0.5]$.

3.6 Noise that Is Not White

White noise can be considered a mathematical abstraction that keeps our analysis (and our algorithms) simple. But this is not a fair statement; in many applications the noise is such that, to a good approximation, it can be considered white.

We have already encountered one definition of a white-noise vector, namely, that the covariance matrix is a scaled identity matrix. Another definition, more closely related to signal processing, is that all frequencies present in the time series signal e must have the same probability. The name "white noise" derives from the analogy with "white light," in which all colors (i.e., frequencies of light) are present.

To state the above more rigorously, we need the $m \times m$ discrete Fourier matrix F (which is complex and symmetric) defined such that the vector e and its discrete Fourier transform $\hat{e}$ are related by

$$\hat{e} = \texttt{fft}(e) = \text{conj}(F)\, e, \qquad e = \texttt{ifft}(\hat{e}) = m^{-1} F\, \hat{e}$$

and satisfying $\text{conj}(F)\, F = m\, I$. Then the vector e is white noise if the covariance matrix for its discrete Fourier transform is a scaled identity. The two definitions are obviously equivalent, because

$$\text{Cov}(\hat{e}) = \text{Cov}\left(\text{conj}(F)\, e\right)) = \text{conj}(F)\, \text{Cov}(e)\, F = m \eta^2\, I,$$

but the requirement to $\hat{e}$ is sometimes more practical. For example, we can say that a signal is "white-noise–like" if $\text{Cov}(\hat{e})$ is close to a scaled identity matrix, and hence its spectral components are nearly uncorrelated and nearly have the same variance. (But note that $\text{Cov}(\hat{e})$ does not imply that that $\text{Cov}(e)$ is close to a scaled identity!)

3.6.1 Signal-Correlated Noise

To illustrate the usefulness of the above definition, we consider noise whose amplitude is proportional to the pure data. Specifically we consider the case where the noisy right-hand side is given by

$$b = b^{\text{exact}} + e_b \qquad \text{with} \qquad e_b = \text{diag}(b^{\text{exact}})\, e, \tag{3.22}$$

where e is a white-noise vector. In other words, the noise vector e_b has elements given by $(e_b)_i = b_i^{\text{exact}} \cdot e_i$ for $i = 1, \ldots, m$, and its covariance matrix is

$$\text{Cov}(e_b) = \text{diag}(b^{\text{exact}})\, \text{Cov}(e)\, \text{diag}(b^{\text{exact}}) = \eta^2\, \text{diag}(b^{\text{exact}})^2.$$

We will now justify that this type of noise is "white-noise–like." That is, we must show that the covariance matrix for the discrete Fourier transform $\hat{e}_b = \text{conj}(F)\, e_b$,

$$\text{Cov}(\hat{e}_b) = \eta^2\, \text{conj}(F)\, \text{diag}(b^{\text{exact}})^2\, F,$$

is close to a scaled identity matrix.

A standard result for discrete convolution (see, e.g., [35]) says that a matrix of the form $\text{conj}(F)\, \text{diag}(d)\, F$ is circulant and its first column is given by $\hat{d} = \text{conj}(F)\, d$.

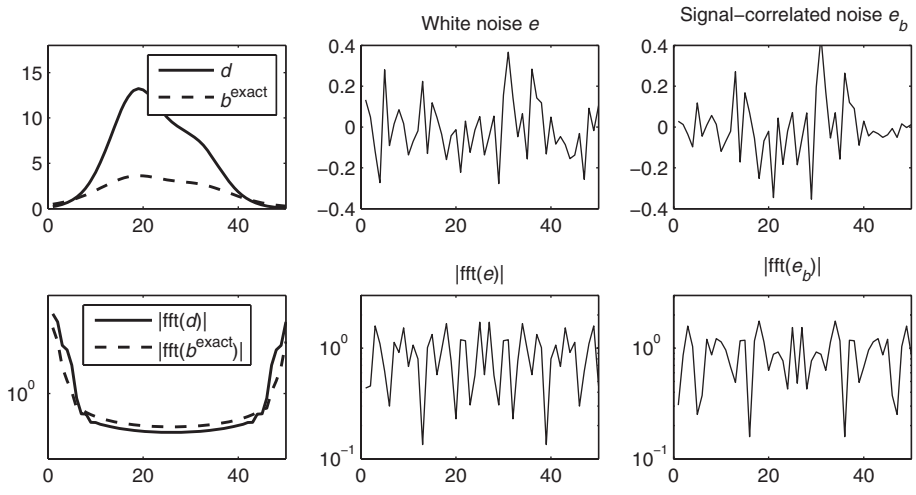

Figure 3.11. *When the noise amplitude is proportional to the signal; case study with* `shaw` *test problem. Top: The vectors* b^{exact}, $d = ((b^{\text{exact}})_1^2, \ldots, (b^{\text{exact}})_m^2)^T$, e, *and* e_b. *Bottom: Fourier spectra for* b^{exact}, d, e, *and* e_b.

Hence, by identifying $d = \left((b^{\text{exact}})_1^2, \ldots, (b^{\text{exact}})_m^2\right)^T$ we conclude that $\text{Cov}(\hat{e}_b)$ is circulant and its first column is

$$\left(\text{Cov}(\hat{e}_b)\right)_{:,1} = \eta^2 \,\text{conj}(F)\left((b^{\text{exact}})_1^2, \ldots, (b^{\text{exact}})_m^2\right)^T = \eta^2\,\hat{d}.$$

That is, the first column is the discrete Fourier transform of the vector d whose elements are the squares of those of the exact right-hand side b^{exact}. Like b^{exact}, the vector d is also dominated by low-frequency components, and hence the largest elements of $\hat{d}$ are located at the top. Hence, $\text{Cov}(\hat{e}_b)$ has large identical elements along its main diagonal, and they decay in magnitude away from the diagonal. Consequently we can say that e_b is "white-noise–like."

Figures 3.11 and 3.12 illustrate the above analysis for the test problem `shaw` from *Regularization Tools*. In particular, note that both b^{exact} and d are dominated by low-frequency components, and that the spectra for e_b resembles that of e; i.e., it is flat, and therefore e_b is "white-noise–like."

3.6.2 Poisson Noise

Poisson noise often arises in connection with data that consist of counts of some quantity, such as light intensity which is proportional to the number of photons hitting the measurement device. We note that such data always consist of nonnegative integers.

To motive our notation for Poisson noise, consider the following alternative way to write data with additive Gaussian white noise with standard deviation η:

$$b_i \sim \mathcal{N}(b_i^{\text{exact}}, \eta^2), \qquad i = 1, \ldots, m.$$

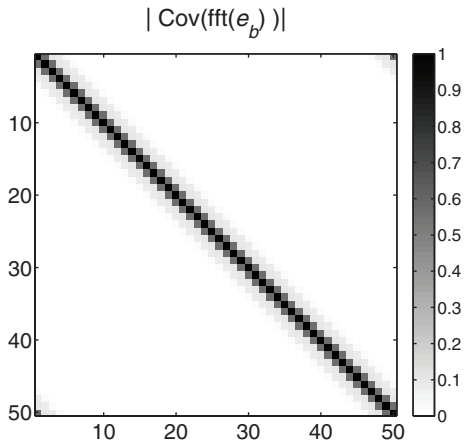

Figure 3.12. *The covariance matrix* $\mathrm{Cov}(\hat{e}_b)$ *for the Fourier transform of the noise vector* e_b *is close to a scaled identity matrix.*

That is, the ith data value b_i has mean value $\mathcal{E}(b_i) = b_i^{\text{exact}}$ and variance $\mathcal{V}(b_i) = \eta^2$. Similarly, the data considered in (3.22) can be written as

$$b_i \sim \mathcal{N}\left(b_i^{\text{exact}}, (\eta b_i^{\text{exact}})^2\right), \qquad i = 1, \ldots, m,$$

which implies that $\mathcal{E}(b_i) = b_i^{\text{exact}}$ and $\mathcal{V}(b_i) = (\eta b_i^{\text{exact}})^2$; i.e., the standard deviation is proportional to $|b_i^{\text{exact}}|$ with proportionality factor η.

For *Poisson noise*, we recall that the Poisson distribution depends on a single parameter which is both the mean and the variance. Hence, data that follow a Poisson distribution[1] can be written as

$$b_i \sim \mathcal{P}(b_i^{\text{exact}}), \qquad i = 1, \ldots, m.$$

Thus, the mean and the variance of the ith data value b_i are now given by $\mathcal{E}(b_i) = b_i^{\text{exact}}$ and $\mathcal{V}(b_i) = b_i^{\text{exact}}$. This is clearly another example of noise that is correlated with the signal.

While there is apparently no proportionality factor in the expression for the variance of Poisson noise, we can artificially introduce such a factor by scaling our exact data. Specifically, if we let $\bar{b}_i^{\text{exact}} = \eta\, b_i^{\text{exact}}$ for some positive η and define the Poisson data $\bar{b}_i = \mathcal{P}(\bar{b}_i^{\text{exact}}) = \mathcal{P}(\eta\, b_i^{\text{exact}})$, then clearly $\mathcal{E}(\bar{b}_i) = \mathcal{V}(\bar{b}_i) = \eta\, b_i^{\text{exact}}$. Hence, if we are willing to accept noninteger data, then we can rescale $\bar{b}_i^{\text{exact}}$ with η^{-1} to obtain the data

$$b_i \sim \mathcal{P}(\eta\, b_i^{\text{exact}})/\eta, \qquad i = 1, \ldots, m,$$

which has mean $\mathcal{E}(b_i) = b_i^{\text{exact}}$ and variance $\mathcal{V}(b_i) = \eta\, b_i^{\text{exact}}/\eta^2 = b_i^{\text{exact}}/\eta$. We can use this mechanism to generate noninteger Poisson-like data in which we can control the proportionality factor between the exact data and the noise.

[1] We recall that when b_i^{exact} is large, then $\mathcal{P}(b_i^{\text{exact}}) \approx \mathcal{N}(b_i^{\text{exact}}, b_i^{\text{exact}})$.

3.6.3 Broad-Band Colored Noise

We conclude this section with a brief discussion of noise that is neither white nor "white-noise–like." Of special interest is noise that is still broad-band (i.e., all frequencies are represented) but dominated by either low-frequency (LF) components or high-frequency (HF) components. Both types of colored noise arise in applications, where they are usually referred to as broad-band colored noise; for brevity we shall refer to them as "LF noise" and "HF noise," respectively.

There are many ways to generate colored noise from a white-noise signal, and we describe just one simple method here. Define the symmetric tridiagonal matrices

$$\Psi_{\mathrm{LP}} = \mathrm{tridiag}(1, 2, 1), \qquad \Psi_{\mathrm{HF}} = \mathrm{tridiag}(-1, 2, -1). \tag{3.23}$$

For example, for $n = 4$ we have

$$\Psi_{\mathrm{LF}} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}, \qquad \Psi_{\mathrm{HF}} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & -2 \end{pmatrix}.$$

Then we can generate colored noise by multiplying these matrices to a white-noise signal:

$$e_{\mathrm{LF}} = \Psi_{\mathrm{LF}}\, e, \qquad e_{\mathrm{HF}} = \Psi_{\mathrm{HF}}\, e \tag{3.24}$$

(in the language of signal processing, we apply a low-pass filter and, respectively, a high-pass filter to a white-noise signal to generate a colored-noise signal).

To study the covariance matrices for the colored-noise vectors, we need the following result for the eigenvalue decompositions of the two matrices Ψ_{LF} and Ψ_{HF} (which follows directly from Example 7.4 in [22], which gives closed-form expressions for the eigenvalues and vectors of tridiagonal matrices):

$$\Psi_{\mathrm{LF}} = S\,\mathrm{diag}(d_1, \ldots, d_m)\,S^T, \qquad \Psi_{\mathrm{HF}} = S\,\mathrm{diag}(d_m, \ldots, d_1)\,S^T,$$

in which the diagonal elements $0 < d_j < 4$ are given by

$$d_j = 2 + 2\cos\bigl(\pi j/(m+1)\bigr), \qquad j = 1, \ldots, m,$$

and S is an orthogonal matrix with elements

$$s_{ij} = \sqrt{2/(m+1)}\,\sin\bigl(\pi ij/(m+1)\bigr), \qquad i, j = 1, \ldots, m.$$

The columns of S are samples of sine functions with increasing frequency, and hence they provide a spectral basis very similar to the Fourier basis. We conclude that the covariance matrices for the colored-noise vectors are given by

$$\mathrm{Cov}(e_{\mathrm{LF}}) = \eta^2\, S\,\mathrm{diag}(d_1^2, \ldots, d_m^2)\,S^T, \tag{3.25}$$

$$\mathrm{Cov}(e_{\mathrm{HF}}) = \eta^2\, S\,\mathrm{diag}(d_m^2, \ldots, d_1^2)\,S^T, \tag{3.26}$$

where η is the standard deviation of the white noise in e in (3.24). Figure 3.13 shows plots of the diagonal elements in these expressions, and we clearly see that the

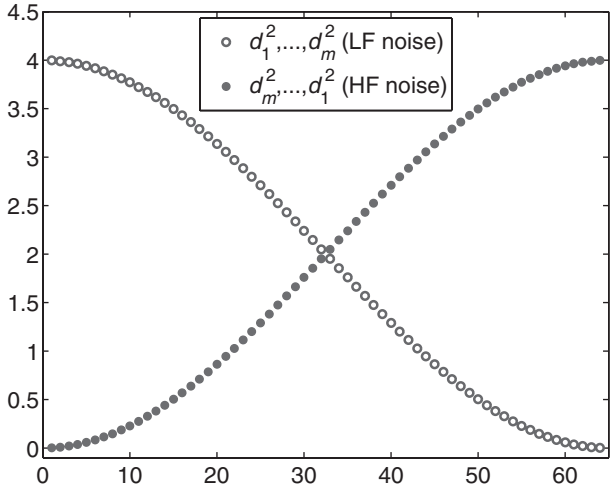

Figure 3.13. *The elements of the diagonal matrices in the expressions* (3.25) *and* (3.26) *for the covariance matrices for colored noise.*

different frequencies, represented by the different columns of S, are represented in very different ways in the vectors e_{LF} and in e_{HF}. Specifically, e_{LF} consists mainly of low-frequency components, while e_{HF} consists mainly of high-frequency components. In Section 4.8 we return to colored noise and its influence on the solutions to ill-posed problems.

3.7 The Story So Far

This chapter deals with the square or overdetermined linear systems of equations $Ax = b$ and $\min_x \|Ax - b\|_2$ that arise in the numerical treatment of inverse problems. We showed how such systems are obtained by simple quadrature and expansion discretization methods applied to the integral equation, and we showed how the singular value decomposition (SVD), given by (3.8)—in analogy with the SVE—is a powerful mathematical and computational tool for analysis of the discretized inverse problem.

In particular, we showed how the properties of the SVE carry over to the SVD and how this can be used to numerically compute the SVE. Motivated by the SVD-SVE relationship, we defined the discrete Picard condition (3.16) for the existence of a meaningful solution to the discretized problem. We also use an example to discuss how the SVD may, or may not, converge to the SVE.

Finally, we discussed the unavoidable noise in the data that constitutes the right-hand side, and we discussed some important noise models for white and nonwhite noise: Gaussian white noise, uniformly distributed white noise, signal-correlated noise, Poisson noise, and broad-band colored noise.

Exercises

3.1. **Convergence of the SVE Approximations**
This exercise illustrates the convergence properties of the approximate SVE quantities, obtained from the discretization of the second derivative test problem in Exercise 2.3, for which the SVE is known analytically.

The test problem `deriv2` in *Regularization Tools* uses the Galerkin discretization method with the orthonormal "top-hat" ⊓⊔ basis functions $\chi_i(t)$ in (3.7). Compute the SVD of the coefficient matrix A for increasing values of n; for example, use $n = 8$, 16, 32, 64, etc. Compare the computed singular values with the exact ones. Can you verify that (3.13) and (3.14) are satisfied? In your experiments, how does the error $\mu_i - \sigma_i^{(n)}$ decay with n?

Next, plot some of the exact left singular functions $u_j(s)$ together with the approximations $u_j^{(n)}(s)$, say, for $j = 1, 2, 3, 4$. Use the commands

```
h = 1/n; stairs((0:n)*h,[U(:,j);U(n,j)]/sqrt(h))
```

to plot the approximate singular function $u_j^{(n)}(s)$ as a piecewise function. You may have to change the sign of this function; cf. the "nitty-gritty details" in Section 3.2.3. Discuss the behavior of the approximations as n increases.

3.2. **Galerkin Discretization of Gravity Problem**
As an example of the use of the Galerkin discretization method, discretize the gravity surveying problem from Section 2.1 with the "top hat" functions $\phi_j(t) = \chi_j(t)$ in (3.7) as the basis functions, and the functions $\psi_i(s) = \delta(s - s_i)$ with $s_i = (i - \frac{1}{2})h$ (corresponding to sampling the right-hand side at the s_i). Derive an expression for the matrix elements a_{ij} based on (3.5).

3.3. **Derivation of Important SVD Expressions**
Give the details of the derivation of the important equations (3.9) and (3.10), as well as the expression (3.11) for the naive solution.

The solution x_{LS} to the linear least squares problem $\min_x \|Ax - b\|_2$ is formally given by $x_{\mathrm{LS}} = (A^T A)^{-1} A^T b$, under the assumption that A has more rows than columns and $A^T A$ has full rank. Use this expression together with the SVD to show that x_{LS} has the *same* SVD expansion (3.11) as the naive solution.

3.4. **SVD Analysis of the Degenerate Kernel Problem**
This exercise illustrates the use of the SVD analysis technique, applied to the problem with a degenerate kernel from Exercise 2.2.

The midpoint quadrature rule plus collocation in the quadrature abscissas lead to a discretized problem with a matrix A whose elements are given by

$$a_{ij} = h\left((i + 2j - \tfrac{3}{2})h - 3\right), \qquad i, j = 1, \ldots, n,$$

with $h = 2/n$. Show that the quadrature abscissas are $t_j = -1 + (j - \frac{1}{2})h$ for $j = 1, \ldots, n$, and verify the above equation for a_{ij}.

Show that the columns of A are related by

$$a_{i,j+1} + a_{i,j-1} = 2\, a_{i,j}, \qquad i = 1, \ldots, n, \quad j = 2, \ldots, n-1,$$

and, consequently, that the rank of A is 2 for all $n \geq 2$. Verify this experimentally by computing the SVD of A for different values of n.

Since, for this matrix, $A = u_1 \sigma_1 v_1^T + u_2 \sigma_2 v_2^T$, it follows that Ax is always a linear combination of u_1 and u_2 for any x. Show this. Then justify (e.g., by plotting the singular vectors u_1 and u_2) that linear combinations of these vectors represent samples of a linear function, in accordance with the results from Exercise 2.2.

3.5. **SVD Analysis of the Gravity Surveying Problem**
The purpose of this exercise is to illustrate how a simple inverse problem is discretized by means of the midpoint quadrature rule. The exercise also illustrates that the coefficient matrix for the resulting system of linear equations is very ill conditioned and that the solution is highly sensitive to errors.

We use the geophysical model problem from Section 2.1. For a problem with n data points and also n unknowns, and using $s_i = t_i$ for the collocation points, derive the following formula for the matrix elements:

$$a_{ij} = \frac{d}{n} \left(d^2 + \big((i-j)/n\big)^2 \right)^{-3/2}, \qquad i, j = 1, \ldots, n,$$

and check that the matrix A is symmetric. This is implemented in the function `gravity` in *Regularization Tools*.

Then use the matrix A to compute right-hand sides $b = Ax$ for two different exact solutions x—one corresponding to a smooth solution (constant and linear functions are not allowed), and one corresponding to a solution with one or more jump discontinuities. Make up your own x vectors. Notice that the right-hand side is always smooth. Study experimentally how the right-hand side varies with the depth d.

Compute the condition number of A, and examine how it varies with n for a fixed value of the depth d. Then keep n fixed, and study how the condition number varies with d; try to explain the observed behavior.

Finally, try to solve the problem by computing the "naive" solution $x = A^{-1}b$ for $n = 32$. First solve the problem with a noise-free right-hand side; then try again with right-hand sides in which a very small amount of Gaussian white noise has been added. How large can you make the noise (as measured by $\|e\|_2$) before the inverted noise starts to dominate the computed solution?

3.6. **SVD Analysis of a One-Dimensional Image Reconstruction Problem**
The purpose of this exercise is to illustrate how the SVD can be used to analyze the smoothing effects of a first-kind Fredholm integral equation. We use the one-dimensional reconstruction test problem, which is implemented in *Regularization Tools* as function `shaw`. The kernel in this problem is given by

$$K(s,t) = \Big(\cos(s) + \cos(t)\Big)^2 \left(\frac{\sin\big(\,\pi(\sin(s) + \sin(t))\,\big)}{\pi(\sin(s) + \sin(t))} \right)^2,$$

$-\pi/2 \leq s, t \leq \pi/2$, while the solution is

$$f(t) = 2\,\exp\big(-6(t-0.8)^2\big) + \exp\big(-2(t+0.5)^2\big).$$

This integral equation models a situation where light passes through an infinitely long slit, and the function $f(t)$ is the incoming light intensity as a function of the incidence angle t. The problem is discretized by means of the midpoint quadrature rule to produce A and x^{exact}, after which the exact right-hand side is computed as $b^{\text{exact}} = A\,x^{\text{exact}}$. The elements of b^{exact} represent the outgoing light intensity on the other side of the slit.

Choose $n = 24$ and generate the problem. Then compute the SVD of A, and plot and inspect the left and right singular vectors. What can be said about the number of sign changes in these vectors?

Use the function `picard` from *Regularization Tools* to inspect the singular values σ_i and the SVD coefficients $u_i^T b^{\text{exact}}$ of the exact solution b^{exact}, as well as the corresponding solution coefficients $u_i^T b^{\text{exact}}/\sigma_i$. Is the Picard condition satisfied?

Add a very small amount of noise e to the right-hand side b^{exact}, i.e., $b = b^{\text{exact}} + e$, with $\|e\|_2/\|b^{\text{exact}}\|_2 = 10^{-10}$. Inspect the singular values and SVD coefficients again. What happens to the SVD coefficients $u_i^T b$ corresponding to the small singular values?

Prelude to the next chapter: Recall that the undesired "naive" solution $x = A^{-1}b$ can be written in terms of the SVD as (3.11). Compute the partial sums

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i}\, v_i$$

for $k = 1, 2, \ldots$, and inspect the vectors x_k. Try to explain the behavior of these vectors.

3.7. **Making Noise**

Choose $m = 40$ and $\eta = 10^{-5}$ and generate a number of instances of white Gaussian noise with standard deviation η, by means of `e = eta*randn(m,1)`. Check that the computed values of $\mathcal{E}(e)$, $\mathcal{E}(\|e\|_2^2)$, and $\mathcal{E}(\|e\|_2)$ are in accordance with the results in (3.20).

Now use the function `shaw` from *Regularization Tools* to generate the test problem from the previous exercise with exact right-hand side b^{exact}. Add the above white-noise vector e (with $\eta = 10^{-5}$) to obtain the noisy right-hand side $b = b^{\text{exact}} + e$. What is the relative noise level $\|e\|_2/\|b^{\text{exact}}\|_2$ for this problem?

Use a semilogarithmic plot to show absolute values of the elements of $U^T b^{\text{exact}}$ and $U^T e$, and explain the behavior of these plots. In particular, explain why both graphs tend to level off in the right part of the plot, and explain the magnitude of the plateau.

Finally, show that if you want to generate a noisy right-hand side with a given relative noise level $\|e\|_2/\|b^{\text{exact}}\|_2 =$ `rnl`, then you should use the MATLAB statements

```
e = randn(m,1);
e = e/norm(e);
e = rnl*norm(bexact)*e;
b = bexact + e;
```

What is the corresponding value of the standard deviation η for the noise vector, expressed in terms of m and `rnl`?

3.8. **Symmetry and Persymmetry***
There are 14 test problems available in *Regularization Tools*. Which of these produce a symmetric coefficient matrix, and which produce a persymmetric matrix (see Section 3.2.2)?

3.9. **An Ambiguity Problem in Potential Theory***
This exercise illustrates a null-space ambiguity and how we can use the SVD to analyze this phenomenon. The model problem that we use here comes from potential theory, with a kernel given by

$$K(s,t) = \ln|s-t|, \qquad s,t \in [0,\ell].$$

For a certain critical length ℓ^* of the interval, the integral operator has a one-dimensional null-space.

If we discretize this problem at n equidistant points on the line segment, then the corresponding matrix A has entries for $i,j = 1,\ldots,n$ given by

$$a_{ij} = (t_{j+1} - s_i)\,\ln(|t_{j+1} - s_i|) - (t_j - s_i)\,\ln(|t_j - s_i|) - (t_{j+1} - t_j),$$

where

$$s_i = (i - 0.5)\,\ell/n, \qquad i = 1,\ldots,n,$$

and

$$t_j = (j-1)\,\ell/n, \qquad j = 1,\ldots,n+1.$$

Note that since we are merely interested in solutions to the homogeneous problem $Ax = 0$, we do not need a particular right-hand side. All that is needed is the SVD of the matrix A; recall that $Av_i = 0$ when $\sigma_i = 0$.

Write a MATLAB script or function that computes the matrix A, given values of n and ℓ, using the above equations. For a range of lengths ℓ, compute the singular values of A and find the critical length ℓ^* for which the homogeneous problem has a solution. The critical length should be slightly larger than 4.

At the critical length, compute a solution to the homogeneous problem, using the SVD of the matrix A.

Chapter 4

Computational Aspects: Regularization Methods

After the careful analysis that we carried out in the previous two chapters, we now have a good understanding of the difficulties associated with discrete ill-posed problems and why the "naive" solution can be expected to be useless. In this chapter we use this insight to devise several different methods for computing approximate solutions that are less sensitive to perturbations than the naive solution.

These methods are called regularization methods because they enforce regularity on the computed solution—typically in the form of a requirement that the solution is smooth, in some sense. And by enforcing this regularity, or smoothness, we suppress some of the unwanted noise components, leading to more stable approximate solutions.

Almost all the regularization methods treated in this book produce solutions which can be expressed as a filtered SVD expansion of the form

$$x_{\text{reg}} = \sum_{i=1}^{n} \varphi_i \, \frac{u_i^T b}{\sigma_i} \, v_i, \tag{4.1}$$

where φ_i are the *filter factors* associated with the method. These methods are commonly referred to as *spectral filtering methods* because, as we saw in Section 2.5, the SVD basis can be considered as a spectral basis. This chapter introduces three important spectral filtering methods: truncated SVD, selective SVD, and Tikhonov's method.

We recall that the coefficient matrix A is allowed to have more rows than columns, i.e.,

$$A \in \mathbb{R}^{m \times n} \qquad \text{with} \qquad m \geq n,$$

and for $m > n$ it is natural to consider the least squares problem $\min_x \|Ax - b\|_2$. Hence, when we use the phrase "naive solution" we mean either the solution $A^{-1}b$ (when $m = n$) or the least squares solution (when $m > n$), and we emphasize the convenient fact that the naive solution has precisely the same SVD expansion (3.11) in both cases. The expression for the condition number of A is also the same, namely, $\text{cond}(A) = \sigma_1/\sigma_n$. Hence, there is really no need to distinguish between the cases $m = n$ and $m > n$ throughout.

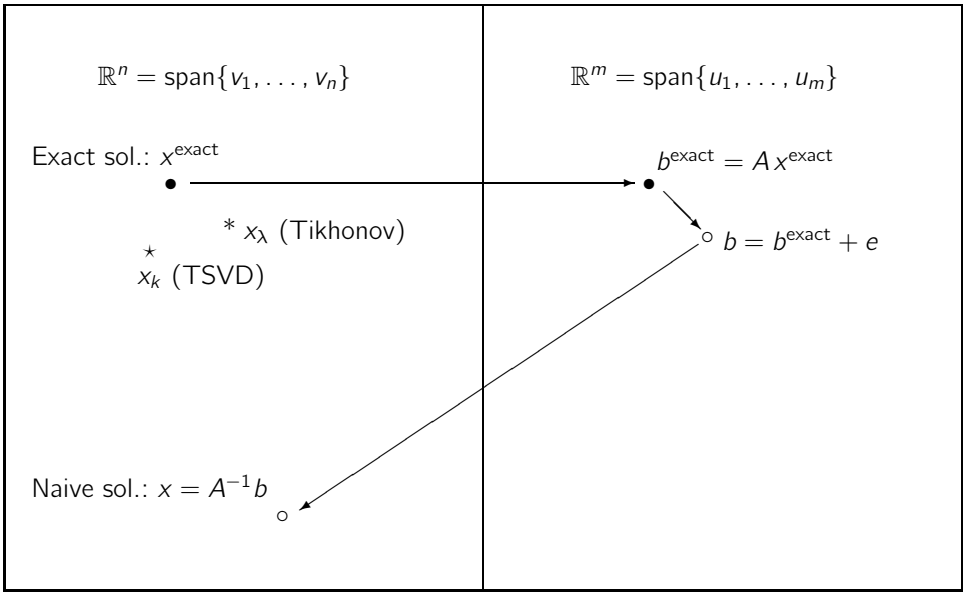

Figure 4.1. *Illustration of the need for regularization.*

4.1 The Need for Regularization

As we have already seen in the previous chapter, discrete ill-posed problems are characterized by having coefficient matrices with a very large condition number. This implies that the naive solution is very sensitive to any perturbation of the right-hand side, representing the errors in the data. Specifically, assume that the exact and perturbed solutions x^{exact} and x satisfy

$$A\,x^{\text{exact}} = b^{\text{exact}}, \qquad A\,x = b = b^{\text{exact}} + e,$$

where e denotes the perturbation. Then classical perturbation theory leads to the bound

$$\frac{\|x^{\text{exact}} - x\|_2}{\|x^{\text{exact}}\|_2} \leq \text{cond}(A)\,\frac{\|e\|_2}{\|b^{\text{exact}}\|_2}.$$

Since $\text{cond}(A)$ is large, this implies that x can be very far from x^{exact}. Moreover, although the above expression is an upper bound only, every experience shows that perturbations close to the upper bound always arise in applications of discrete inverse problems. Hence we need regularization methods that can compute less sensitive approximations to x^{exact}.

Figure 4.1 illustrates the need for regularization. Here, the exact solution x^{exact} gives rise to an exact right-hand side b^{exact}, while the perturbed right-hand side is $b = b^{\text{exact}} + e$. Due to the ill conditioning of A, the naive solution $x = A^{-1}b$ can be expected to be far from the exact solution x^{exact}, even when the perturbation is small, i.e., when $\|e\|_2 \ll \|b^{\text{exact}}\|_2$. Regularized solutions, such as the truncated SVD

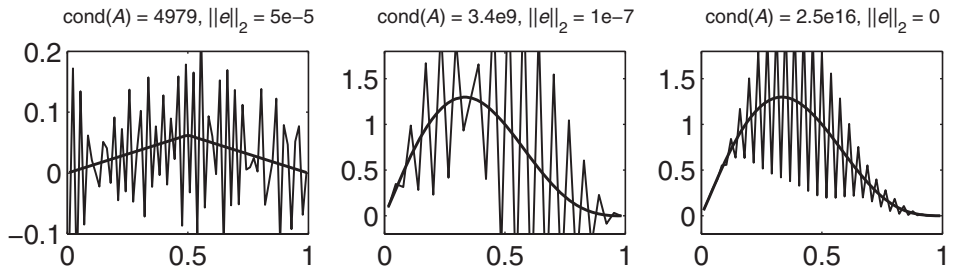

Figure 4.2. *Exact solutions (smooth lines) together with the naive solutions (jagged lines) to two test problems. Left:* `deriv2` *with* $n = 64$. *Middle and right:* `gravity` *with* $n = 32$ *and* $n = 53$. *Due to the large condition numbers (especially for* `gravity`*), the small perturbations lead to useless naive solutions.*

(TSVD) solution x_k and the Tikhonov solution x_λ, to be introduced in the next two sections, are (we hope) good approximations to x^{exact}.

To further illustrate this aspect we consider two test problems, namely, the second derivative problem from Exercise 2.3 (implemented in `deriv2`) of size $n = 64$, and the gravity surveying test problem from Exercise 3.5 (implemented in `gravity`) with $n = 32$ and $n = 53$. Figure 4.2 shows the exact solutions to these problems, together with the naive solutions computed for small perturbations of the right-hand side.

For `deriv2` the perturbation is scaled such that $\|e\|_2 = 5 \cdot 10^{-5}$, and with this noise level we cannot find any information about x^{exact} in the naive solution. For `gravity` with $n = 32$ the noise level is much smaller, $\|e\|_2 = 10^{-7}$, but due to the much larger condition number for this problem such a small perturbation is still large enough to produce a useless naive solution. If we increase the problem size to $n = 53$, then the matrix is now so ill conditioned that even with $e = 0$ the rounding errors during the computation of the naive solution render this solution useless.

By means of the regularization methods that we develop in this chapter, as well as in Chapters 6 and 8, we are able to compute approximations to x^{exact} that are much less sensitive to the perturbations of the right-hand side. In particular, we return to regularized solutions to the gravity surveying problem in Sections 4.2 and 4.4 below.

4.2 Truncated SVD

From the analysis in the two previous chapters, it is clear that the extremely large errors in the naive solution come from the noisy SVD components associated with the smaller singular values. In particular, the results in (3.21) show that the naive solution is dominated by SVD coefficients of the form $u_i^T b/\sigma_i \approx u_i^T e/\sigma_i$ (where e is the perturbation of b) corresponding to the smaller singular values; see Figure 3.7.

The good news is that our analysis also reveals that some of the SVD coefficients are rather trustworthy, namely, the coefficients of the form $u_i^T b/\sigma_i \approx u_i^T b^{\text{exact}}/\sigma_i$ (in which $b^{\text{exact}} = A\,x^{\text{exact}}$ is the exact right-hand side) corresponding to the larger singular

values; again, see Figure 3.7. This gives us hope that we can actually recover some of the information about the solution to the problem.

Another piece of good news is that we can assume that the exact right-hand side satisfies the discrete Picard condition (otherwise there is no point in trying to solve the discrete ill-posed problem). As a consequence, the SVD components of the exact solution with largest magnitude are precisely those coefficients that are approximated reasonably well, because for the small indices i we have $u_i^T b/\sigma_i \approx u_i^T b^{\text{exact}}/\sigma_i = v_i^T x^{\text{exact}}$.

These considerations immediately lead to a "brute force" method for computing regularized approximate solutions: simply chop off those SVD components that are dominated by the noise. Hence, we define the *truncated SVD* (TSVD) solution x_k as the solution obtained by retaining the first k components of the naive solution:

$$\boxed{x_k \equiv \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i.} \tag{4.2}$$

The *truncation parameter* k should be chosen such that all the noise-dominated SVD coefficients are discarded. A suitable value of k often can be found from an inspection of the Picard plot; for example, for the two noise problems in Figure 3.7 we would choose $k = 16$ and $k = 9$, respectively.

There is an alternative formulation of the TSVD method which is important. While the definition of x_k in (4.2) takes its basis in a specific formula for computing the regularized solution, we can also define a regularized solution as the solution to a modified and better conditioned problem. Let us introduce the TSVD matrix A_k, which is the rank-k matrix defined as

$$A_k = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{pmatrix} \begin{pmatrix} | & & | \\ v_1 & \cdots & v_k \\ | & & | \end{pmatrix}^T = \sum_{i=1}^{k} u_i \sigma_i v_i^T. \tag{4.3}$$

It can be shown that the condition number of this matrix is $\text{cond}(A_k) = \sigma_1/\sigma_k$, and it is typically much smaller than the condition number $\text{cond}(A) = \sigma_1/\sigma_n$ of the original matrix A.

Hence, it seems like a good idea to replace the original and ill-conditioned problem $Ax = b$ or $\min \|Ax - b\|_2$ with the better conditioned least squares problem $\min \|A_k x - b\|_2$. The least squares formulation is needed, because we can no longer expect to find an x such that $A_k x = b$. However, since A_k is rank deficient (as long as $k < n$), there is not a unique solution to this least squares problem; it is easy to show that the general solution has the form

$$x = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i + \sum_{i=k+1}^{n} \zeta_i v_i, \qquad \zeta_i = \text{arbitrary}. \tag{4.4}$$

To define a unique solution, we must supplement the least squares problem with an additional constraint on the solution x. A natural constraint, in many applications, is

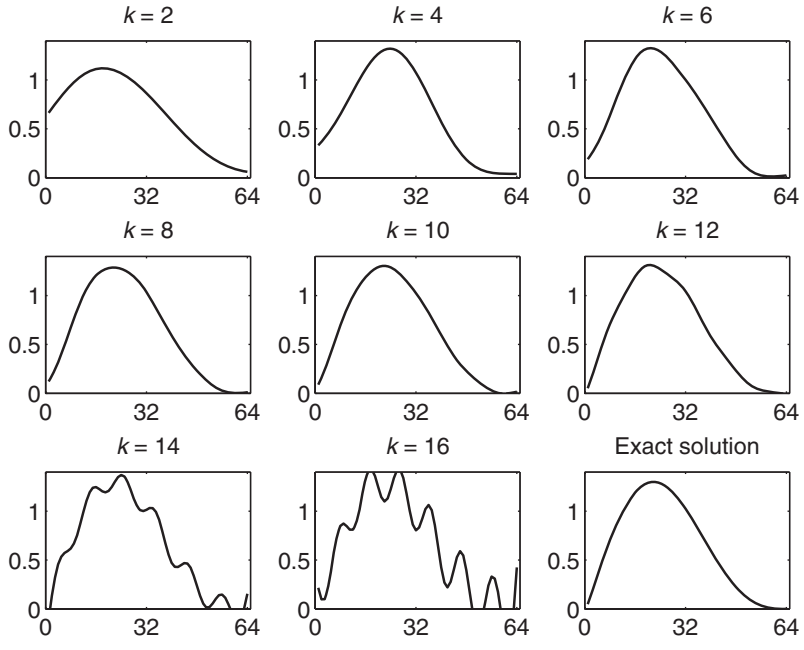

Figure 4.3. *TSVD solutions x_k to the gravity surveying problem for eight different values of k. The exact solution is shown in the bottom right corner. A good approximation is achieved for $k = 10$, while the noise is clearly visible in x_k for $k = 14$ and $k = 16$.*

that we seek the solution with minimum 2-norm:

$$\min \|x\|_2 \qquad \text{subject to} \qquad \|A_k\, x - b\|_2 = \min\,. \tag{4.5}$$

It is easy to show that the solution to this constrained problem is precisely the TSVD solution x_k in (4.2). Hence, (4.5) is an alternative definition of the TSVD solution, in which the regularity requirement on x, in the form of the minimization of its norm $\|x\|_2$, is explicit.

From the above definition of x_k it follows that we can also write the TSVD solution in the form

$$x_k = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_k \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_k^{-1} \end{pmatrix} \begin{pmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{pmatrix}^T b,$$

showing that there exists a matrix[2] $A_k^\dagger = \sum_{i=1}^k v_i\, \sigma_i^{-1} u_i^T$ such that we can write $x_k = A_k^\dagger b$. As a consequence, it follows that if e is Gaussian white noise with $\mathrm{Cov}(e) = \eta^2 I$,

[2]The matrix $A_k^\dagger$ is the pseudoinverse of the TSVD matrix A_k defined in (4.3).

then the covariance matrix for the TSVD solution is

$$\mathrm{Cov}(x_k) = A_k^\dagger \,\mathrm{Cov}(b)\,(A_k^\dagger)^T = \eta^2 \sum_{i=1}^{k} \sigma_i^{-2}\, v_i\, v_i^T.$$

The norm of this matrix is

$$\|\mathrm{Cov}(x_k)\|_2 = \eta^2/\sigma_k^2,$$

and since σ_k is always larger—and often much larger—than σ_n, we conclude that the elements of $\mathrm{Cov}(x_k)$ are generally smaller—and often much smaller—than those of the covariance matrix $\mathrm{Cov}(x)$ for the naive solution.

The price we pay for this reduction of the variance in x_k, compared to the naive solution $x = A^{-1}b$, is bias. While $A^{-1}b$ is unbiased, i.e., the expected value is the exact solution, $\mathcal{E}(A^{-1}b) = x^{\mathrm{exact}}$, the TSVD solution x_k has a nonzero bias:

$$\mathcal{E}(x_k) = \sum_{i=1}^{k} (v_i^T x^{\mathrm{exact}})\, v_i = x^{\mathrm{exact}} - \sum_{i=k+1}^{n} (v_i^T x^{\mathrm{exact}})\, v_i.$$

However, due to the discrete Picard condition, the coefficients $|v_i^T x^{\mathrm{exact}}|$ in the bias term, and therefore also the norm of the bias term,

$$\left\| \sum_{i=k+1}^{n} (v_i^T x^{\mathrm{exact}})\, v_i \right\|_2 = \left(\sum_{i=k+1}^{n} (v_i^T x^{\mathrm{exact}})^2 \right)^{1/2},$$

are typically small, compared to $\|x^{\mathrm{exact}}\|_2 = \left(\sum_{i=1}^{n} (v_i^T x^{\mathrm{exact}})^2\right)^{1/2}$.

A somewhat common, but unfortunate, mistake in connection with TSVD is to base the criterion for choosing the truncation parameter k on the size of the singular values σ_i. As long as the noise is restricted to the right-hand side, the truncation parameter k must define the break-point between the retained and discarded (filtered) SVD coefficients, and therefore the choice of k is determined by the behavior of the noisy coefficients $u_i^T b$.

The confusion seems to arise because the numerical rank of a noisy matrix is related to the singular values as well as the norm of the perturbation matrix; see, e.g., [32] for details about these issues. Also, if $|u_i^T b|$ decay just slightly faster than σ_i, and if they are scaled such that $|u_i^T b| \approx \sigma_i$, then one may not notice the difference.

4.3 Selective SVD

The basic idea in TSVD regularization is to include all the SVD components corresponding to the largest singular values. The rationale underlying these and many other regularization methods is that the underlying problem satisfies the Picard condition. Hence, as demonstrated by our analysis in Section 4.6, we are assured that the regularized solutions will capture significant components of the exact solution.

But is it always necessary to include *all* these SVD components? The answer is, of course, negative. For example, we can easily imagine a problem in which, say, every

second SVD component is zero ($v_2^T x^{\text{exact}} = v_4^T x^{\text{exact}} = v_6^T x^{\text{exact}} = \cdots = 0$). There is no need to include these SVD components.

Bert Rust [61] has suggested a variant of the TSVD method which we shall refer to as *selective SVD*[3] (SSVD). This method is of interest in its own right, and it is also important in connection with some of the iterative methods to be discussed in Chapter 5. In the SSVD method we include, or select, only those SVD components which make significant contributions to the regularized solution. Specifically, given a threshold τ for the right-hand side's SVD coefficients, the SSVD solution x_τ is defined as

$$\boxed{x_\tau \equiv \sum_{|u_i^T b| > \tau} \frac{u_i^T b}{\sigma_i} v_i.} \tag{4.6}$$

We sum all the SVD components $(u_i^T b)/\sigma_i$ for which the absolute value $|u_i^T b|$ of the right-hand side's SVD coefficient is above the threshold τ. Thus, the filter factors for the SSVD method are

$$\varphi_i^{[\tau]} = \begin{cases} 1, & |u_i^T b| \geq \tau, \\ 0, & \text{otherwise.} \end{cases}$$

It is natural to choose the threshold τ such that the SSVD filters remove the coefficients $u_i^T b$ below their noise level. From the analysis in Section 3.5 we know that if the noise in the right-hand side b is white, then the noise in the coefficients $u_i^T b$ is also white with the same statistics. Thus, we can choose

$$\tau = \nu_S \, \eta, \tag{4.7}$$

where ν_S is a "safety factor" which we can set to, say, 3 or 5, in order to avoid including components at the noise level.

Figure 4.4 illustrates the advantage of the SSVD method for a problem where some of the SVD coefficients corresponding to large singular values are small. We use the second-derivative test problem from Exercise 2.3, and the exact solution is a "sawtooth function" $\wedge\!\vee$ constructed such that only each fourth SVD component is nonzero. We added Gaussian white noise with $\eta = 10^{-5}$, and consequently the zero coefficients $u_i^T b^{\text{exact}}$ are perturbed and take the form $u_i^T e$ with $|u_i^T e| \approx 0.8 \cdot 10^{-5}$. The TSVD solution x_k that best approximates x^{exact} is obtained for $k = 30$, leading to the error $\|x^{\text{exact}} - x_k\|_2 \approx 5.7 \cdot 10^{-3}$. However, if we choose the SSVD threshold $\tau = 2.5\,\eta = 2.5 \cdot 10^{-5}$, then we include only SVD coefficients 2, 6, 10, 14, 18, 22, and 26, leading to the smaller error $\|x^{\text{exact}} - x_\tau\|_2 \approx 4.3 \cdot 10^{-3}$. We note that for problems where all the coefficients $u_i^T b^{\text{exact}}$ decay, we cannot expect much difference between the TSVD and SSVD solutions.

The iterative CGLS regularization method, which we will introduce in Section 6.3 for large-scale problems, is another regularization method which seeks to include the significant SVD components only.

[3] The name used in [61] is Truncated Singular Component Method. Another name could be Thresholded SVD, which unfortunately has the same acronym as Truncated SVD.

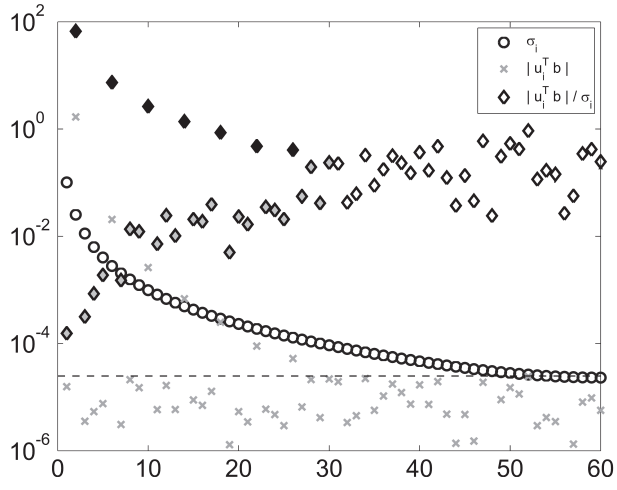

Figure 4.4. *Picard plot that illustrates the advantage of the SSVD method. Only each fourth SVD component gives a significant contribution to the solution, and therefore it can be advantageous to leave out those components in between. The filled diamonds show the nine SVD components that contribute to the SSVD solution for* $\tau = 2.5\,\eta = 2.5 \cdot 10^{-5}$*. The diamonds with gray fill show the additional components that contribute to the optimal TSVD solution.*

4.4 Tikhonov Regularization

The advantage of the TSVD method is that it is intuitive, and it is easy to compute TSVD solutions x_k for different truncation parameters, once the SVD has been computed. The disadvantage is that it explicitly requires the computation of the SVD or, at least, the principal k singular values and vectors. This computational task can be too overwhelming for large-scale problems, and therefore there is a need for other regularization methods that are better suited for large computational problems.

Probably the most successful regularization method of all times is *Tikhonov regularization*. This method has been invented several times, in several incarnations (see Appendix C); but Tikhonov's name is rightly associated with the method because of his ground-breaking work in connection with this method [66]. The method can also be derived from a Bayesian perspective; see, e.g., Section 3.3 in [8]. Similarly with the formulation of TSVD in (4.5), Tikhonov's method explicitly incorporates the regularity requirement in the formulation of the problem. Specifically, the Tikhonov solution x_λ is defined as the solution to the problem

$$\boxed{\min_x \left\{ \|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2 \right\}.} \tag{4.8}$$

Here, the *regularization parameter* λ is a positive parameter that controls the weighting between the two ingredients of the criterion function.

- The first term $\|Ax-b\|_2^2$ measures the goodness-of-fit, i.e., how well the solution x predicts the given (noisy) data b. Obviously, if this term is too large, then x cannot be considered as a good solution because it does not "solve the problem." On the other hand, intuitively we should not make the residual smaller than the average size of the errors in b; similar to data-fitting problems, we do not want to fit the noise in the data.

- The second term $\|x\|_2^2$ measures the regularity of the solution. The incorporation of this term is based on our knowledge that the naive solution is dominated by high-frequency components with large amplitudes, and the hope is therefore that if we control the norm of x, then we can suppress (most of) the large noise components.

- The balance between the two terms is controlled by the factor λ^2. The larger the λ, the more weight is given to the minimization of the solution norm $\|x\|_2$ and thus the regularity of the solution (we note that $x \to 0$ as $\lambda \to \infty$). On the other hand, the smaller the λ, the more weight is given to fitting the noisy data, resulting in solutions that are less regular (we obtain the original problem and the "naive" solution when $\lambda = 0$).

The goal is to find a good balance between these two terms, via a suitable value of λ, such that the regularized solution x_λ is sufficiently regular and—at the same time—fits the data well enough. The hope is then that we achieve a regularized solution that approximates the exact solution.

While it may not be immediately clear from the formulation in (4.8), this is a linear least squares problem in x. But if we use the fact that, for arbitrary vectors y and z,

$$\left\| \begin{pmatrix} y \\ z \end{pmatrix} \right\|_2^2 = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} y \\ z \end{pmatrix} = y^T y + z^T z = \|y\|_2^2 + \|z\|_2^2 \,,$$

then it follows immediately that the Tikhonov problem can be reformulated as

$$\min_x \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 , \tag{4.9}$$

which is clearly a linear least squares problem. The normal equations for this problem take the form

$$\begin{pmatrix} A \\ \lambda I \end{pmatrix}^T \begin{pmatrix} A \\ \lambda I \end{pmatrix} x = \begin{pmatrix} A \\ \lambda I \end{pmatrix}^T \begin{pmatrix} b \\ 0 \end{pmatrix}$$

or simply

$$(A^T A + \lambda^2 I)\, x = A^T b \qquad \Leftrightarrow \qquad x_\lambda = (A^T A + \lambda^2 I)^{-1} A^T b,$$

the latter of which is also frequently found in the literature. However, for reasons of both computational efficiency and numerical stability, algorithms for computing Tikhonov solutions should be based on the formulation in (4.9).

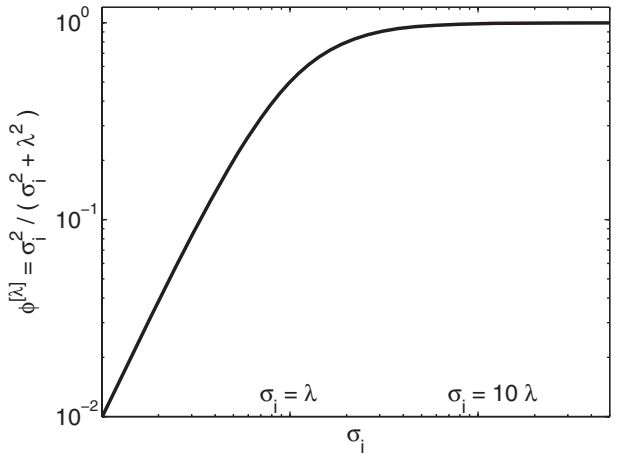

Figure 4.5. *The filter factors* $\varphi_i^{[\lambda]} = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ *for Tikhonov regularization.*

We can use the SVD to obtain more insight into the Tikhonov solution x_λ. When we insert the SVD of A into the normal equations and use that $I = V\,V^T$, then we obtain

$$\begin{aligned} x_\lambda &= (V\,\Sigma^2\,V^T + \lambda^2\,V\,V^T)^{-1} V\,\Sigma\,U^T b \\ &= V\,(\Sigma^2 + \lambda^2\,I)^{-1} V^T V\,\Sigma\,U^T b \\ &= V\,(\Sigma^2 + \lambda^2\,I)^{-1} \Sigma\,U^T b, \end{aligned}$$

which leads to the following expression:

$$x_\lambda = V\,(\Sigma^2 + \lambda^2\,I)^{-1} \Sigma\,U^T b.$$

If we insert the singular values and vectors, we get

$$x_\lambda = \sum_{i=1}^{n} \varphi_i^{[\lambda]}\,\frac{u_i^T b}{\sigma_i}\,v_i, \tag{4.10}$$

where we have introduced the *filter factors* $\varphi_i^{[\lambda]}$ for $i = 1, \ldots, n$, which satisfy

$$\varphi_i^{[\lambda]} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \approx \begin{cases} 1, & \sigma_i \gg \lambda, \\ \sigma_i^2/\lambda^2, & \sigma_i \ll \lambda. \end{cases} \tag{4.11}$$

The behavior of these filter factors is illustrated in Figure 4.5. We see that for singular values σ_i larger than the parameter λ, the filter factors are close to one and the corresponding SVD components contribute to x_λ with almost full strength. On the other hand, for singular values much smaller than λ the filter factors are small, and therefore these SVD components are damped or filtered. We note that in the latter case, the filter factors $\varphi_i^{[\lambda]}$ are proportional to σ_i^2, and therefore they decay fast enough to "kill" the increasing factors $u_i^T b/\sigma_i \approx u_i^T e/\sigma_i$.

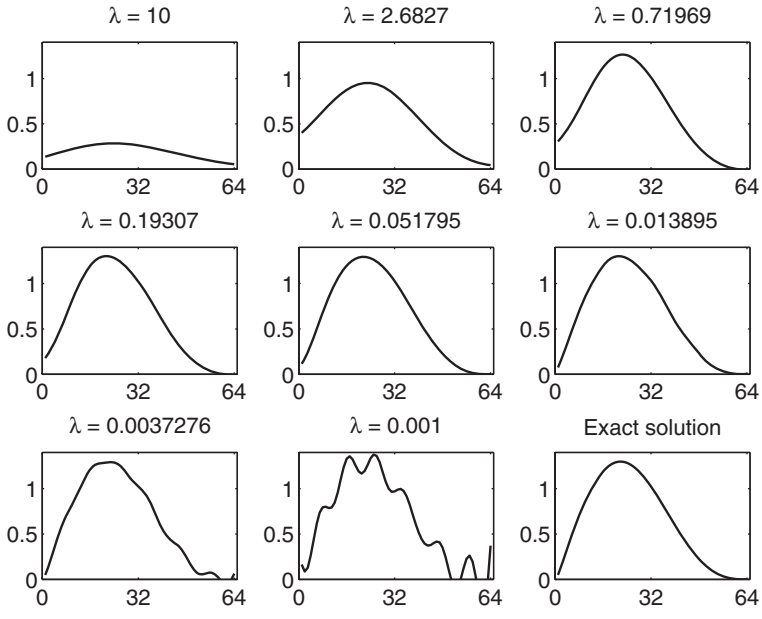

Figure 4.6. *Tikhonov solutions x_λ to the gravity surveying problem, for eight different values of the regularization parameter λ. The exact solution is shown in the bottom right figure. The values $\lambda = 0.0037$ and $\lambda = 0.001$ are clearly too small, producing solutions with too many noisy SVD components.*

The conclusion of this SVD analysis is that the Tikhonov solution is a filtered solution, in much the same manner as the TSVD solution. The transition in the filter factors $\varphi_i^{[\lambda]}$ sets in for singular values whose size is comparable to λ, and in this way we can use λ to control the filtering (similar to the use of the truncation parameter k in TSVD); see the example in Figure 4.6 and compare the Tikhonov solutions with the TSVD solutions in Figure 4.3. However, the transition between retained and filtered SVD components is smoother, and the filtering can be achieved without explicit computation of the SVD; all that is needed is to solve the least squares problem in (4.9).

Again, there are alternative formulations of the problem which can give insight and/or be used to devise algorithms for certain classes of problems. For example, we can specify an upper bound δ for the norm of the solution and solve the constrained least squares problem

$$\min_x \|Ax - b\|_2^2 \qquad \text{subject to} \qquad \|x\|_2^2 \leq \delta^2. \tag{4.12}$$

The constraint is active whenever δ is smaller than the norm $\|A^{-1}b\|_2$ of the naive solution, which should always be the case if we want to regularize the solution. From optimization theory, we know that we can incorporate the constraint via a Lagrange

multiplier γ, and the problem becomes

$$\min_x \left\{ \|Ax - b\|_2^2 + \gamma \left(\|x\|_2^2 - \delta^2 \right) \right\}.$$

The gradient with respect to x is $A^T(Ax - b) + \gamma x$, and by setting the gradient to zero we obtain the normal equations for the Tikhonov solution with $\gamma = \lambda^2$. Hence, the constrained problem (4.12) is equivalent to the Tikhonov problem.

Similarly, we can minimize the norm of the solution (again, to suppress high-frequency components with large amplitude), subject to the constraint that the residual norm is smaller than some upper bound ε:

$$\min_x \|x\|_2^2 \qquad \text{subject to} \qquad \|Ax - b\|_2^2 \leq \varepsilon^2.$$

Again, the Lagrange multiplier formulation of this constrained problem leads to the Tikhonov formulation.

Analogous with the TSVD solution, let us take a brief look at the statistical aspects of the Tikhonov solution. Via the relation $x_\lambda = (A^TA + \lambda^2 I)^{-1}A^Tb$, and assuming again that $\mathrm{Cov}(e) = \eta^2 I$, we can show that the covariance matrix for the Tikhonov solution is

$$\mathrm{Cov}(x_\lambda) = \eta^2 \sum_{i=1}^{n} \left(\varphi_i^{[\lambda]}\right)^2 \sigma_i^{-2}\, v_i\, v_i^T,$$

and its norm is bounded as

$$\|\mathrm{Cov}(x_\lambda)\|_2 \leq \frac{\eta^2}{(2\,\lambda)^2}.$$

Obviously, if λ is chosen somewhat larger than the smallest singular value σ_n, then we obtain a reduction in the variance for the Tikhonov solution, compared to the naive solution. Again, we achieve this at the expense of introducing a bias in the solution:

$$\mathcal{E}(x_\lambda) = \sum_{i=1}^{n} \varphi_i^{[\lambda]}\, (v_i^T x^{\mathrm{exact}})\, v_i = x^{\mathrm{exact}} - \sum_{i=1}^{n} (1 - \varphi_i^{[\lambda]})\, (v_i^T x^{\mathrm{exact}})\, v_i.$$

Since $1 - \varphi_i^{[\lambda]} = \lambda^2/(\sigma_i^2 + \lambda^2)$, and since we assume that the discrete Picard condition is satisfied, we can visually expect that the bias term is small compared to $\|x^{\mathrm{exact}}\|_2$.

4.5 Perturbation Theory*

Regularization methods are designed to filter the influence from the noise, such that the solution is less dominated by the inverted noise, provided we can choose the regularization parameter (k or λ) properly. The question is then: How sensitive is the regularized solution to the perturbations, and how does the sensitivity depend on the regularization parameter?

In order to study this sensitivity, throughout this section we will be studying two related problems, namely,

$$Ax = b \qquad \text{and} \qquad \widetilde{A}\,\widetilde{x} = \widetilde{b},$$

where $\widetilde{A}$ and $\widetilde{b}$ are perturbed versions of A and b,

$$\widetilde{A} = A + \Delta A \qquad \text{and} \qquad \widetilde{b} = b + \Delta b,$$

i.e., the matrix ΔA is the perturbation of A, and the vector Δb is the perturbation of the right-hand side b. Think of the two problems $Ax = b$ and $\widetilde{A}\widetilde{x} = \widetilde{b}$ as two different noisy realizations of the underlying problem with exact data. The vectors x and $\widetilde{x}$ are the solutions to the two problems, and we are interested in bounding the difference $\widetilde{x} - x$. More precisely, we are interested in upper bounds for the relative normwise difference $\|\widetilde{x} - x\|_2 / \|x\|_2$, which is independent on the scaling of A, b, and x. These bounds tell us how sensitive the solution is to (variations in) the noise.

Naive solutions. The perturbation bound for the naive solutions $x = A^{-1}b$ and $\widetilde{x} = \widetilde{A}^{-1}\widetilde{b}$ (when A is square and invertible) can be found in many text books on numerical analysis, and it takes the following form. If the perturbation matrix ΔA satisfies $\|\Delta A\|_2 < \sigma_n$, then

$$\frac{\|\widetilde{x} - x\|_2}{\|x\|_2} \leq \frac{\mathrm{cond}(A)}{1 - \gamma} \left(\frac{\|\Delta b\|_2}{\|b\|_2} + \frac{\|\Delta A\|_2}{\|A\|_2} \right),$$

where

$$\gamma = \|\Delta A\|_2 \, \|A^{-1}\|_2 = \frac{\|\Delta A\|_2}{\sigma_n}.$$

The requirement that ΔA satisfy $\|\Delta A\|_2 < \sigma_n \Leftrightarrow \gamma < 1$ is necessary for ensuring that the perturbed matrix $\widetilde{A}$ stays nonsingular. The perturbation is governed by A's condition number $\mathrm{cond}(A) = \sigma_1/\sigma_n$.

Least squares solutions. The perturbation bound for the least squares solutions to $\min \|Ax - b\|_2$ and $\min \|\widetilde{A}\widetilde{x} - \widetilde{b}\|_2$ (when $m > n$ and A has full rank) can be found in many text books on least squares problems (such as [5]). If the perturbation matrix satisfies $\|\Delta A\|_2 < \sigma_n$ (which ensures that ΔA also has full rank), then

$$\frac{\|\widetilde{x} - x\|_2}{\|x\|_2} \leq \frac{\mathrm{cond}(A)}{1 - \gamma} \left(\frac{\|\Delta b\|_2}{\|b_n\|_2} + \frac{\|\Delta A\|_2}{\|A\|_2} + \gamma \frac{\|b - b_n\|_2}{\|b_n\|_2} \right),$$

where $b_n = Ax$ and $\gamma = \|\Delta A\|_2/\sigma_n$. Again, the perturbation is governed by the condition number $\mathrm{cond}(A) = \sigma_1/\sigma_n$.

TSVD solutions [28]. Let x_k and $\widetilde{x}_k$ denote the TSVD solutions for the same truncation parameter k, and assume that the perturbation matrix ΔA satisfies $\|\Delta A\|_2 < \sigma_k - \sigma_{k+1}$; then

$$\frac{\|\widetilde{x}_k - x_k\|_2}{\|x_k\|_2} \leq \frac{\kappa_k}{1 - \gamma_k} \left(\frac{\|\Delta b\|_2}{\|b_k\|_2} + \frac{\|\Delta A\|_2}{\|A\|_2} + \frac{\gamma_k}{1 - \gamma_k - \hat{\gamma}_k} \frac{\|b - b_k\|_2}{\|b_k\|_2} \right),$$

where $b_k = Ax_k$ and

$$\kappa_k = \mathrm{cond}(A_k) = \frac{\sigma_1}{\sigma_k} = \frac{\|A\|_2}{\sigma_k}, \qquad \gamma_k = \frac{\|\Delta A\|_2}{\sigma_k}, \qquad \hat{\gamma}_k = \frac{\sigma_{k+1}}{\sigma_k}.$$

This result shows that the condition number for the TSVD solution is $\kappa_k = \sigma_1/\sigma_k$, which can be much smaller than the condition number for A.

Tikhonov solutions [26]. Let x_λ and $\widetilde{x}_\lambda$ denote the Tikhonov solutions for the same regularization parameter λ, and assume that the perturbation matrix ΔA satisfies $\|\Delta A\|_2 < \lambda$; then

$$\frac{\|\widetilde{x}_\lambda - x_\lambda\|_2}{\|x_\lambda\|_2} \leq \frac{\kappa_\lambda}{1-\gamma_\lambda} \left(\frac{\|\Delta b\|_2}{\|b_\lambda\|_2} + 2\,\frac{\|\Delta A\|_2}{\|A\|_2} + \gamma_\lambda\,\frac{\|b-b_\lambda\|_2}{\|b_\lambda\|_2} \right),$$

where $b_\lambda = A\,x_\lambda$ and

$$\kappa_\lambda = \frac{\sigma_1}{\lambda} = \frac{\|A\|_2}{\lambda}\ , \qquad \gamma_\lambda = \frac{\|\Delta A\|_2}{\lambda}\ .$$

Hence the condition number for the Tikhonov solution is $\kappa_\lambda = \sigma_1/\lambda$, and, similar to the TSVD condition number, it can be much smaller than the condition number for A.

The perturbation bounds for the TSVD and Tikhonov solutions are, to no big surprise, quite similar to each other. In both cases, the condition number is governed by the choice of the regularization parameter, k or λ, and the more filtering the smaller the condition number. And in both cases, the size of the residual $b - b_k = b - A\,x_k$ or $b - b_\lambda = b - A\,x_\lambda$ enters the perturbation bound. Also, in both cases, we assume an upper bound on the norm of the perturbation matrix ΔA; this bound ensures that the SVD components of the perturbed and the unperturbed problems are related.

The main difference between the two perturbation bounds is the factor $\hat{\gamma}_k = \sigma_{k+1}/\sigma_k$, which measures the gap between the smallest retained singular value and the largest discarded one, and which appears only in the TSVD bound. If $\hat{\gamma}_k$ is close to one, the $1-\hat{\gamma}_k$ is small, and only a very small perturbation of the matrix is allowed. Hence, one should refrain from truncating in the middle of a cluster of singular values, i.e., a set of singular values which are very close and well separated from the others. The same difficulty does not apply to the Tikhonov solution; if one chooses a value of λ inside a cluster of singular values, then all these SVD components are included in the Tikhonov solution because all the associated filter factors $\varphi_i^{[\lambda]} = \sigma_i^2/(\sigma_i^2+\lambda^2)$ are of the same size.

In addition to studying the errors in the solutions, as we have done in the above results, it may be of interest to study the "prediction errors," i.e., the sensitivity of the vectors $A\,x_k$ and $A\,x_\lambda$. Under the same assumption as before, we have for the TSVD and Tikhonov solutions

$$\frac{\|\widetilde{A}\,\widetilde{x}_k - A\,x_k\|_2}{\|b\|_2} \leq \frac{\gamma_k}{1-\gamma_k} + \frac{\|\Delta b\|_2}{\|b\|_2},$$
$$\frac{\|\widetilde{A}\,\widetilde{x}_\lambda - A\,x_\lambda\|_2}{\|b\|_2} \leq \gamma_\lambda + \frac{\|\Delta b\|_2}{\|b\|_2}.$$

The interesting result here is that the prediction errors do not depend on the condition number—a result which is well known for the naive solution.

We finish this section with a small example that illustrates the sensitivity of the Tikhonov solution x_λ to perturbations of b, as a function of the regularization parameter λ. The matrix, the unperturbed right-hand side, and the unperturbed

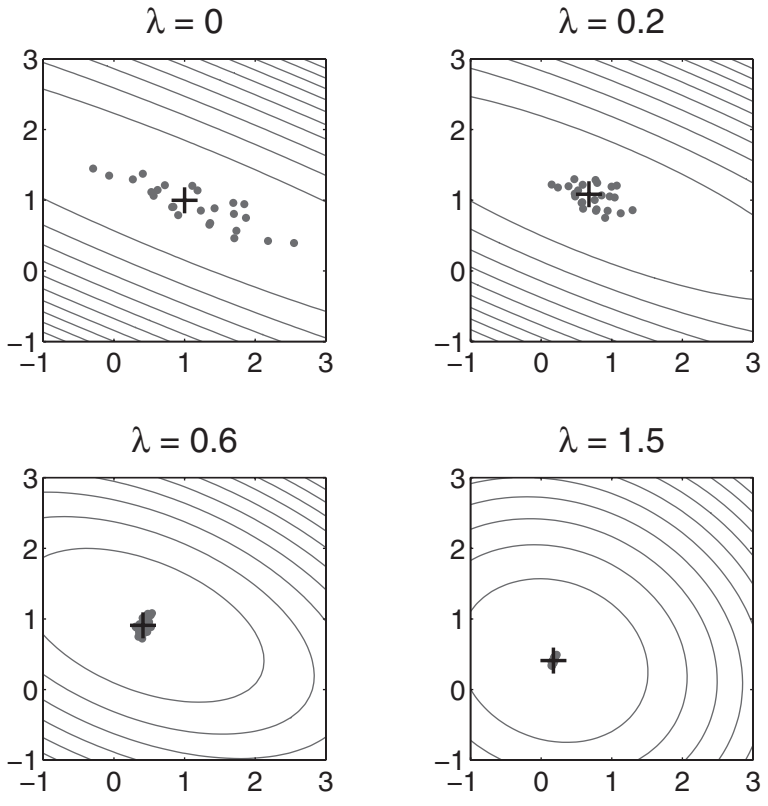

Figure 4.7. *Illustration of the sensitivity of the Tikhonov solutions to perturbations of the right-hand side for a small* 2×2 *test problem. For each of* 25 *random perturbations we computed* x_λ *for four different values of* λ*, given in the top of the plots. The ellipsoidal curves are the level curves for the associated Tikhonov problem.*

solution are are

$$A = \begin{pmatrix} 0.41 & 1.00 \\ -0.15 & 0.06 \end{pmatrix}, \qquad b = \begin{pmatrix} 1.41 \\ -0.09 \end{pmatrix}, \qquad x = \begin{pmatrix} 1.00 \\ 1.00 \end{pmatrix}.$$

We generated 25 random perturbations $\widetilde{b} = b + \Delta b$ with the perturbation scaled such that $\|\Delta b\|_2 / \|b\|_2 = 0.15$, and for each perturbed problem we computed the Tikhonov solutions corresponding to four values of λ. The results are shown in Figure 4.7, where the black cross indicates the unperturbed solution, while the gray dots represent the perturbed solutions. Also included in the plots are the ellipsoidal level curves for the Tikhonov (least squares) problem for the particular λ.

We see that as λ increases, and more regularization (or filtering) is imposed on the problem, the less sensitive the Tikhonov solution is to the perturbations. As λ increases, the condition number becomes smaller and the level curves become less elliptic. Note that for small values where the ellipsoids are more elongated, the per-

turbation is mainly in the direction of the longer semiaxis. This direction is defined by the right singular vector v_2 corresponding to the smaller singular value σ_2, and so the observed results agree with the theory, namely, that the largest perturbations occur in the directions corresponding to the smallest singular values.

4.6 The Role of the Discrete Picard Condition*

We have already explained that satisfaction of the discrete Picard condition is crucial for the existence of a meaningful solution to discrete ill-posed problems. In this section we take another look at this condition and the role it plays in the computation of regularized solutions.

Throughout this section (which is based on [27]) we will study a model problem in which the distribution of singular values is "arbitrary" (still assuming that they decay gradually to zero), and where the SVD coefficients of the exact right-hand side are given by the model

$$u_i^T b^{\text{exact}} = \sigma_i^\alpha , \qquad i = 1, \ldots, n, \tag{4.13}$$

in which $\alpha \geq 0$ is a model parameter that controls the decay of these coefficients, relative to the decay of the singular values.

- The larger the α, the faster the decay of the SVD coefficients.
- For $\alpha > 1$ the coefficients decay *faster* than the singular values, and the discrete Picard condition is satisfied.

While this is, indeed, a crude model, it reflects the overall behavior often found in real problems.

Figure 4.8 shows two examples of this model for different values of α. In the left part we show the singular values σ_i and the SVD coefficient $|u_i^T b|$ and $|u_i^T b/\sigma_i|$ for the `shaw` test problem from Exercise 3.6, together with the model quantities σ_i^α and $\sigma_i^{\alpha-1}$ for $\alpha = 1.3$. While the latter come from the crude model (4.13), they are indeed able to track the overall decay of the SVD quantities for this test problem, where the right-hand side coefficients decay faster than the singular values, and where the solution coefficients also decay. (The same is true for other model problems, with other choices of α.) In the right part of Figure 4.8 we show the model quantities σ_i^α and $\sigma_i^{\alpha-1}$ for $\alpha = 0.6$. This model corresponds to a situation where the right-hand side coefficients decay more slowly than the singular values, and the solution coefficients increase with i.

Recall from the previous sections that we can always write the TSVD solution as $x_k = A_k^\dagger b$, where $A_k^\dagger = \sum_{i=1}^k v_i \, \sigma_i^{-1} u_i^T$. Inserting $b = b^{\text{exact}} + e$, we obtain

$$x_k = A_k^\dagger \, b^{\text{exact}} + A_k^\dagger \, e.$$

Similarly, we can always write the Tikhonov solution as $x_\lambda = (A^T A + \lambda^2 I)^{-1} A^T b$, and therefore we have

$$x_\lambda = (A^T A + \lambda^2 I)^{-1} A^T \, b^{\text{exact}} + (A^T A + \lambda^2 I)^{-1} A^T \, e.$$

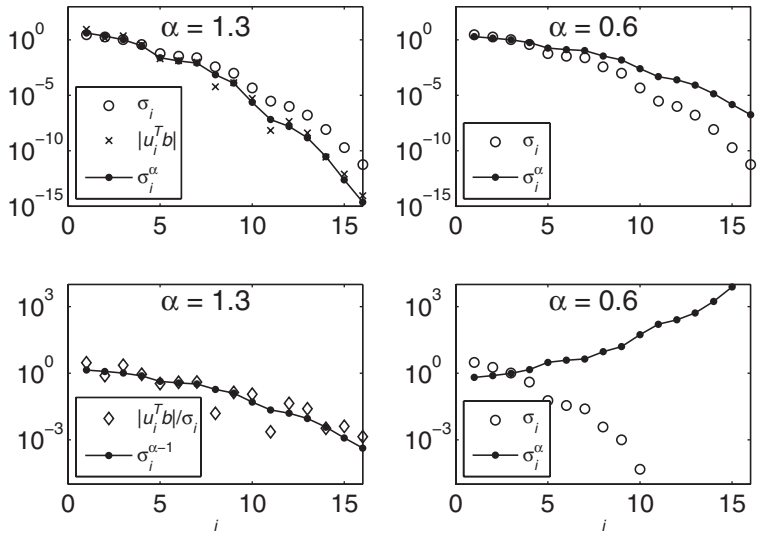

Figure 4.8. *Illustration of the model in* (4.13) *using the singular values for the* `shaw` *test problem from Exercise* 3.6. *Note that we have decaying solution coefficients only when* $\alpha > 1$. *In the left part we also show the SVD quantities* $|u_i^T b|$ *and* $|u_i^T b|/\sigma_i$ *for the* `shaw` *problem.*

In both methods, we want to choose the regularization parameter such that the second term, the inverted and filtered noise, is sufficiently suppressed. At the same time, we do not want to impose too much filtering; we want the first term, either $A_k^\dagger\, b^{\text{exact}}$ or $(A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}}$, to be as close to x^{exact} as possible. This is at the heart of the parameter-choice methods for regularization algorithms, and we return to this important issue in Chapter 5.

Our aim here is to take a closer look at the differences between the exact solution x^{exact} and the first term $A_k^\dagger\, b^{\text{exact}}$ or $(A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}}$. These differences are called the *regularization errors* and they are given by

$$\varepsilon_k \equiv x^{\text{exact}} - A_k^\dagger\, b^{\text{exact}} = \sum_{i=k+1}^{n} (v_i^T x^{\text{exact}})\, v_i,$$

$$\varepsilon_\lambda \equiv x^{\text{exact}} - (A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}} = \sum_{i=1}^{n} \frac{\lambda^2}{\sigma_i^2 + \lambda^2}\, (v_i^T x^{\text{exact}})\, v_i.$$

Note that the regularization errors defined here are identical to the bias terms, as discussed previously.

We will now give upper bounds for the infinity-norms $\|\varepsilon_k\|_\infty = \max_i |(\varepsilon_k)_i|$ and $\|\varepsilon_\lambda\|_\infty = \max_i |(\varepsilon_\lambda)_i|$, i.e., the largest components in absolute value of the errors. Assume that the exact right-hand side satisfies the model (4.13). Then the regularization

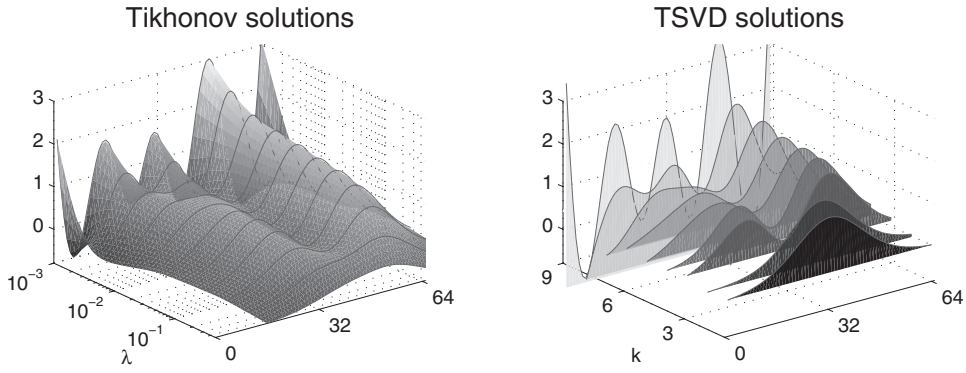

Figure 4.9. *Tikhonov and TSVD solutions to the same problem. The left figure shows Tikhonov solutions x_λ, varying continuously with λ between 10^{-3} and 1. The right figure shows nine TSVD solutions x_k for $k = 1, \ldots, 9$.*

error (or bias term) for the TSVD solution is bounded as

$$\frac{\|\varepsilon_k\|_\infty}{\|x^{\text{exact}}\|_2} \leq \begin{cases} 1, & 0 \leq \alpha < 1, \\ \left(\frac{\sigma_k}{\sigma_1}\right)^{\alpha-1}, & 1 \leq \alpha. \end{cases} \tag{4.14}$$

The corresponding bound for the Tikhonov solution is

$$\frac{\|\varepsilon_\lambda\|_\infty}{\|x^{\text{exact}}\|_2} \leq \begin{cases} 1, & 0 \leq \alpha < 1, \\ \left(\frac{\lambda}{\sigma_1}\right)^{\alpha-1}, & 1 \leq \alpha < 3, \\ \left(\frac{\lambda}{\sigma_1}\right)^{2}, & 3 \leq \alpha. \end{cases} \tag{4.15}$$

We conclude that if the discrete Picard condition is not satisfied, i.e., if $\alpha \leq 1$, then the upper bounds are 1 and there is no guarantee that the regularization errors (bias terms) are small. In other words, there is no guarantee that the terms $A_k^\dagger\, b^{\text{exact}}$ or $(A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}}$ will approximate x^{exact}. On the other hand, if $\alpha > 1$ and the discrete Picard condition is satisfied, then the faster the decay the better these terms approximate x^{exact}. This clearly illustrates the importance of the discrete Picard condition.

TSVD and Tikhonov regularization are clearly related, due to the similarity of the filter factors for the two methods. Figure 4.9 illustrates this: for each value of k there exists a λ such that $x_k \approx x_\lambda$. It is therefore interesting to study the similarity of the TSVD and Tikhonov solutions; more precisely, we want to study the similarity of the noise-free regularized terms $A_k^\dagger\, b^{\text{exact}}$ or $(A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}}$. Let

$$\varepsilon_{k,\lambda} \equiv A_k^\dagger\, b^{\text{exact}} - (A^T A + \lambda^2 I)^{-1} A^T\, b^{\text{exact}},$$

and assume again that b^{exact} satisfies the model (4.13). Then, for a fixed value of k,

$$\min_{\lambda} \frac{\|\varepsilon_{k,\lambda}\|_\infty}{\|x^{\text{exact}}\|_2} \leq \begin{cases} \left(\frac{\sigma_{k+1}}{\sigma_k}\right)^{\alpha-1}, & 0 \leq \alpha < 1, \\ \left(\frac{\sigma_k}{\sigma_1}\right)^{\alpha-1}, & 1 \leq \alpha < 3, \\ \left(\frac{\sigma_k}{\sigma_1}\right)^{2}, & 3 \leq \alpha, \end{cases}$$

and the minimum is attained for $\lambda = (\sigma_k^3\, \sigma_{k+1})^{1/4}$. Again, the importance of the discrete Picard condition is obvious: α must be larger than 1 in order to ensure that the TSVD and Tikhonov solutions produce similar results.

4.7 The L-Curve

In the previous sections we have used the SVD extensively to define and study the behavior of two important regularization methods. We continue our use of the SVD as an analysis tool, and we take a closer look at the solution norm and the corresponding residual norm. These norms play a central role in the practical treatment of discrete ill-posed problems, because they can always be computed no matter which regularization method is used; they do not require the computation of the SVD or any other decomposition of the matrix.

The norm of the TSVD solution and associated residual vary monotonically with k, as can be seen from the expressions

$$\|x_k\|_2^2 = \sum_{i=1}^{k} \left(\frac{u_i^T b}{\sigma_i}\right)^2 \;\leq\; \|x_{k+1}\|_2^2, \tag{4.16}$$

$$\|A\,x_k - b\|_2^2 = \sum_{i=k+1}^{n} (u_i^T b)^2 + \varepsilon_\perp^2 \;\geq\; \|A\,x_{k+1} - b\|_2^2, \tag{4.17}$$

where $\varepsilon_\perp = \|(I - U\,U^T)\; b\|_2$ is the norm of that component of b which lies outside the column space of A. Similarly, the norm of the Tikhonov solution and its residual are given by

$$\|x_\lambda\|_2^2 = \sum_{i=1}^{n} \left(\varphi_i^{[\lambda]}\, \frac{u_i^T b}{\sigma_i}\right)^2, \tag{4.18}$$

$$\|A\,x_\lambda - b\|_2^2 = \sum_{i=1}^{n} \left((1 - \varphi_i^{[\lambda]})\, u_i^T b\right)^2 + \varepsilon_\perp^2, \tag{4.19}$$

where $\varphi_i^{[\lambda]} = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ are the filter factors. To see that these norms vary monotonically with λ, we introduce the notation

$$\xi = \|x_\lambda\|_2^2 \qquad \text{and} \qquad \rho = \|A\,x_\lambda - b\|_2^2.$$

Then we can show that

$$\xi' \equiv \frac{d\xi}{d\lambda} = -\frac{4}{\lambda}\sum_{i=1}^{n}(1-\varphi_i^{[\lambda]})\left(\varphi_i^{[\lambda]}\right)^2\frac{(u_i^T b)^2}{\sigma_i^2},$$

$$\rho' \equiv \frac{d\rho}{d\lambda} = \frac{4}{\lambda}\sum_{i=1}^{n}(1-\varphi_i^{[\lambda]})^2\,\varphi_i^{[\lambda]}\,(u_i^T b)^2 = -\lambda^2\,\xi'.$$

We see that $\xi' < 0$ and $\rho' > 0$ for all λ, confirming the monotonicity of the norms as functions of λ. Also, from the relation $\rho' = -\lambda^2\,\xi'$ we obtain $d\xi/d\rho = -\lambda^{-2}$, showing that that the squared solution norm $\|x_\lambda\|_2^2$ is a monotonically decreasing function of the squared residual norm $\|A\,x_\lambda - b\|_2^2$. Since the square root function is monotonic, the same is true for the norms themselves.

There is also a simple relationship between the second derivatives of ξ and ρ. Specifically, we have

$$\rho'' \equiv \frac{d^2\rho}{d\lambda^2} = \frac{d}{d\lambda}\left(-\lambda^2\,\xi'\right) = -2\,\lambda\,\xi' - \lambda^2\,\xi'',$$

in which $\xi'' \equiv d^2\xi/d\lambda^2$. Consider now the curve obtained by plotting ξ versus ρ, with λ as the parameter. The curvature c_λ of this curve, as a function of λ (see any book on analytical geometry for its definition), is given by

$$c_\lambda = \frac{\rho'\,\xi'' - \rho''\,\xi'}{((\rho')^2 + (\xi')^2)^{3/2}} = \frac{2\,\lambda\,(\xi')^2}{((\rho')^2 + (\xi')^2)^{3/2}},$$

where we have inserted the above relation for ρ''. Hence, $c_\lambda > 0$ for all λ, and therefore the curve (ρ, ξ) is convex.

The curve is of interest because it shows how the regularized solution changes as the regularization parameter λ changes. We can say more about this curve; for example, it is easy to see that

$$0 \le \|x_\lambda\|_2 \le \|A^{-1}b\|_2 \qquad \text{and} \qquad 0 \le \|A\,x_\lambda - b\|_2 \le \|b\|_2.$$

Also, since any point (ρ, ξ) on the curve is a solution to the problem

$$\rho = \min\ \|A\,x - b\|_2^2 \qquad \text{subject to} \qquad \|x\|_2^2 \le \xi$$

in (4.12), it follows that the curve defines a boundary between two regions of the first quadrant: it is impossible to choose a vector x such that the point $(\ \|A\,x-b\|_2^2\,,\ \|x\|_2^2\ )$ lies below the curve; only points on or above the curve can be attained. The same is, of course, true for the curve $(\ \|A\,x - b\|_2\,,\ \|x\|_2\ )$. But when either of these curves is plotted in linear scale, it is difficult to inspect its features because of the large range of values for the two norms.

As shown in [40], the features become more pronounced (and easier to inspect) when the curve is plotted in double-logarithmic scale. The associated curve

$$(\ \tfrac{1}{2}\log\rho\,,\ \tfrac{1}{2}\log\xi\ ) = (\ \log\|A\,x_\lambda - b\|_2\,,\ \log\|x_\lambda\|_2\ )$$

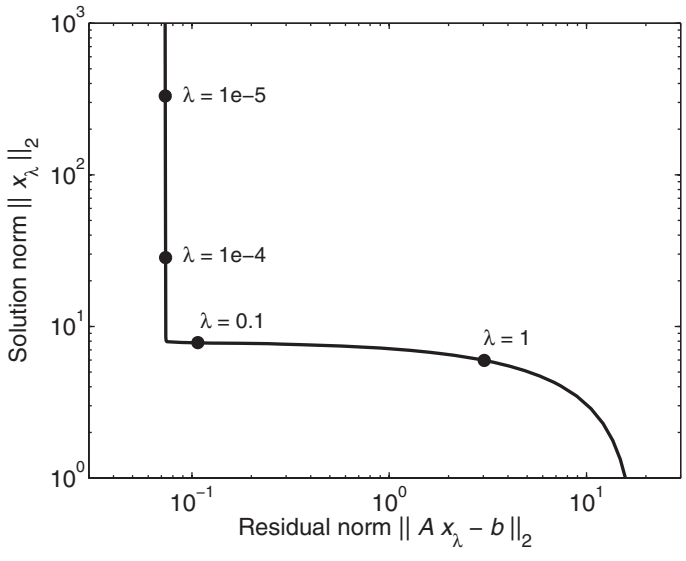

Figure 4.10. *The L-curve for Tikhonov regularization: A log-log plot of the solution norm $\|x_\lambda\|_2$ versus the residual norm $\|A\,x_\lambda - b\|_2$ with λ as the parameter. Notice the vertical and horizontal parts, corresponding to solutions that are under- and over-smoothed, respectively.*

is referred to as the *L-curve* for Tikhonov regularization, and it plays a central role in the analysis of discrete ill-posed problems. Figure 4.10 shows an example of a typical L-curve.

We have already seen that the Tikhonov solution behaves quantitatively differently for small and large values of λ. The same is true for the different parts of the L-curve. When λ is large, then x_λ is dominated by SVD coefficients whose main contribution is from the exact right-hand side b^{exact}—the solution is oversmoothed. A careful analysis (see [34]) shows that for large values of λ we have that

$$\|x_\lambda\|_2 \approx \|x^{\text{exact}}\|_2 \text{ (a constant)} \qquad \text{and} \qquad \|A\,x_\lambda - b\|_2 \text{ increases with } \lambda.$$

For small values of λ the Tikhonov solution is dominated by the perturbation errors coming from the inverted noise—the solution is undersmoothed, and we have that

$$\|x_\lambda\|_2 \text{ increases with } \lambda^{-1} \qquad \text{and} \qquad \|A\,x_\lambda - b\|_2 \approx \|e\|_2 \text{ (a constant)}.$$

The conclusion is that the L-curve has two distinctly different parts, namely, a part that is approximately horizontal, and a part that is approximately vertical. We return to these two parts of the L-curve in Section 5.3.

The "corner" that separates these two parts is located roughly at the point

$$(\,\log\|e\|_2\,,\,\log\|x^{\text{exact}}\|_2\,).$$

Toward the right, for $\lambda \to \infty$, the L-curve starts to bend down as the increasing amount of regularization forces the solution norm toward zero. Likewise, toward the

top left and above the vertical part, the L-curve will eventually become less steep as $\lambda \to 0$ and $\|x_\lambda\|_2 \to \|A^{-1}b\|_2$ (this part is not visible in Figure 4.10).

This curve is found in many books and papers (perhaps starting with the pioneering book by Lawson and Hanson [51]), and a careful analysis of many of its properties can be found in [30]. It can also be used to determine a suitable value of the regularization parameter (this use of the L-curve was not mentioned in [51]); we return to this aspect in Chapter 5.

4.8 When the Noise Is Not White—Regularization Aspects

The analysis in this chapter has, so far, assumed that the noise in the right-hand side is white. We will now study how the TSVD regularization method behaves when the noise is colored, we will demonstrate that HF and LF noise (cf. Section 3.6.3) of the same noise level has different impact on the regularized solution, and we will introduce the technique of prewhitening.

4.8.1 Dealing with HF and LF Noise

We start by generating the `deriv2` test problem for $n = 64$, and we use the call `deriv2(n,3)`, which generates a problem whose exact solution has the shape of a triangle. Then we generate white Gaussian noise e = `randn(n,1)` as well as colored HF and LF noise e_{HF} and e_{LF}. The colored noise is obtained by multiplying (or filtering) the SVD components of the white noise with factors between 0.01 and 1 as follows:

$$e_{\mathrm{HF}} = \text{U*(logspace(-2,0,n)'*(U'*e)),}$$

$$e_{\mathrm{LF}} = \text{U*(logspace(0,-2,n)'*(U'*e)).}$$

In the HF noise, the high-frequency components (which correspond to the smallest singular values) are 100 times larger than the low-frequency components, and vice versa for the LF noise. All three noise vectors are then scaled to have a relative noise level of $\mathtt{rnl} = 10^{-3}$.

Figure 4.11 shows the SVD components $U^T b^{\mathrm{exact}}$ of the exact right-hand side together with the SVD components $U^T e$, $U^T e_{\mathrm{HF}}$, and $U^T e_{\mathrm{LF}}$ of the three different noise realizations. The coefficients $u_i^T b^{\mathrm{exact}}$ decay, as expected for a right-hand side that satisfies the discrete Picard condition. Also, as expected, the coefficients $u_i^T e$ for the white noise have an overall "flat" behavior independent of the index i. Moreover, we see that the coefficients $u_i^T e_{\mathrm{HF}}$ for the HF noise increase with i, reflecting that this noise is dominated by high-frequency components. Similarly, the coefficients $u_i^T e_{\mathrm{LF}}$ for the LF noise decrease with i.

From Figure 4.11 it is evident that the HF noise allows a larger number of SVD components to be included in the regularized solution, because the noise starts to dominate the right-hand side's SVD coefficients $u_i^T b^{\mathrm{exact}} + u_i^T e_{\mathrm{HF}}$ at a later stage than for white noise. On the other hand, for the LF noise, which more resembles a "signal" due to its decreasing SVD coefficients, the noise starts to dominate the SVD coefficients at an earlier stage than for white noise.

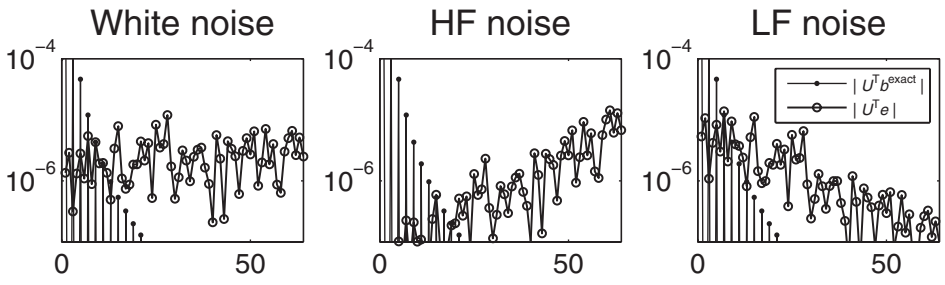

Figure 4.11. *The SVD coefficients for the exact right-hand side (lines with dots) and for the three noise realizations (lines with circles). The different types of noise are clearly revealed in their SVD coefficients.*

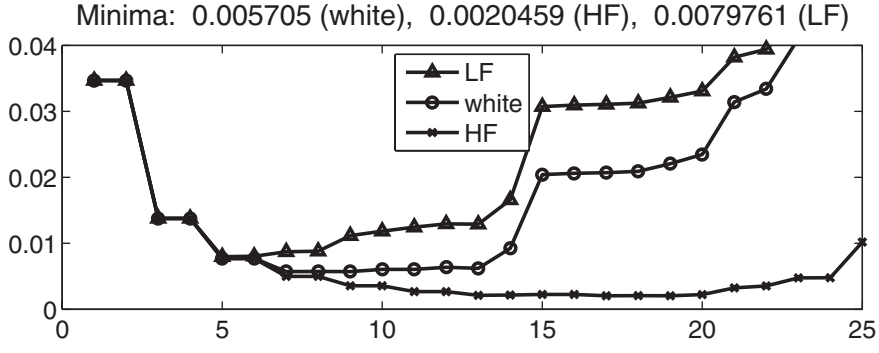

Figure 4.12. *The errors $\|x^{\text{exact}} - x_k\|_2$ in the TSVD solutions to the three problems with different realizations of the noise. For the same noise levels, the HF noise allows the most accurate solution, while the LF noise produces the least accurate solution.*

This is confirmed by Figure 4.12, which shows the errors $\|x^{\text{exact}} - x_k\|_2$ in the TSVD solutions for the three noise realizations. The minima are obtained for $k = 9$, 15, and 5, respectively. Consequently, we can compute a more accurate TSVD solution for the HF noise than for white noise, even though the noise levels are identical. For the LF noise the accuracy of the best TSVD solution is inferior to that for white noise, again with identical noise levels.

We conclude that the more the noise in the data resembles a valid right-hand side (that satisfies the discrete Picard condition)—that is, the more low-frequent the noise—the more severe is the influence of the noise on the obtainable accuracy in the regularized solution.

4.8.2 Good Old Prewhitening

We like white noise! The characterization of white noise is simple and, in principle, it is not too difficult to distinguish between signal (the exact right-hand side) and white

noise, which is important for choosing a good regularization parameter (a topic which we deal with in the next chapter).

If the noise in our data is not white, then we may be able to transform the problem into one with white noise by means of a well-established technique in statistics and signal processing called *prewhitening*; see Section 5.1 in [8]. As always, we assume the model $b = b^{\text{exact}} + e$ for the noisy data, where b^{exact} is the exact data and e is noise with covariance matrix $\text{Cov}(e)$. Now compute the Cholesky factorization of this covariance matrix,

$$\text{Cov}(e) = C^T C, \qquad C = \text{upper triangular},$$

and compute the transformed data

$$b^{\text{pw}} = C^{-T} b = C^{-T} b^{\text{exact}} + C^{-T} e,$$

where C^{-T} denotes the transposed inverse of C. The covariance for the transformed noise is

$$\text{Cov}\big(C^{-T} e\big) = C^{-T}\,\text{Cov}(e)\,C^{-1} = C^{-T} C^T C\, C^{-1} = I,$$

showing that the transformed noise is white. The name "prewhitening" alludes to the preprocessing of the data (by the multiplication with C^{-T}) to make the noise white.

We know that the exact data b^{exact} is dominated by low-frequency components, due to the Picard condition, but what happens to the transformed exact data $C^{-T} b^{\text{exact}}$? If e is HF noise, i.e., it is dominated by high-frequency components, then the prewhitening dampens high frequencies the most (to obtain white nose), and for that reason $C^{-T} b$ will be even more rich in low-frequency components. On the other hand, if e is LF noise (being dominated by low frequencies), then the prewhitening raises the high frequencies in $C^{-T} b$.

In order to use prewhitening for solving discrete inverse problems, we must apply the same transformation to the coefficient matrix A to arrive at a problem to be regularized of the form

$$\big(C^{-T} A\big)\, x = C^{-T} b \qquad \text{or} \qquad \min \big\| \big(C^{-T} A\big)\, x - C^{-T} b \big\|_2 . \tag{4.20}$$

In statistics this is known as generalized least squares, because it is a generalization of the well-known statistical technique of weighting each equation with the inverse of the standard deviations of that equation's noise component. We emphasize that prewhitening cannot improve the ill-posedness of the problem—the transformed problem is just as hard to solve as the original problem. The advantage of prewhitening is to obtain a problem in which the noise is white, making it potentially easier for a parameter-choice method to separate signal and noise in the data and thus select a good regularization parameter.

Figure 4.13 illustrates the use of prewhitening with the `phillips` test problem from *Regularization Tools* (see Section 3.2.2). The left and middle parts show the Picard plots and the errors in the TSVD solutions for the case of white and LF noise, respectively. The LF nature of the noise is clearly visible in the Picard plot where the right-hand side coefficients $|u_i^T b|$ decay for $i > 15$, when these coefficients are dominated by noise. The top right figure shows the Picard plot for the transformed

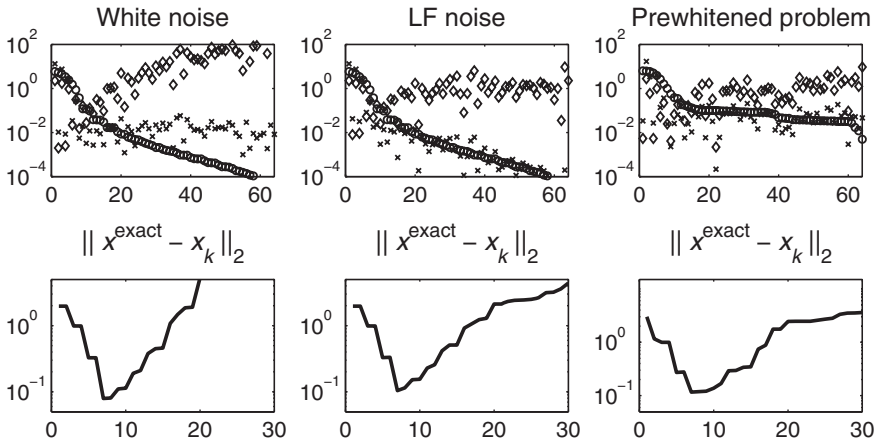

Figure 4.13. *Left and middle: Picard plots and errors in TSVD solutions for the* `phillips` *test problem with white and LF noise, respectively. Right: Ditto for the prewhitened problem* (4.20)*; prewhitening does not improve the obtainable error, but the noise in the transformed problem is white. Circles, crosses, and diamonds denote* σ_i*,* $|u_i^T b|$*, and* $|u_i^T b|/\sigma_i$*, respectively.*

problem $\left(C^{-T}A\right) x = C^{-T} b$; it is obvious that the prewhitening has produced a new problem with white noise, as the coefficients $|u_i^T b|$ are almost constant for $i > 15$. The bottom right figure shows the errors in the TSVD solutions for the transformed problem, and we see that we do not obtain a more accurate solution. The advantage lies in the flat spectrum of the noise in the prewhitened problem, thus allowing us to use parameter-choice methods designed for white noise.

4.9 Rank-Deficient Problems → Different Creatures*

A *rank-deficient problem* involves a mathematical model that, by itself, leads to linear dependence among the rows/columns of the matrix. Such problems are not uncommon in applications such as structural mechanics and electric circuit design, where they appear in connection with differential-algebraic equations (see, e.g., Chapter 6 in [6]). The coefficient matrices in such problems are numerically rank deficient: typically there is a distinct gap between the large and small singular values σ_i, and the size of the small singular values reflects a combination of approximation errors in the model (if any) and rounding errors.

Discrete ill-posed problems, on the other hand, have singular values σ_i that decay gradually to zero, and only the rounding errors prevent them from approaching zero as the matrix dimensions increase. The behavior is due to the nature of the underlying problems, the first-kind integral equations, whose singular values μ_i indeed decay gradually to zero. Hence the condition number of the coefficient matrices in discrete inverse problems is practically infinitely large. We have already seen many numerical examples of this behavior.

Without the important insight about the underlying problems that we obtained in Chapters 2 and 3, one might be lead to the incorrect conclusion that the matrix in a discrete ill-posed problem is numerically rank-deficient. While this is technically true (some rows/columns are indeed almost linear combinations of others), this is *not* a consequence of the underlying mathematical model. Rather, it is a consequence of the "machinery," the floating-point arithmetic system and the finite-precision arithmetic, that we use to solve the problem on our computer.

As discussed in Section 2.4, certain integral operators have a nontrivial null space, e.g., when the kernel is degenerate. Whether this fact is revealed in the SVD of the associated coefficient matrix depends on the discretization method and the decay of the nonzero singular value. Clearly, if a computed singular value σ_i is smaller than, or of the order of, the machine precision times σ_1, then we cannot determine if this σ_i represents a singular value μ_i of the kernel that is genuinely zero, or small but nonzero. Exercises 3.4 and 3.9 show cases where a small singular value indeed reveals the null space, while Exercise 4.9 illustrates a case where some of the small singular values are solely due to approximation and roundoff errors.

When a matrix problem is numerically rank-deficient, then one should solve it by means of a method that is tailored to treat the inherent rank deficiency. Specifically, since there is an infinite number of (least squares) solutions to the problem, one should define how to single out a specific solution, and this choice should reflect the physical model that underlies the problem. For a rank-deficient problem with rank k, the TSVD solution x_k is the unique solution with minimum 2-norm; i.e., if A_k is the TSVD matrix (4.3), then x_k solves the problem

$$\min_x \|x\|_2 \qquad \text{subject to} \qquad \min \|A_k\, x - b\|_2 = \min .$$

The numerical treatment of the more general problem with $\|x\|_2$ replaced by $\|L\, x\|_2$ is discussed in [41]. In practice, under the influence of model/approximation errors and rounding errors, the truncation parameter k is chosen equal to the numerical rank of the matrix. See [32] for more about rank-deficient problems.

We emphasize that while we can use the *same* techniques and numerical methods to solve rank-deficient problems and discrete inverse problems, and while these techniques and methods appear similar, they are in fact very different in that they solve very different problems. Thus, the fact that TSVD can successfully solve a rank-deficient matrix problem does not imply that a discrete inverse problem is rank-deficient if it can also be solved by means of TSVD. To conclude, the main difference between the two types of problems is:

- For rank-deficient problems the choice of the truncation parameter k reflects the numerical rank of the matrix, i.e., the number of singular values larger than the "noise level" for the matrix components.
- For the discrete inverse problems treated in this book, where the singular values decay gradually to zero (and where the concept of numerical rank does not make sense), the choice of the truncation parameter depends on the level of the noise in the data.
- Exceptions are problems with a nontrivial null space, which typically lead to coefficient matrices with both features.

4.10 The Story So Far

After motivating the need for regularization as a means of suppressing the influence of the noise in the data, this chapter introduces two of the most important regularization methods, namely, truncated SVD (TSVD) and Tikhonov regularization. Both methods obtain stability by filtering the contributions to the regularized solution from those components that correspond to the smallest singular values (which are those components that are most sensitive to the noise).

The main justification for the TSVD method is its simplicity, which allows us to derive simple expressions for the properties of the TSVD solution as a function of the truncation parameter. It also motivates the selective SVD (SSVD) method, which incorporates large solution components only (instead of large singular values only). For large-scale problems it is intractable (or impossible) to actually compute the SVD.

Tikhonov's method, on the other hand, requires only the solution of a least squares problem, and it still obtains the desired damping or filtering of the unwanted SVD components (as shown by the SVD analysis of this method). Thus, Tikhonov's method is a very versatile regularization method that has demonstrated its usefulness in a variety of applications for almost half a century.

Next we showed that the solutions computed by the TSVD and Tikhonov methods are indeed less sensitive to perturbations than the naive solution, and we discussed the interplay between the decay of the solution coefficients, the decay of the singular values, and the stability of the solution. Basically, the faster the decay of the solution coefficients, the more successful the regularization methods. This analysis, in turn, provides another motivation for satisfaction of the discrete Picard condition (3.16).

We then introduced the L-curve as a tool for analysis of regularized Tikhonov solutions and their dependence on the regularization parameter, and we demonstrated how the appearance of the L-curve in log-log scale is closely related to how the inverted noise enters the regularized solution. For example, the noise level (for white noise) and the norm of the exact solution can easily be estimated from the L-curve.

The last part of the chapter deals with various practical aspects, such as the influence of nonwhite noise and rank deficiency.

Exercises

4.1. **TSVD Solutions**

Use the function `shaw` from *Regularization Tools* to generate the `shaw` test problem for $n = 60$, and add Gaussian white noise with relative noise level $\texttt{rnl} = 10^{-3}$. Then use the calls

```
[U,s,V] = csvd(A); [X,rho,eta] = tsvd(U,s,V,b,1:15);
```

to compute the TSVD solutions for truncation parameters $k = 1, 2, \ldots, 15$, along with the solution and residual norms. The function `csvd` from *Regularization Tools* computes the "thin SVD" in (3.8) instead of the "full SVD" computed by MATLAB's `svd` function, and it stores the singular values in a vector. The function `tsvd` needs this form of the SVD as input.

Inspect the TSVD solutions x_k, stored as the columns of the matrix `X`; what value of k gives the best approximate solution? What is the norm of the corresponding residual, and can you relate this norm to the norm of the error vector e? Why are the norms of x_k and x^{exact} almost identical for the optimal k?

4.2. **SSVD in Action**

Write a MATLAB function `ssvd` that computes SSVD solutions according to (4.6). Your function should be similar to the function `tsvd` from *Regularization Tools*, except that it should take the threshold τ as input instead of the TSVD truncation parameter k. That is, the function call should take the form

```
[x_tau,rho,eta] = ssvd(U,s,V,b,tau);
```

Test your function on the test problem `deriv2` (which implements the second-derivative test problem from Exercise 2.3) using `example = 3`, problem size $n = 60$, and noise level $\eta = 10^{-8}$. Run a number of tests, and record how often the optimal SSVD solution x_τ is better than the optimal TSVD solution x_k.

4.3. **A Closer Look at the Tikhonov Filter Factors**

The goal is to derive Taylor expansions of the filter factors for the Tikhonov regularization methods. Specifically, determine the two different $O(\cdot)$-parts in the expression

$$\varphi_i^{[\lambda]} = \begin{cases} 1 + O(\cdot), & \sigma_i \gg \lambda, \\ \sigma_i^2/\lambda^2 + O(\cdot), & \sigma_i \ll \lambda. \end{cases}$$

Hint: for $\sigma_i \gg \lambda$ rewrite the filter factor as

$$\varphi_i^{[\lambda]} = \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} = \frac{1}{1 + (\lambda/\sigma_i)^2}$$

and use the Taylor expansion $(1 + \varepsilon)^{-1} = 1 - \varepsilon + O(\varepsilon^2)$. Use the same basic technique for the case $\sigma_i \ll \lambda$.

4.4. **Tikhonov Solutions via SVD**

The purpose of this exercise is to illustrate Tikhonov regularization of the second derivative test problem. Use the `deriv2` function to generate the test problem, and set a $n = 32$. Then use `[U,s,V] = csvd(A)` to compute the SVD of A, and inspect the singular values.

Add a small amount of noise to the right-hand side, e.g.,

```
e = 1e-3*randn(size(b)).
```

This noise is certainly not visible when plotting the right-hand side vector, but it is very significant with respect to the regularization. For a number of different regularization parameters λ in the range 10^{-3} to 1, compute the corresponding filter factors $\varphi_i^{[\lambda]}$ by means of `fil_fac`, as well as the corresponding Tikhonov solution x_λ by means of

```
X = tikhonov(U,s,V,b,lambda).
```

For each λ, plot both the filter factors and the solution, and comment on your results. Use a logarithmic distribution of λ-values, as generated by MATLAB's `logspace` function.

4.5. **From Oversmoothing to Undersmoothing**
The purpose of this exercise is to illustrate how the Tikhonov solution x_λ, its norm $\|x_\lambda\|_2$, and the residual norm $\|Ax_\lambda - b\|_2$ change as λ goes from large values (oversmoothing) to small values (undersmoothing). Use the `shaw` test problem from Exercise 3.6 with $n = 32$, and Gaussian white noise with standard deviation $\eta = 10^{-3}$.

Use `lambda = logspace(1,-5,20)` to generate 20 logarithmically distributed values of λ from 10^{-5} to 10. Then use `csvd` to compute the SVD of A, and use

```
X = tikhonov(U,s,V,b,lambda)
```

to compute the 20 corresponding Tikhnonov solutions x_λ, stored as columns of the matrix `X`. Inspect the columns of the matrix `X` (e.g., by means of `mesh` or `surf`) in order to study the progression of the regularized solution x_λ as λ varies from oversmoothing to undersmoothing.

For each value of λ, compute the solution norm $\|x_\lambda\|_2$ and the residual norm $\|Ax_\lambda - b\|_2$, and plot the corresponding L-curve. Explain how the behavior of the L-curve is related to the behavior of the regularized solutions.

For the given λ-values in `lambda` (or perhaps more values in the same interval), compute the error norm $\|x^{\text{exact}} - x_\lambda\|_2$ and plot it versus λ. Determine the optimum value of λ—the one that leads to the smallest error norm—and locate the position of the corresponding solution on the L-curve. Is it near the "corner"?

4.6. **The L-Curve**
This exercise illustrates the typical behavior of the L-curve for a discrete ill-posed problem, using the second-derivative test problem from Exercise 2.3. Generate the test problem `deriv2` with $n = 64$, and add Gaussian white noise scaled such that $\|e\|_2/\|b^{\text{exact}}\|_2 = 10^{-2}$. Then use `l_curve` from *Regularization Tools* to plot the L-curves corresponding to the three different right-hand sides b^{exact}, e, and $b = b^{\text{exact}} + e$. Try to "zoom in" on the area near the corner.

What happens to the corner if you switch to lin-lin scale?

Switch back to log-log scale and add a horizontal line at $\|x^{\text{exact}}\|_2$, the norm of the exact solution, and a vertical line at $\|e\|_2$, the norm of the perturbation. Relate the positions of these lines to the different parts of the L-curve.

Find (by trial and error) a Tikhonov regularization parameter λ^* that approximately minimizes the error $\|x^{\text{exact}} - x_\lambda\|_2$ between the exact solution x^{exact} and the regularized solution x_λ. Add the point $(\,\|Ax_{\lambda^*} - b\|_2\,,\,\|x_{\lambda^*}\|_2\,)$ to the L-curve (it must lie on the L-curve corresponding to b). Is it near the corner?

4.7. **Limitations of TSVD and Tikhonov Methods**
This exercise illustrates one of the limitations of TSVD and Tikhonov solutions, namely, that they are not so well suited for computing regularized solutions

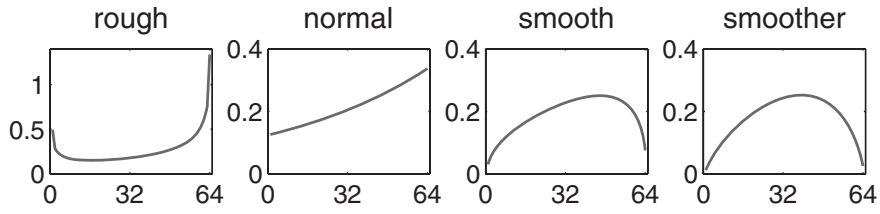

Figure 4.14. *Solutions with increasing smoothness.*

when the exact solution is discontinuous. We use the model problem `wing` from *Regularization Tools*, whose solution has two discontinuities. Since we are mainly interested in the approximation properties of the TSVD and Tikhonov solutions, we do not add any noise in this exercise.

Generate the model problem using `wing`, plot the exact solution, and notice its form. Compute TSVD and Tikhonov solutions for various regularization parameters. Monitor the solutions and try to find the "best" value of k and λ. Notice how difficult it is to reconstruct the discontinuities.

4.8. **The Smoothness of the Solution***

This exercise illustrates the influence of the solution's smoothness on the accuracy of the regularized solution. Here, smoothness is associated with the decay of the exact solution's SVD coefficients; the faster the decay, the smoother the solution, and the fewer SVD components are needed to approximate the solution well.

First use the test problem `deriv2` to generate a test problem with solution x; we recommend using the call `deriv2(n,2)`, but you are also encouraged to try the other two solutions (with the second parameter set to 1 and 3). Then compute the SVD of A and define three new solutions x_r, x_s, and x_t whose SVD coefficients are given by

$$\left.\begin{array}{rcl} v_i^T x_r &=& (s_i/s_1)^{-1/5}\, v_i^T x \\ v_i^T x_s &=& (s_i/s_1)^{1/4}\, v_i^T x \\ v_i^T x_t &=& (s_i/s_1)^{1/2}\, v_i^T x \end{array}\right\} \qquad i = 1, \ldots, n.$$

With these definitions, x_r is "rougher" than x (its SVD coefficients decay more slowly), while x_s and x_t are smoother than x (their coefficients decay more quickly). The four solutions are shown in Figure 4.14. Then compute the corresponding right-hand side by multiplying with A, and add white Gaussian noise with standard deviation $\eta = 10^{-4}$. Study the Picard plots, and check that the SVD coefficients behave as expected.

For all four problems, compute the TSVD solutions for $k = 1, \ldots, n$, and determine the truncation parameters for which the error in the TSVD solution is smallest. How does this truncation parameter, as well as the corresponding error, depend on the smoothness of the solution?

4.9. **SVD Analysis of the Inverse Heat Problem***

Due to the form of the kernel, the approximations in the discretization method,

and the finite-precision arithmetic, the null space of the matrix A in the discretized problem may not always reflect the annihilator of the operator.

This exercise illustrates this phenomenon, using a discretization of the inverse heat equation problem. The underlying ill-posed problem is a Volterra integral equation of the first kind, of the form

$$\int_0^s k(s-t)\, f(t)\, dt = g(s), \qquad 0 \le s \le 1,$$

with a convolution kernel k given by

$$k(\tau) = \frac{\tau^{-3/2}}{2\,\kappa\,\sqrt{\pi}}\, \exp\left(-\frac{1}{4\,\kappa^2\,\tau^2}\right).$$

The annihilator for this operator is the delta function located at $t = 1$, i.e.,

$$\int_0^s k(s-t)\, \delta(t-1)\, dt = 0.$$

Discretization of this problem by means of the midpoint rule leads to an ill-conditioned lower triangular matrix. This discretization is implemented in the function `heat` in *Regularization Tools*. For $n = 64$, use `heat` to compute the matrix A using the default value $\kappa = 1$. Then compute the SVD of A, and check that the matrix is rank-deficient. What is the dimension of the null space?

Plot the right singular vectors that span the null space of A, and comment on their resemblance to the annihilator, i.e., the delta function $\delta(t-1)$.

Now repeat the computations with the larger value $\kappa = 4$. What is the rank and null space of A now?

Chapter 5

Getting Serious: Choosing the Regularization Parameter

At this stage we have at our disposal several regularization methods, based on filtering of the SVD components. We have also seen that for small problems it is often fairly straightforward to "eyeball" a good TSVD truncation parameter from the Picard plot.

The main thing that we are still missing is a reliable and automated technique for choosing the regularization parameter, such as k (for TSVD) or λ (for Tikhonov). Specifically, what we would like is an efficient, robust, and reliable method for computing the regularization parameter from the given data, which does not require the computation of the SVD (which is infeasible for large problems) or any human inspection of a plot.

Unfortunately, such a method has yet to be found; at the present time, no one has devised a general-purpose parameter-choice algorithm which will always produce a good k or λ, such that the regularized solution is a good approximation to the unattainable exact solution. What we currently have at our disposal is a collection of methods which, under certain assumptions, tend to work well; but all of them can and will occasionally fail to produce good results.

We do not intend to cover all parameter-choice methods developed over the years; instead we focus on four different methods, each with its own background and motivation. We start with a discussion of the errors in the regularized solution, which serves as a motivation for the parameter-choice methods, and which naturally leads to the first two methods, namely, the discrepancy principle and the L-curve criterion. Next we describe two methods, generalized cross validation and the NCP method, both of which are rooted in a statistical description of the problem. We finish with a comparison of all the methods.

More technical details and references for the first three methods can be found in [32]. It appears to be the first time the fourth method is presented in book form. As we already mentioned, there are many more parameter-choice methods around, based on a variety of principles, error estimates, and heuristics, and new methods are being developed as you read this book.

In this book we are not concerned with asymptotic results about the behavior of the methods or the solutions as the problem dimensions go to infinity (and the

discretization errors go to zero). A careful analysis of these aspects can be found in [74, Chapter 7].

5.1 Regularization Errors and Perturbation Errors

Any method for choosing the regularization parameter should seek to minimize the errors in the regularized solution, and it is therefore instructive to take a closer look at these errors. The outset for our discussion is the Tikhonov solution, but we will also discuss the TSVD solution. Define the diagonal matrix $\Phi^{[\lambda]}$ consisting of the Tikhonov filter factors,

$$\Phi^{[\lambda]} = \begin{pmatrix} \varphi_1^{[\lambda]} & & \\ & \ddots & \\ & & \varphi_n^{[\lambda]} \end{pmatrix},$$

where $\varphi_i^{[\lambda]} = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ (for TSVD this matrix has diagonal elements 1 and 0). Then we can write the regularized solution x_λ as

$$x_\lambda = V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T b. \tag{5.1}$$

A similar expression holds for the TSVD solution with $\Phi^{[\lambda]}$ replaced by the filter matrix $\Phi^{[k]} = \mathrm{diag}(1,\ldots,1,0,\ldots,0)$.

Now we remind ourselves that the given right-hand side consists of an exact "signal" plus additive noise: $b = A\,x^{\text{exact}} + e$. It follows that the error in the Tikhonov regularized solution is given by

$$\begin{aligned} x^{\text{exact}} - x_\lambda &= x^{\text{exact}} - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T b \\ &= x^{\text{exact}} - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T A\,x^{\text{exact}} - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T e \\ &= \left(I - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T U\,\Sigma\,V^T\right) x^{\text{exact}} - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T e \\ &= V\left(I - \Phi^{[\lambda]}\right) V^T x^{\text{exact}} - V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T e. \end{aligned} \tag{5.2}$$

The first term in the above expression is the *regularization error*, coming from the introduction of the filtering:

$$\Delta x_{\text{bias}} = V\left(I - \Phi^{[\lambda]}\right) V^T x^{\text{exact}} = \sum_{i=1}^{n} \left(1 - \varphi_i^{[\lambda]}\right) \left(v_i^T x^{\text{exact}}\right) v_i, \tag{5.3}$$

and we recognize this as (minus) the bias term introduced in the previous chapter. It expresses the deviation of the expected value of x_λ from the exact solution.

The second error term is the *perturbation error*, coming from inverting and filtering the noise component in the data:

$$\Delta x_{\text{pert}} = V\,\Phi^{[\lambda]}\,\Sigma^{-1}\,U^T e. \tag{5.4}$$

The main purpose of the regularization via spectral filtering, in the presence of the filter matrix $\Phi^{[\lambda]}$, is to prevent this perturbation error from blowing up in magnitude and spoiling the solution.

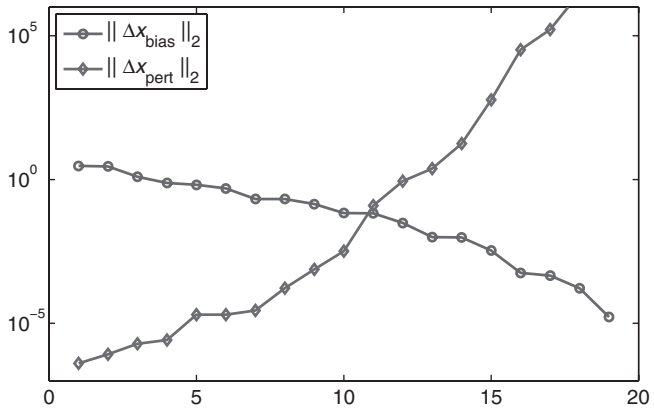

Figure 5.1. *The norm of the regularization and perturbation error for TSVD as a function of the truncation parameter k. The two different errors approximately balance each other for $k = 11$.*

For TSVD solutions, the regularization and perturbation errors take the form

$$\Delta x_{\text{bias}} = \sum_{i=k+1}^{n} (v_i^T x^{\text{exact}})\, v_i, \qquad \Delta x_{\text{pert}} = \sum_{i=1}^{k} \frac{u_i^T e}{\sigma_i}\, v_i.$$

We see that we can use the truncation parameter k to prevent the perturbation error from blowing up, due to the division by the small singular values, at the cost of introducing bias in the regularized solution.

Both perturbation errors and regularization errors are always present in the regularized solution, and their size depends on the regularization parameter. If k is close to n, then most SVD components are included and the bias is small, but the perturbation error is large because we apply very little filtering. On the other hand, if k is small, then many SVD components are left out and thus the perturbation error will also be small, but then the regularization error (the bias) will be large. We can therefore say that the goal of choosing k is to balance the size of two error terms Δx_{bias} and Δx_{pert}, as illustrated in Figure 5.1.

Precisely the same is true, of course, for the choice of the Tikhonov regularization parameter λ. If λ is very small, then all filter factors $\varphi_i^{[\lambda]}$ are close to 1; hence $\Phi^{[\lambda]}$ is close to I, and the regularization error is small (while the perturbation error is large). And if λ is large, then many filter factors are small, and thus the perturbation error will also be small, but then $\Phi^{[\lambda]}$ is not close to the identity matrix, and the regularization error (the bias) will be large.

Let us now perform a careful analysis of the behavior of the regularized solution, as the regularization parameter changes. To simplify the presentation we restrict our analysis to the TSVD method, and we have already seen that as k increases, the characterization of the solution goes from oversmoothing to undersmoothing. We shall see that this behavior is closely related to the behavior of the regularization and perturbation errors.

As always, we assume that the discrete Picard condition is satisfied. Hence the noisy right-hand side's SVD coefficients $u_i^T b$ will decay in magnitude, on the average, until they level off at a plateau determined by the noise's standard deviation η. See also (3.21). Let k_η denote the index that marks the transition between decaying and flat coefficients $|u_i^T b|$. Due to the discrete Picard condition, the coefficients $|u_i^T b|/\sigma_i$ will also decay, on the average, for all $i < k_\eta$.

We now use this behavior in the analysis of the solution norm, using the relation (4.16) to obtain rough approximations. We must distinguish between the two cases $k < k_\eta$ and $k > k_\eta$:

$$k < k_\eta: \quad \|x_k\|_2^2 \approx \sum_{i=1}^{k} \left(\frac{u_i^T b^{\text{exact}}}{\sigma_i} \right)^2,$$

$$k > k_\eta: \quad \|x_k\|_2^2 \approx \sum_{i=1}^{k_\eta} \left(\frac{u_i^T b^{\text{exact}}}{\sigma_i} \right)^2 + \sum_{i=k_\eta+1}^{k} \left(\frac{\eta}{\sigma_i} \right)^2 \approx \|x^{\text{exact}}\|_2^2 + \eta^2 \sum_{i=k_\eta+1}^{k} \sigma_i^{-2}.$$

We see that for $k < k_\eta$ (oversmoothing), the solution norm increases slowly with k, while for $k > k_\eta$ (undersmoothing), it increases in a much more dramatic way. The index $k = k_\eta$ marks the transition between the two types of behavior.

In a similar fashion we can analyze the residual norm, now using the expression (4.17) to obtain rough approximations, and again distinguishing between the two cases for k:

$$k < k_\eta: \quad \|A x_k - b\|_2^2 \approx \sum_{i=k+1}^{k_\eta} (u_i^T b)^2 + (n - k_\eta)\eta^2 \approx \sum_{i=k+1}^{k_\eta} (u_i^T b^{\text{exact}})^2,$$

$$k > k_\eta: \quad \|A x_k - b\|_2^2 \approx (n - k)\eta^2.$$

We conclude that for $k < k_\eta$ (oversmoothing), the residual norm decreases steadily with k, while for $k > k_\eta$ (undersmoothing), it decreases much more slowly. Again the transition between the two types of behavior occurs at $k = k_\eta$ when the regularization and perturbation errors are balanced.

Figure 5.2 illustrates the quality of the above approximations to the norms $\|x_k\|_2$ and $\|A x_k - b\|_2$ for the `shaw` test problem (see Exercise 3.6) with $n = 20$ and $\eta = 10^{-6}$.

We conclude that a good value of the truncation parameter is obviously $k = k_\eta$, for which we extract all the SVD coefficients for which the "signal" dominates, i.e., for which $|u_i^T b^{\text{exact}}| > \eta$. If we want to compute an approximation to this k_η, then we should locate the transition point in the solution norm and the residual norm.

Unfortunately, the automatic determination of such a transition point—and thus of the regularization parameter—is not an easy task. Many factors (such as problem scaling and the actual decay rates of the singular values and the coefficients $u_i^T b$) must be taken into account. For the same reason, as we illustrate in Exercise 5.4, it is not possible to design an a priori parameter-choice method.

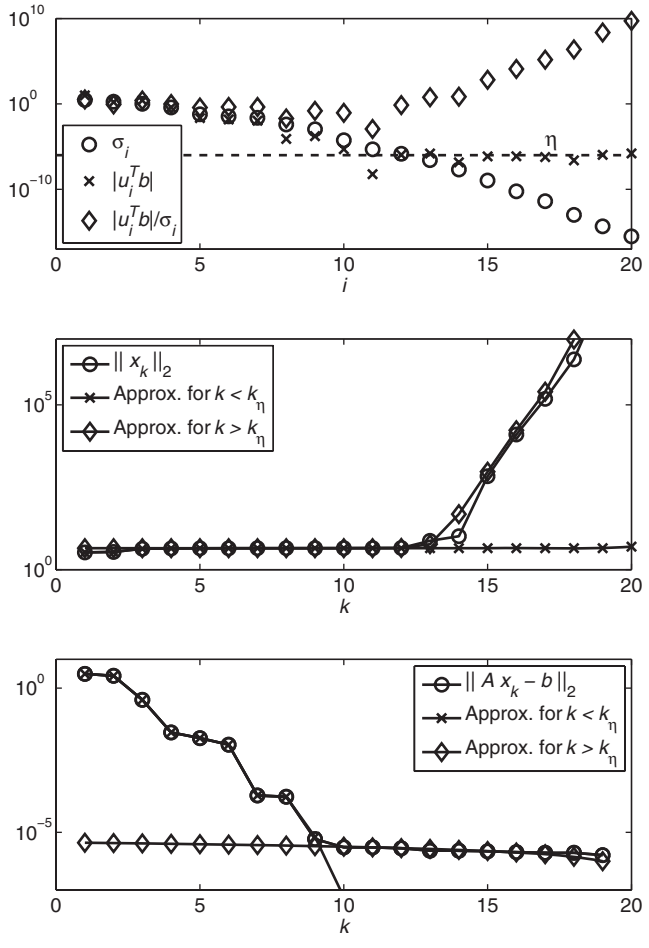

Figure 5.2. *Top: Picard plot for the* `shaw` *test problem with* $n = 20$ *and* $\eta = 10^{-6}$. *Middle: The TSVD solution norm* $\|x_k\|_2$ *and the two approximations valid for* $k < k_\eta$ *and* $k > k_\eta$. *Bottom: The corresponding residual norm* $\|A\,x_k - b\|_2$ *and its approximations.*

5.2 Simplicity: The Discrepancy Principle

Instead of locating the truncation parameter $k = k_\eta$ via a study of the right-hand side's SVD coefficients, we can choose k such that the residual norm is equal to $(n - k_\eta)^{1/2}\eta$ (ideally, this happens exactly at $k = k_\eta$). There are two drawbacks, however: k_η appears explicitly in the formula, and even if we happen to have an estimate of $(n-k_\eta)^{1/2}\eta$, then this criterion for choosing k is very sensitive to the quality of the estimate. Instead one uses a slightly larger value which is easier to obtain, namely, the expected value of the perturbation norm $\|e\|_2 = n^{1/2}\eta$. Figure 5.3 illustrates these

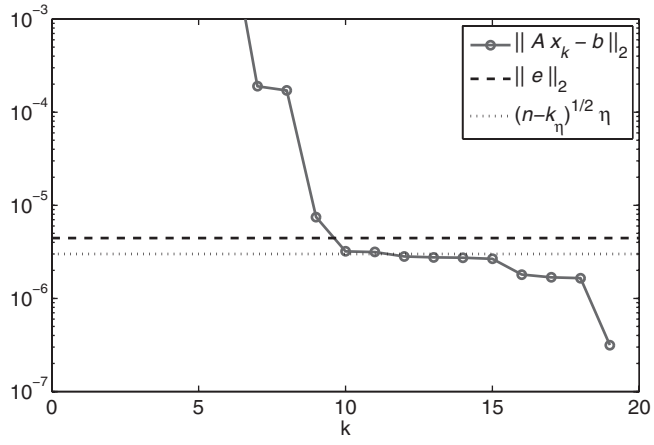

Figure 5.3. *Illustration of the discrepancy principle. The choice* $\|A\,x_k - b\|_2 \approx (n - k_\eta)^{1/2}\eta$ *leads to a too large value of the truncation parameter* k*, while the more conservative choice* $\|A\,x_k - b\|_2 \approx \|e\|_2$ *leads to a better value of* k*.*

two choices. An even safer approach is to include a "safety factor" ν_{dp} equal to, say, 2, and use the threshold $\nu_{\text{dp}}\|e\|_2$ for the residual norm.

The resulting method is called the *discrepancy principle*. It says that we should choose the truncation parameter k such that the residual norm equals the "discrepancy" in the data, as measured by $\nu_{\text{dp}}\|e\|_2$. It is rarely the case that the relation can be satisfied exactly, so instead we choose the *largest* k such that $\|A\,x_k - b\|_2 \geq \nu_{\text{dp}}\|e\|_2$:

$$\boxed{\text{Choose } k = k_{\text{DP}} \text{ such that } \|A\,x_k - b\|_2 \geq \nu_{\text{dp}}\|e\|_2 > \|A\,x_{k+1} - b\|_2.} \tag{5.5}$$

The same strategy applies, of course, to the Tikhonov method with x_k replaced by x_λ, and here equality can be obtained:

$$\boxed{\text{Choose } \lambda = \lambda_{\text{DP}} \text{ such that } \|A\,x_\lambda - b\|_2 = \nu_{\text{dp}}\|e\|_2.} \tag{5.6}$$

In both cases there is a unique solution because the residual norm varies monotonically with k and λ.

Due to its simplicity, the discrepancy principle is often the favored parameter-choice method in theoretical studies of regularization methods, e.g., when proving that a regularized solution converges to the exact solution as $\|e\|_2 \to 0$.

The main computational advantage of this parameter-choice method is also its simplicity: it requires only an efficient root finder for the problem in (5.6). For Tikhonov regularization, the derivative of $\|A\,x_\lambda - b\|_2$ with respect to λ can be computed from the relation $\rho' = -\lambda^2\,\xi'$ derived in Section 4.7, together with the technique for computing ξ' that we describe in the next section.

The main disadvantage of the discrepancy principle is that we often do not know $\|e\|_2$ (or η) exactly, and therefore we must use an estimate of the norm (or the standard deviation). But unfortunately the quality of the computed regularization

parameter k_{DP} or λ_{DP} is very sensitive to the accuracy of the estimate for $\|e\|_2$. In particular, a too small estimate can lead to dramatic undersmoothing (because k is chosen too large, and λ is too small).

In terms of the L-curve from Figure 4.10, the regularization parameter λ_{DP} chosen by the discrepancy principle corresponds to that point where the L-curve that intersects a vertical line located at $\|A\,x_\lambda - b\|_2 = \nu_{\mathrm{dp}}\|e\|_2$. Now recall that the L-curve's almost vertical part is actually located at $\|A\,x_\lambda - b\|_2 \approx \|e\|_2$; clearly an underestimate of the error norm in the discrepancy principle will force the intersection of the L-curve and the vertical line to a point with a large solution norm (i.e., a highly undersmoothed solution). Even a small underestimate of the error norm can have a large impact on the regularization parameter, making this a somewhat risky parameter-choice method for real data. We return to examples of this issue in Section 5.6.

Several variants of the discrepancy principle are discussed in [32, Section 7.2]; none of these variants seem to be used much in practice.

5.3 The Intuitive L-Curve Criterion

The overall behavior of the L-curve from Section 4.7 is closely related to the size of the regularization error Δx_{bias} and the perturbation error Δx_{pert}. Therefore, one can learn a lot about the solution's dependence on the regularization error by studying the L-curve. Here, for simplicity, we consider the TSVD method; recall that k_η denotes the index at the transition between decaying and flat $|u_i^T b|$-coefficients.

- When k is smaller than k_η, then the regularization error dominates in x_k, and thus the overall behavior of its norm is largely determined by how many (or few) SVD components are effectively included. The fewer the components, the smaller the solution norm $\|x_k\|_2$ and the larger the residual norm $\|A\,x_k - b\|_2$. Using the analysis from the previous subsection (see also Figure 5.2), we conclude that these norms depend roughly on k as follows:
  - The solution norm $\|x_k\|_2$ is almost a constant given by $\|x^{\mathrm{exact}}\|_2$ except for very small k where $\|x_k\|_2$ gets smaller as $k \to 0$.
  - The residual norm $\|A\,x_k - b\|_2$ increases as $k \to 0$, until it reaches its maximum value $\|b\|_2$ for $k = 0$.
- When k is larger than k_η, then the perturbation error dominates x_k, and the overall behavior of its norm is determined by this component. Now the norms depend on k in the following way:
  - The solution norm $\|x_k\|_2$ increases (sometimes dramatically) as k increases, because more and more and increasingly larger error components are included.
  - The residual norm $\|A\,x_k - b\|_2$, on the other hand, stays almost constant at the approximate value $\|e\|_2$ (except when $k \approx n$).

From this analysis it follows that the L-curve must have two distinctly different parts—one part where it is quite flat (when the regularization error dominates), and one part

that is more vertical (when the perturbation error dominates). The log-log scale for the L-curve emphasizes the different characteristics of these two parts.

One important observation here is that the L-curve must also include a part with the transition between the vertical and horizontal parts. Very often this transition takes place in a small region, and then the L-curve will have a distinct corner between the two parts. Moreover, this transition region is associated with those values of λ or k for which the dominating error component changes between the perturbation error Δx_{bias} and the regularization error Δx_{pert}.

The key idea in the *L-curve criterion* is therefore to choose a value of λ that corresponds to the L-curve's corner, in the hope that this value will provide a good balance of the regularization and perturbation errors. We emphasize that this line of arguments is entirely heuristic; but it is supported with some analysis. It is for this reason that we allow ourselves to label the L-curve criterion as "intuitive."

We still need an operational definition of the corner of the L-curve, such that we can compute it automatically. For Tikhonov regularization, a natural definition of the corner is the point on the L-curve with maximum curvature. We emphasize that this curvature must be measured in the log-log system; i.e., we must compute the curvature $\hat{c}$ of the curve $(\log \|A x_\lambda - b\|_2 , \log \|x_\lambda\|_2)$.

At this stage it is most convenient to consider the L-curve for Tikhonov regularization. Following the analysis in Section 4.7, we introduce the quantities

$$\hat{\xi} = \log \|x_\lambda\|_2^2 \qquad \text{and} \qquad \hat{\rho} = \log \|A x_\lambda - b\|_2^2$$

such that the L-curve is given by $(\frac{1}{2}\hat{\rho} , \frac{1}{2}\hat{\xi})$. If $'$ denotes differentiation with respect to λ, then it follows that

$$\hat{\xi}' = \frac{\xi'}{\xi} \qquad \text{and} \qquad \hat{\rho}' = \frac{\rho'}{\rho}.$$

When we use these relations, then it follows immediately that the second derivatives of $\hat{\xi}$ and $\hat{\rho}$ are given by

$$\hat{\xi}'' = \frac{\xi''\xi - (\xi')^2}{\xi^2} \qquad \text{and} \qquad \hat{\rho}'' = \frac{\rho''\rho - (\rho')^2}{\rho^2}.$$

When we insert these relations into the definition of the curvature,

$$\hat{c}_\lambda = 2 \, \frac{\hat{\rho}' \, \hat{\xi}'' - \hat{\rho}'' \, \hat{\xi}'}{\left((\hat{\rho}')^2 + (\hat{\xi}')^2 \right)^{3/2}},$$

it turns out that we can express the curvature of the log-log L-curve entirely in terms of the three quantities ξ, ρ, and ξ':

$$\hat{c}_\lambda = 2 \, \frac{\xi \, \rho}{\xi'} \, \frac{\lambda^2 \xi' \rho + 2 \, \lambda \, \xi \, \rho + \lambda^4 \xi \, \xi'}{(\lambda^2 \xi^2 + \rho^2)^{3/2}} \ .$$

The two quantities $\xi = \|x_\lambda\|_2^2$ and $\rho = \|A x_\lambda - b\|_2^2$ are often easy to compute, so it remains to demonstrate how to compute ξ' efficiently. Using the SVD it is straightforward to verify that this quantity is given by

$$\xi' = \frac{4}{\lambda} \, x_\lambda^T z_\lambda,$$

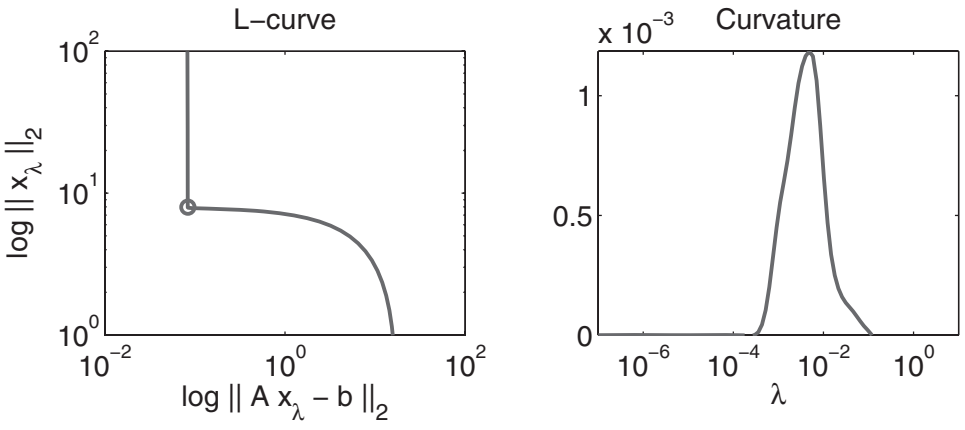

Figure 5.4. *An L-curve and the corresponding curvature $\hat{c}_\lambda$ as a function of λ. The corner, which corresponds to the point with maximum curvature, is marked by the circle; it occurs for $\lambda_L = 4.86 \cdot 10^{-3}$.*

where the vector z_λ is given by

$$z_\lambda = (A^T A + \lambda^2 I)^{-1} A^T (A x_\lambda - b).$$

Recalling the formulas in Section 4.4 for the Tikhonov solution, we see immediately that z_λ is the solution to the Tikhonov least squares problem

$$\min_z \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} z - \begin{pmatrix} A x_\lambda - b \\ 0 \end{pmatrix} \right\|_2 .$$

Therefore, this vector is easy to compute by solving a new Tikhonov problem with the same coefficient matrix and whose right-hand side is the residual vector $A x_\lambda - b$.

The conclusion of this analysis is that for each value of λ, we can compute the curvature $\hat{c}_\lambda$ with little overhead, and thus we can use our favorite optimization routine to find the value λ_L that maximizes $\hat{c}_\lambda$:

Choose $\lambda = \lambda_L$ such that the curvature $\hat{c}_\lambda$ is maximum.	(5.7)

Figure 5.4 shows an L-curve and the corresponding curvature $\hat{c}_\lambda$ as a function of λ.

The above analysis is not immediately valid for the TSVD method, because this method produces a finite (small) set of solutions for $k = 1, 2, \ldots$, and thus the corresponding "L-curve" really consists of a finite set of points. However, the analysis and the arguments do carry over to this case, and it still often makes good sense to locate the TSVD solution at the overall corner of the discrete L-curve:

Choose $k = k_L$ at the *overall corner* of the discrete L-curve.	(5.8)

How to actually compute this corner is not as straightforward as it may seem, because a discrete L-curve can have many small local corners; we refer to the reader [9] and [37] for recent algorithms (*Regularization Tools* uses the latter one).

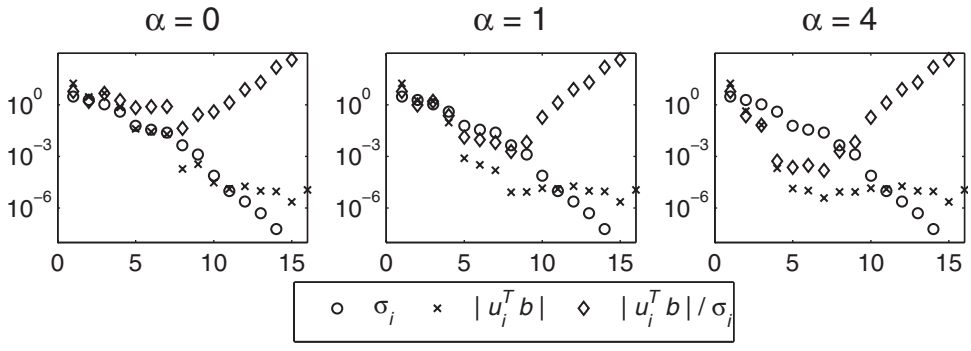

Figure 5.5. *Picard plots for the smooth exact solutions generated by Eq.* (5.9)*, where* x^{exact} *is from the* `shaw` *test problem, and* α *controls the smoothness (the larger the* α*, the faster the decay of the coefficients* $|v_i^T x^{\text{exact},\alpha}|$*. For all three values of* α *the L-curve criterion gives the truncation parameter* $k_{\text{L}} = 10$*. While this is a good choice for* $\alpha = 0$*, the smallest error in the TSVD solution is obtained for* $k = 8$ *for* $\alpha = 1$*, and* $k = 6$ *for* $\alpha = 4$*.*

We conclude this discussion of the L-curve criterion with a few words of caution. The criterion is essentially a (good) heuristic, and there is no guarantee that it will always produce a good regularization parameter, i.e., that λ_{L} produces a solution in which the regularization and perturbation errors are balanced.

One instance where the L-curve criterion is likely to fail is when the SVD components $v_i^T x^{\text{exact}}$ of the exact solution decay quickly to zero (in which case x^{exact} will appear very smooth, because it is dominated by the first few—and smooth—right singular vectors v_i). The L-curve's corner is located at the point where the solution norm becomes dominated by the noisy SVD components. Often this is precisely where the components start to increase dramatically; but for very smooth solutions this happens for larger k when we have included too many noisy components, and hence λ_{L} will lead to an undersmoothed solution.

Figure 5.5 illustrates this deficiency of the L-curve criterion with artificially created smooth solutions generated by the model

$$x^{\text{exact},\alpha} = \|A\|_2^{-\alpha} (A^T A)^{\alpha/2} \, x^{\text{exact}} = V \, (\Sigma/\sigma_1)^{\alpha} \, V^T x^{\text{exact}}, \tag{5.9}$$

such that the ith SVD component is damped by the factor $(\sigma_i/\sigma_1)^{\alpha}$. The larger the α, the smoother the solution. For all three values of α used in Figure 5.5, the L-curve chooses $k_{\text{L}} = 10$, while for $\alpha = 4$ (the smoothest solution), the optimal choice is actually $k = 6$.

Another instance where the L-curve criterion can fail is when the change in the residual and solution norms is small for two consecutive values of k. Again, in this case, it is necessary to include many noisy SVD coefficients in the solution before its norm has increased so much that the corner appears on the L-curve. Unfortunately this behavior of the SVD coefficients is often observed for large-scale problems, and we illustrate this in Exercise 6.4.

One final critical remark on the behavior of the L-curve criterion: If we let the noise go to zero, then it has been shown that the regularization parameter λ_L tends to diverge from the optimal one, and hence x_{λ_L} may not converge to x^{exact} as $\|e\|_2 \to 0$. This undesired nonconverging behavior is not shared by the discrepancy principle (or the generalized cross validation (GCV) method discussed in the next section). Fortunately, noise that goes to zero is a rare situation in practice; usually we work with fixed data with a fixed noise level.

5.4 The Statistician's Choice—Generalized Cross Validation

The third method discussed here represents a somewhat different approach to choosing the regularization parameter, in that the goal here is to find the value of λ or k such that $A\,x_\lambda$ or $A\,x_k$ predicts the *exact* data b^{exact} as well as possible.

It is easiest to carry out the derivation for TSVD, for which the filter matrix is $\Phi^{[k]} = \text{diag}(1,\ldots,1,0,\ldots,0)$. Now consider the difference between b^{exact} and the predictor $A\,x_k$:

$$\begin{aligned} A\,x_k - b^{\text{exact}} &= A\,V\,\Phi^{[k]}\,\Sigma^{-1}U^T(b^{\text{exact}} + e) - b^{\text{exact}} \\ &= U\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} U^T b^{\text{exact}} + U\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} U^T e - U\,U^T b^{\text{exact}} \\ &= U\begin{pmatrix} I_k & 0 \\ 0 & 0 \end{pmatrix} U^T e - U\begin{pmatrix} 0 & 0 \\ 0 & I_{n-k} \end{pmatrix} U^T b^{\text{exact}}. \end{aligned}$$

Therefore, using again the SVD analysis, the norm of the prediction error satisfies

$$\|A\,x_k - b^{\text{exact}}\|_2^2 = \sum_{i=1}^{k}(u_i^T e)^2 + \sum_{i=k+1}^{n}(u_i^T b^{\text{exact}})^2 \approx k\,\eta^2 + \sum_{i=k+1}^{n}(u_i^T b^{\text{exact}})^2.$$

Once again, let us split the analysis into two cases, depending on k. Recalling that $|u_i^T b^{\text{exact}}| < \eta$ for $i > k_\eta$, we obtain

$$\begin{aligned} k < k_\eta: &\quad \|A\,x_k - b^{\text{exact}}\|_2^2 \approx k\,\eta^2 + \sum_{i=k+1}^{k_\eta}(u_i^T b^{\text{exact}})^2, \\ k > k_\eta: &\quad \|A\,x_k - b^{\text{exact}}\|_2^2 \approx k\,\eta^2. \end{aligned}$$

This analysis shows that for $k < k_\eta$ the norm of the prediction error decreases with k (because we replace the term $u_k^T b^{\text{exact}}$ with η—which is smaller—when going from k to $k+1$), while for $k > k_\eta$ the norm increases with k. The minimum arises near the transition, i.e., for $k \approx k_\eta$. Hence it makes good sense to search for the regularization parameter that minimizes the prediction error.

How can we estimate this parameter for a given problem, where b^{exact} is not available? Cross validation is a classical statistical technique that comes into good

use here. In cross validation, one separates the given data—here, the elements of our right-hand side—into two sets and uses one of the sets to compute a solution which is then used to predict the elements in the other set. For example, we could leave out b_i, the ith element of b, and then compute the Tikhonov solution based on the reduced problem,

$$x_\lambda^{(i)} = \left((A^{(i)})^T A^{(i)} + \lambda^2 I_{n-1} \right)^{-1} (A^{(i)})^T b^{(i)},$$

where $A^{(i)}$ and $b^{(i)}$ are the shortened versions of A and b with the ith row and element, respectively, left out. Then we can use $x_\lambda^{(i)}$ to predict the element b_i that was left out via the "missing" row of A, through the expression $A(i,:)\, x_\lambda^{(i)}$. The goal is then to choose the regularization parameter λ such that it minimizes the prediction errors for all data elements:

$$\min_\lambda \ \frac{1}{m} \sum_{i=1}^{m} \left(A(i,:)\, x_\lambda^{(i)} - b_i \right)^2.$$

This seems like a formidable computation task because m different Tikhonov problems are involved. However, using some technical arguments that we leave out, one can show that the above minimization problem can be replaced by

$$\min_\lambda \ \frac{1}{m} \sum_{i=1}^{m} \left(\frac{A(i,:)\, x_\lambda - b_i}{1 - h_{ii}} \right)^2,$$

where x_λ is the Tikhonov solution, and h_{ii} are the diagonal elements of the matrix $A\,(A^TA + \lambda^2 I)^{-1} A^T$.

This minimization problem is easier to work with, because only one Tikhonov problem is involved. Unfortunately, the diagonal elements h_{ii} will change if we permute the rows of A, and thus the solution depends on the particular ordering of the data, i.e., the elements in b. This is clearly a disadvantage in problems where there is no natural ordering of these elements.

The method of *generalized cross validation* (GCV) was introduced to remedy this inconvenience, by replacing each diagonal element h_{ii} with the average of the diagonal elements. Thus, the simplified minimization problem takes the form

$$\min_\lambda \ \frac{1}{m} \sum_{i=1}^{m} \left(\frac{A(i,:)\, x_\lambda - b_i}{1 - \mathrm{trace}\,(A\,(A^TA + \lambda^2 I)^{-1} A^T)\,/m} \right)^2,$$

and we recognize the sum as the squared residual norm $\|A\,x_\lambda - b\|_2^2$ divided by $\left(1 - \mathrm{trace}\left(A\,(A^TA + \lambda^2 I)^{-1} A^T\right)/m\right)^2$. If we insert the SVD of A, then it is easy to show that the trace term takes the form

$$\mathrm{trace}\left(A\,(A^TA + \lambda^2 I)^{-1} A^T\right) = \mathrm{trace}(U\,\Phi^{[\lambda]} U^T) = \mathrm{trace}(\Phi^{[\lambda]}) = \textstyle\sum_{i=1}^{n} \varphi_i^{[\lambda]}.$$

Neglecting a factor m, the GCV parameter-choice method for Tikhonov regularization thus takes the following form:

$$\boxed{\text{Choose } \lambda = \lambda_{\mathrm{GCV}} \text{ as the minimizer of } G(\lambda) = \frac{\|A\,x_\lambda - b\|_2^2}{\left(m - \sum_{i=1}^{n} \varphi_i^{[\lambda]}\right)^2}.} \qquad (5.10)$$

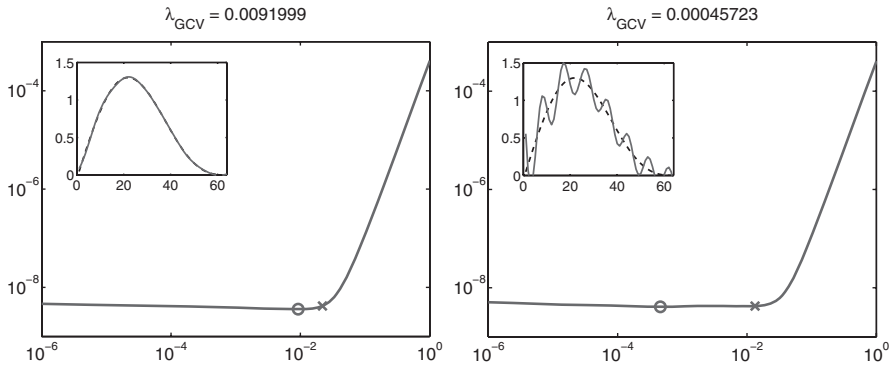

Figure 5.6. *The GCV function $G(\lambda)$ for Tikhonov regularization; the circles show the parameter λ_{GCV} as the minimum of the GCV function, while the crosses indicate the location of the optimal parameter. Left: The typical behavior of GCV, where the minimum is located near the transition between the "flat" and the "steep" parts of the curve. Right: The minimum is incorrectly located on the "flat" part, yielding a parameter that is much smaller than the optimal value.*

For the TSVD method, the trace term is much simpler, $\text{trace}(\Phi^{[k]}) = k$, and the GCV method takes the following form:

$$\boxed{\text{Choose } k = k_{\text{GCV}} \text{ as the minimizer of } G(k) = \frac{\|Ax_k - b\|_2^2}{(m-k)^2}.} \tag{5.11}$$

Experience shows that this parameter-choice method is quite robust and accurate, as long as the noise is white (because this assumption underlies the derivation of the GCV function). See Figure 5.6 for two examples of the GCV function $G(\lambda)$ for Tikhonov regularization. The left plot shows the typical behavior of GCV when the method works well and produces a good regularization parameter λ_{GCV} near the optimal one—which is near the transition between the "flat" and the "steep" parts of the GCV function. The right plot shows an example where GCV fails, which always occurs when the minimum is located somewhere on the "flat" part of the curve, leading to a λ_{GCV} that is much too small. This occasional failure of GCV is well understood, and it often reveals itself by the ridiculous undersmoothing it leads to.

We note in passing that it has been suggested to introduce a weighting factor in the GCV function for Tikhonov regularization, by replacing the function $G(\lambda)$ in (5.10) with the function

$$G_w(\lambda) = \frac{\|Ax_\lambda - b\|_2^2}{\left(m - w\sum_{i=1}^n \varphi_i^{[\lambda]}\right)^2}, \qquad w = \text{user-specified weight.} \tag{5.12}$$

While the idea seems to be completely heuristic, it can occasionally be used to improve the approximation underlying GCV.

The statistical and asymptotic properties of the GCV method and its underlying theory is very well understood due to the long line of work in this area by Prof. Grace

Wahba, whose book [75] should be consulted for details about the GCV method and its derivation.

5.5 Squeezing the Most Out of the Residual Vector—NCP Analysis

At this stage, recall that when the solution is oversmoothed (too few SVD components are included), then the residual vector is dominated by SVD components from the exact data b^{exact}. And when the solution is undersmoothed (too many SVD components are included), then the residual vector is dominated by SVD components from the noise e.

We already mentioned that, ideally, in order to choose the truncation parameter k for TSVD, we could monitor the Picard plot and select the regularization parameter such that precisely all coefficients $|u_i^T b|$ above the noise level η are included in the regularized solution. When the coefficients $|u_i^T b^{\text{exact}}|$ associated with the exact right-hand side decay monotonically, then this corresponds to choosing the TSVD truncation parameter k_η such that

$$|u_{k_\eta}^T b^{\text{exact}}| > \eta \geq |u_{k_\eta+1}^T b^{\text{exact}}|.$$

In practice, we should choose k as the index where the noisy coefficients $u_i^T b$ change from overall decaying behavior to leveling off.

The challenge is to implement this choice without computing the SVD, or involving a visual inspection of the Picard plot, for that matter. The approach taken in [38] and [62] (based on ideas originally developed in [61]) is to view the residual vector as a time series, and consider the exact right-hand side b^{exact} (which we know represents a smooth function) as a "signal" which appears distinctly different from the noise vector e. The goal is thus to find the regularization parameter for which the residual changes behavior from being signal-like (dominated by components from b^{exact}) to being noise-like (dominated by components from e).

Specifically, let us consider the TSVD method for which the residual vector is given by $r_k = \sum_{i=k+1}^{n} u_i^T b\, u_i$. If $m > n$, then we deliberately neglect the component $(I - UU^T)b$ outside the column space of A because it consists of noise only (by definition, b^{exact} is always in the column space of A). When k is too small, then, in the SVD basis, the residual vector r_k includes both signal components $u_i^T b \approx u_i^T b^{\text{exact}}$ (for i in the range $k < i < k_\eta$) and noise components $u_i^T b \approx \pm\eta$ (for $k_\eta < i \leq n$). And when k is too large, then r_k includes only noise components $u_i^T b \approx \pm\eta$ (for all i in the range $k < i \leq n$). So our aim is to choose k_η as the smallest k for which r_{k+1} appears statistically like noise.

The above idea is somewhat similar to the idea underlying the discrepancy principle from Section 5.2, where we choose the regularization parameter such that the norm of the residual equals the norm of the noise e. But there, the single piece of information we utilize about e is its standard deviation η. Here we wish to develop a more sophisticated criterion, based on the statistical question: When can the residual vector be considered as noise?

The discrete Fourier transform (DFT) lets us answer this question for the important case of white noise, via a technique that has been developed in both signal processing and statistics (using different nomenclature, of course). Let $\hat{r}_\lambda$ denote the DFT of the Tikhonov residual vector r_λ,

$$\hat{r}_\lambda = \mathrm{dft}(r_\lambda) = \Big((\hat{r}_\lambda)_1,\, (\hat{r}_k)_2,\, \ldots,\, (\hat{r}_\lambda)_m \Big)^T \in \mathbb{C}^m. \tag{5.13}$$

The DFT is always computed by means of the computationally efficient fast Fourier transform (FFT) algorithm. The power spectrum of r_λ is defined as the real vector

$$p_\lambda = \Big(|(\hat{r}_\lambda)_1|^2,\, |(\hat{r}_\lambda)_2|^2,\, \ldots,\, |(\hat{r}_\lambda)_{q+1}|^2 \Big)^T, \qquad q = \lfloor m/2 \rfloor, \tag{5.14}$$

in which q denotes the largest integer such that $q \leq m/2$. The elements of the power spectrum $p_\lambda \in \mathbb{R}^{q+1}$ represent the power in the signal at each of the basic spectral components, with the first component $(p_\lambda)_1$ representing the constant signal (the 0th frequency or "DC component"), and the last component $(p_\lambda)_{q+1}$ representing the highest frequency.

We now define the *normalized cumulative periodogram* (NCP) for the residual vector r_λ as the vector $c(r_\lambda) \in \mathbb{R}^q$ whose elements[4] involve the cumulated sums of the power spectrum:

$$c(r_\lambda)_i = \frac{(p_\lambda)_2 + \cdots + (p_\lambda)_{i+1}}{(p_\lambda)_2 + \cdots + (p_\lambda)_{q+1}}, \qquad i = 1, \ldots, q. \tag{5.15}$$

Note the normalization which ensures that the largest element is 1, $c(r_\lambda)_q = 1$, and that the first component of p_λ is not included in computing the NCP. Precisely the same definition holds for the NCP $c(r_k)$ associated with the residual vector for the TSVD method, with truncation parameter k.

If the vector r_λ consists of white noise, then, by definition, the expected power spectrum is flat, i.e., $\mathcal{E}((p_\lambda)_2) = \mathcal{E}((p_\lambda)_3) = \cdots = \mathcal{E}((p_\lambda)_{q+1})$, and hence the points on the expected NCP curve $\big(i\,,\, \mathcal{E}(c(r_\lambda)_i) \big)$ lie on a straight line from $(0, 0)$ to $(q, 1)$. Actual noise does not have an ideal flat spectrum, but we can still expect the NCP to be close to a straight line. In statistical terms, with a 5% significance level the NCP curve must lie within the Kolmogorov–Smirnoff limits $\pm 1.35\, q^{-1/2}$ of the straight line; see the examples in the left part Figure 5.7.

We have already seen examples of noise that is not white, and in Sections 3.6 and 4.8 we discussed LF and HF noise, i.e., noise dominated by low and high frequencies, respectively. The examples in the middle and right parts of Figure 5.7 show NCPs for realizations of such noise; notice that all the curves have segments outside the Kolmogorov–Smirnoff limits.

In this way, we can use the NCP to track the appearance of the residual vector as the regularization parameter changes. Figure 5.8 shows an example of the evolution of the Tikhonov NPC $c(r_\lambda)$ as the regularization parameter changes from $\lambda = 10^{-3}$ to $\lambda = 10^{-6}$ (i.e., from over- to undersmoothing). In between these values, there is

[4] We note the alternative expression $c(r_\lambda)_i = \|\hat{r}_\lambda(2\colon i{+}1)\|_2^2 / \|\hat{r}_\lambda(2\colon q{+}1)\|_2^2$, $i = 1, \ldots, q$, which avoids the use of p_λ.

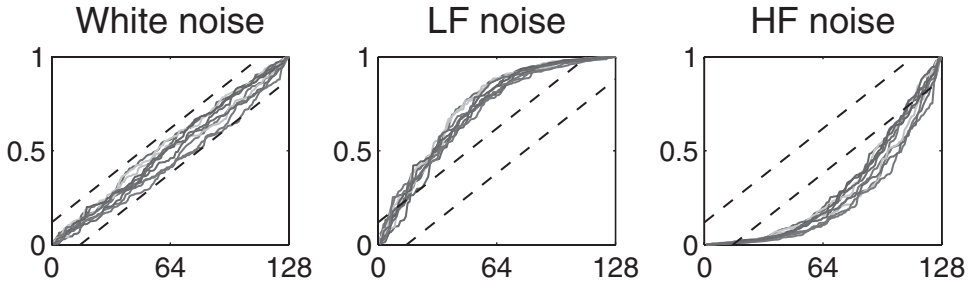

Figure 5.7. *Left to right:* 10 *instances of white-noise residuals,* 10 *instances of residuals dominated by low-frequency components, and* 10 *instances of residuals dominated by high-frequency components. All signals have length* $m = 256$ *such that* $q = 128$. *The dashed lines show the Kolmogorov–Smirnoff limits* $\pm 1.35\, q^{-1/2} \approx \pm 0.12$ *for a 5% significance level.*

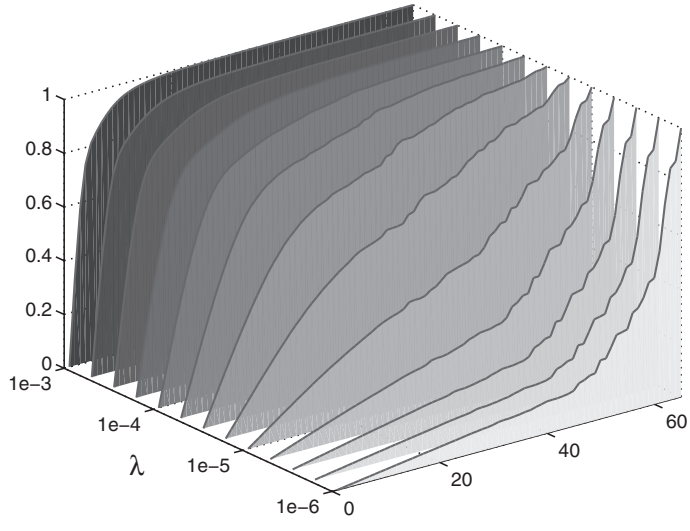

Figure 5.8. *Plots of NCPs* $c(r_\lambda)$ (5.15) *for various Tikhonov regularization parameters* λ, *for the test problem* `deriv2(128,2)` *with relative noise level* $\|e\|_2/\|b^{\text{exact}}\|_2 = 10^{-5}$. *The residual changes from being dominated by low-frequency components (SVD components of* b^{exact}*) for* $\lambda = 10^{-3}$ *to being dominated by high-frequency components (from the noise* e*) for* $\lambda = 10^{-6}$.

a λ such that the corresponding residual vector r_λ is white noise, as revealed in an almost linear NCP.

Clearly, as long as the noise is white, we can use the NCP as the basis for a parameter-choice method, because the NCP reveals precisely when the residual vector can be considered to be white noise. Specifically, we would increase the TSVD parameter k, or decrease the Tikhonov parameter λ, until we first encounter a situation

where $c(r_k)$ or $c(r_\lambda)$ fall entirely within the Kolmogorov–Smirnoff limits. As long as the residual vector is explicitly available, such a test is computationally feasible because of the reasonable overhead, $O(m \log m)$ flops, of the FFT algorithm.

Unfortunately, the above requirement may never be achieved in practice, partly because real data may fail to have completely white noise to begin with, and partly because we actually extract some SVD components of the noise, leaving the remaining noise component in the residual vector not quite white. Therefore, we obtain a more robust NCP-based parameter choice criterion if we choose that regularization parameter for which the residual vector most resembles white noise, in the sense that the NCP is closest to a straight line. In the implementation `ncp` in *Regularization Tools* we measure this as the 2-norm between the NCP and the vector $c_{\text{white}} = (1/q, 2/q, \ldots, 1)^T$; other norms can be used as well.

In summary, the *NCP parameter-choice criterion* for Tikhonov regularization takes the following form:

$$\boxed{\text{Choose } \lambda = \lambda_{\text{NCP}} \text{ as the minimizer of } d(\lambda) = \|c(r_\lambda) - c_{\text{white}}\|_2.} \tag{5.16}$$

For TSVD the NCP criterion takes the form

$$\boxed{\text{Choose } k = k_{\text{NCP}} \text{ as the minimizer of } d(k) = \|c(r_k) - c_{\text{white}}\|_2.} \tag{5.17}$$

Here the NCPs $c(r_\lambda)$ and $c(r_k)$ are defined in(5.15), and the use of the 2-norm is not essential. The extension of this method to 2D data (such as digital images) using the 2D FFT is described in [38].

While we always perform our analysis of regularization methods and regularized solutions in terms of the SVD, the discrete Fourier basis is the natural choice for the NCP analysis described above. The good news is that the NCP analysis still reveals information that ideally belongs in an SVD setting, due to the close relationship between the singular functions and the Fourier basis functions described in Section 2.5. In Chapter 7 we introduce so-called deconvolution problems where this relationship is even stronger, and Exercise 7.4 develops a variant of the NCP criterion for such problems.

5.6 Comparison of the Methods

With so many parameter-choice methods around, we would like to provide a guide to choosing among them. If a (really) good estimate of the error norm $\|e\|_2$ is available, then the discrepancy principle is a good choice. Otherwise, we must use one of the other methods that do not need this information explicitly, but which to prefer? Unfortunately there is no general answer; each inverse problem has its own characteristics and error model, and it is somewhat unpredictable which parameter-choice method is optimal for a given problem.

To illustrate this dilemma, we study the solution of two different test problems, namely, the gravity surveying problem `gravity` from Section 2.1, and the one-dimensional image reconstruction problem `shaw` from Exercise 3.6. For each test problem, we generate 500 instances of Gaussian white noise with standard deviation $\eta = 10^{-4}$,

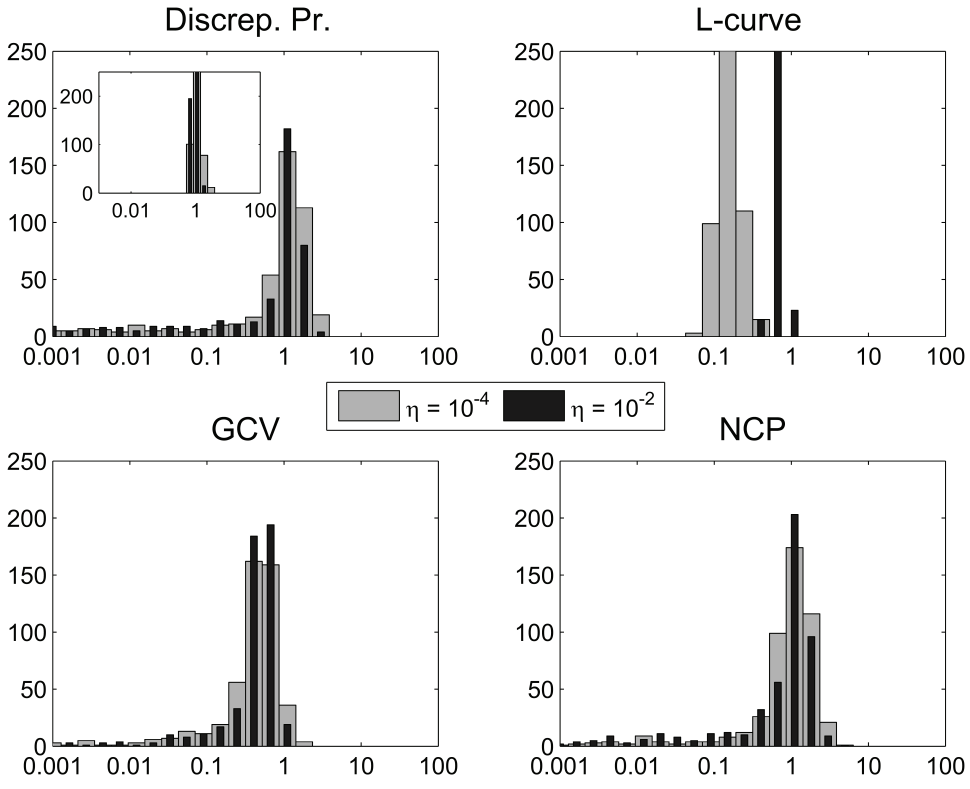

Figure 5.9. *Histograms of the four ratios R_{DP}, R_{L}, R_{GCV}, and R_{NCP} defined in* (5.18) *for the* `gravity` *test problem. The wide lighter bars represent the small noise level $\eta = 10^{-4}$, and the narrow darker bars represent the larger noise level $\eta = 10^{-2}$; there are* 500 *noise realizations for each level. The insert in the top left figure shows the histograms if we use the exact value of $\|\mathrm{e}\|_2$ in the discrepancy principle.*

and another 500 instances with $\eta = 10^{-2}$. For every instance of the noise, we compute the Tikhonov regularization parameter λ by means of the discrepancy principle (5.6) using $\nu_{\mathrm{dp}} = 1$ and the estimate $n^{1/2}\eta$ for the error norm, the L-curve criterion (5.7), GCV (5.10), and the NCP criterion (5.16). All computations used the relevant functions from *Regularization Tools*.

In order to evaluate the performance of the four methods, we also need to compute the optimal regularization parameter λ_{opt} which, by definition, minimizes the error in the regularized solution:

$$\lambda_{\mathrm{opt}} = \mathrm{argmin}_{\lambda} \|x^{\mathrm{exact}} - x_{\lambda}\|_2.$$

This allows us to compute the four ratios

$$R_{\mathrm{DP}} = \frac{\lambda_{\mathrm{DP}}}{\lambda_{\mathrm{opt}}}, \qquad R_{\mathrm{L}} = \frac{\lambda_{\mathrm{L}}}{\lambda_{\mathrm{opt}}}, \qquad R_{\mathrm{GCV}} = \frac{\lambda_{\mathrm{GCV}}}{\lambda_{\mathrm{opt}}}, \qquad R_{\mathrm{NCP}} = \frac{\lambda_{\mathrm{NCP}}}{\lambda_{\mathrm{opt}}}, \qquad (5.18)$$

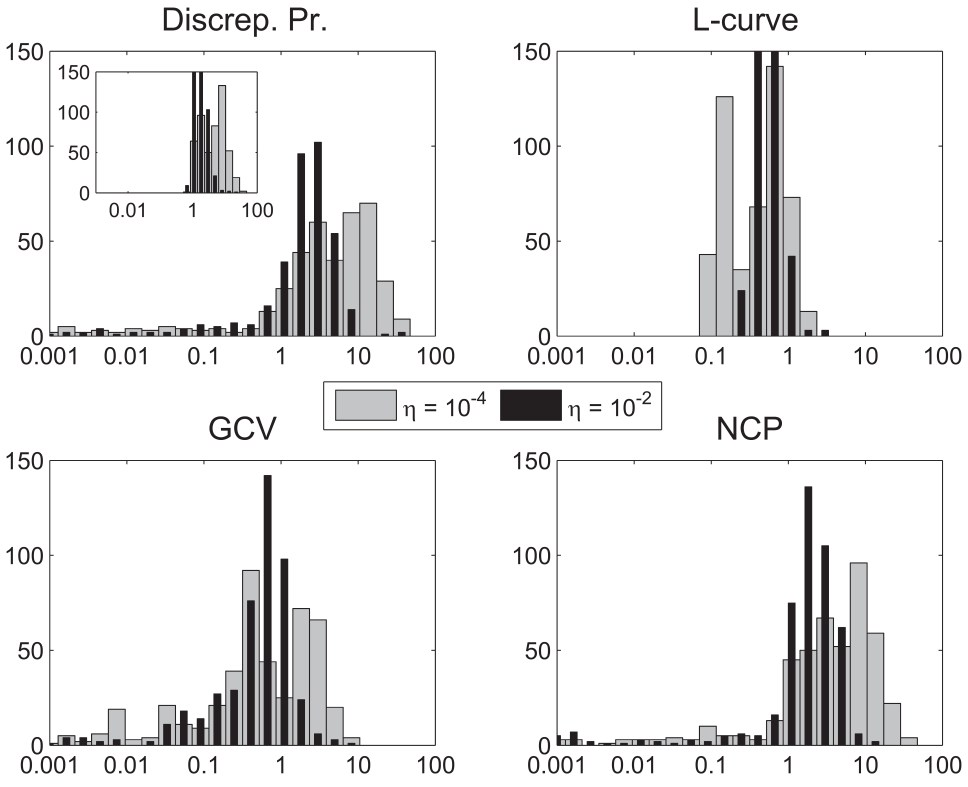

Figure 5.10. *Same legend as in Figure* 5.9*, except these results are for the* `shaw` *test problem.*

one for each parameter-choice method, and study their distributions via plots of their histograms (in log scale). The closer these ratios are to 1, the better, so a spiked histogram located at 1 is preferable.

Consider first the results for the `gravity` problem in Figure 5.9. Except for the L-curve criterion, these results are independent of the noise level.

- **Discrepancy principle.** The peak of the histogram is located close to 1, and it is quite sharp. The optimal parameter is never overestimated by more than a factor 3. Unfortunately the histogram has a wide tail toward small ratios, with some ratios as small as 0.001—a really severe underestimate of the optimal parameter. The insert shows histograms for the situation where we know the exact error norm $\|e\|_2$ in each instance of the noise; now the tail vanishes, but it is rare that we have such good estimates.

- **L-curve criterion.** For the larger noise level, this method works exceptionally well—the histogram is very peaked and there is no tail. For the smaller noise level the average parameter λ_L is much too small; this is caused by the very smooth solution whose SVD coefficients decay quickly, leading to the phenomenon of

including too many SVD components in the regularized solution discussed in Section 5.3.

- **GCV.** This histogram resembles that for the discrepancy principle, except that the peak is located at about 0.5, and the tail toward small ratios is a lot thinner (i.e., there is a smaller risk of underestimation). The existence of the tail reflects the occasional failure of the GCV method already mentioned in Section 5.4.

- **NCP criterion.** This histogram also resembles that for the discrepancy principle, except that the tail is much thinner, especially for the larger noise level. The peak is located quite close to one.

For this problem, the NCP appears to be the method of choice, because it works well for both noise levels, giving a robust estimate λ_{NCP} of the regularization parameter that is often very close to the optimal one (with a small risk for underestimation). If we know that the noise is always large, however, then the L-curve criterion is the winner, but it is a bad choice for small noise levels (due to the smooth solution). Among the other two methods, whose performance is independent of the noise level, the discrepancy principle is more accurate than GCV on average, but it is also more likely to fail.

Now consider the results for the `shaw` test problem shown in Figure 5.10. These results show a quite different behavior of the methods than we saw before; for example, the results now depend on the noise level for all four methods.

- **Discrepancy principle.** This method is very likely to overestimate the regularization parameter by a factor of 3 (for $\eta = 10^{-2}$) or even 10 (for $\eta = 10^{-4}$). Still, the tail toward underestimation is also nonnegligible. Even with the exact norm of the noise (shown in the insert), the method tends to overestimate.

- **L-curve.** Some underestimation must be expected for this method—by up to a factor 10 for $\eta = 10^{-4}$, and better for $\eta = 10^{-2}$. There is no tail toward underestimation, so the method is quite robust.

- **GCV.** The mean of the histograms is located at values slightly smaller than 1, and on average the behavior of this method is good, but there is a risk of over- or underestimating the regularization parameter by up to a factor 10. Also, severe underestimation is possible.

- **NCP criterion.** This method overestimates the regularization parameter considerably, by up to a factor 10 or more for the smaller noise level, and almost as bad for the larger noise level. There is also a slight risk of severe underestimation.

For this problem, it is highly advisable not to use the discrepancy principle or the NCP criterion. GCV, on the other hand, performs very well (except for the occasional failure). The L-curve criterion never fails, but the average estimate is too small. So the choice is between robustness and high quality of the estimate.

We could show more examples of the diverse behavior of the four methods, but the above two examples should suffice to show how difficult it is to make general

statements about the different parameter-choice methods. For a given inverse problem, it is always recommended to try several parameter-choice methods on artificial data!

5.7 The Story So Far

This chapter deals with the important topic of methods for automatically choosing the regularization parameter from the data, preferably without using any information about the data that may not be readily available. The only assumption used throughout the chapter is that the noise is white.

We started with an analysis of the regularized solution by splitting it into a noise component or perturbation error, and a bias component or regularization error, and we stated our goal: to find a regularization parameter that balances these two error components in the solution. This corresponds to extracting all available "signal" from the data (the right-hand side) until only noise is left in the residual vector. We then presented four different parameter-choice methods based on quite different ideas.

- The discrepancy principle is a simple method that seeks to reveal when the residual vector is noise-only. It relies on a good estimate of the error norm $\|e\|_2$ which may be difficult to obtain in practice.
- The L-curve criterion is based on an intuitive heuristic and seeks to balance the two error components via inspection (manually or automated) of the L-curve. This method fails when the solution is very smooth.
- GCV seeks to minimize the prediction error, and it is often a very robust method—with occasional failure, often leading to ridiculous undersmoothing that reveals itself.
- The NCP criterion is a statistically based method for revealing when the residual vector is noise-only, based on the power spectrum. It can mistake LF noise for a signal and thus lead to undersmoothing.

A comparative study of the four methods for two different test problems illustrates that it is generally not possible to favor one method over the others, since the behavior of the methods is very problem-dependent. Hence, it is preferable to have several parameter-choice methods at our hands for solving real-world problems.

Exercises

5.1. **The Discrepancy Principle**
The purpose of this exercise is to illustrate the sensitivity of the discrepancy principle to variations of the estimate of the error norm. First generate the `shaw` test problem for $n = 100$, and add Gaussian noise with standard deviation $\eta = 10^{-3}$ to the right-hand side.

Use the discrepancy principle to compute the Tikhonov solution. The discrepancy principle is implemented in the MATLAB function `discrep` from *Regularization Tools*. This function may occasionally have convergence problems; if this happens, then create a new noise realization (or change η slightly) and try again.

You should first use the "safety factor" $\nu_{\text{dp}} = 1$. As the "right-hand side" $\|e\|_2$ in (5.6), try to use both the norm estimate $\sqrt{n}\,\eta$ and the actual 2-norm $\|e\|_2$ of the perturbation vector e (perhaps try with different perturbations). Is there a significant difference in the results?

Still with the "safety factor" $\nu_{\text{dp}} = 1$, use several values of $\sqrt{n}\,\eta$ and/or $\|e\|_2$ that are slightly too large and slightly too small; this simulates a situation where only a rough estimate is available. For example, scale $\sqrt{n}\,\eta$ and/or $\|e\|_2$ up and down by 5–10%. How sensitive is the regularized solution to overestimates and underestimates of the noise level?

Finally, repeat the experiments with the "safety factor" $\nu_{\text{dp}} = 2$, and comment on the new results.

5.2. **The GCV and L-curve Methods**

This exercise illustrates the use of the GCV and L-curve methods for choosing the regularization parameter, and we compare these methods experimentally. As part of this comparison we investigate how robust—or reliable—the methods are, i.e., how often they produce a regularization parameter close to the optimal one. We use the `shaw` test problem and the parameter-choice functions `gcv` and `l_curve` from *Regularization Tools*.

Plot the GCV function for, say, 5 or 10 different perturbations with the same η, and note the general behavior of the GCV function. Is the minimum always at the transition region between the flat part and the more vertical part (cf. Figure 5.6)?

Use the L-curve criterion to compute the regularization parameter for the same 5 or 10 perturbations as above. Does the regularization parameter computed by means of `l_curve` always correspond to a solution near the corner (cf. Figure 5.4)?

For a fixed perturbation of the right-hand side, compute the regularization error and the perturbation error for Tikhonov's method, for a range of regularization parameters λ. Plot the norms of the two error components Δx_{bias} and Δx_{pert}, and relate the behavior of this plot to the two regularization parameters found by the GCV method and the L-curve criterion.

5.3. **NCP Studies**

This exercise illustrates the use of the NCP for analysis of the spectral behavior of filtered signals. Choose a signal length, say, $m = 128$, and generate a white-noise signal vector x of length m: Then use the following recursion to generate a filtered signal y of the same length as x:

$$y_1 = x_1, \qquad y_i = x_i + c\,x_{i-1}, \quad i = 2, \ldots, m,$$

where c is a constant in the range $-1 \le c \le 1$ (this is a common simple filter in signal processing for generating colored noise). Use the NCP to answer the

following questions: For which sign of c is y dominated by high frequencies? In which range should you choose c if the NCP for y has to stay within the Kolmogorov–Smirnoff limits?

5.4. **Sensitivity Analysis***

The purpose of this exercise is to investigate how sensitive the regularized solutions are to perturbations once a regularization parameter has been computed by means of GCV or the L-curve criterion. The exercise also illustrates that the regularization parameter is so dependent on the noise that it is impossible to select the parameter a priori.

Generate your favorite test problem, add a particular perturbation e (η should be neither too large nor too small), and compute the regularization parameters λ_{GCV} and λ_{L} for Tikhonov regularization by means of the GCV and L-curve methods. Make sure that these parameters are reasonable; otherwise, choose another perturbation and try again.

Now create a series of problems with different realizations of the noise e (with the same η as above) added to b^{exact}. For each noise realization compute the Tikhonov solutions using the same two regularization parameters λ_{GCV} and λ_{L} as above, and compute their errors. How sensitive are the solutions to the variations in e?

Chapter 6

Toward Real-World Problems: Iterative Regularization

We could stop our presentation at this stage! We have at our disposal several regularization methods, as well as several methods for choosing the regularization parameter from the given data. The study of these methods has provided us with a good understanding of the fundamental mechanisms of regularization, and by combining the methods we have a good chance of solving a moderately sized discrete inverse problem.

The methods we have discussed so far are designed for problems where it is feasible to compute the SVD or compute Tikhonov solutions via the least squares formulation for several values of the regularization parameter. Many real problems, however, lead to large matrices where these factorization methods are infeasible with respect to computing time and/or storage.

Discretizations of some classes of large-scale inverse problems naturally lead to sparse coefficient matrices; i.e., A contains so many zeros that it is computationally advantageous to exploit the zero-nonzero structure via sparse matrix storage formats. Discretizations of other large-scale problems involve matrices A that are not explicitly stored; instead we have software (a "black-box" function) that computes matrix-vector products with A and A^T. Examples of both classes of problems are given in Chapter 7, and in both cases factorization is infeasible.

For a regularization method to be useful for solving large-scale problems, it must thus avoid factorization and instead satisfy the following two requirements.

- The main "building blocks" of a large-scale method must be matrix-vector multiplications, such that any factorization of the matrix is avoided.
- A large-scale method should allow the user to select the regularization parameter, via one of the parameter-choice methods, without solving the problem from scratch for each new parameter.

The first requirement naturally leads to the use of iterative methods, because they require only matrix-vector products. One approach that immediately may come to mind is to use an iterative least squares method for solving the Tikhonov problem in the form (4.9). The main disadvantage of this approach is that for each new regularization parameter λ one needs to rerun the iterative method, which is intractable when the problem and the computing time are large.

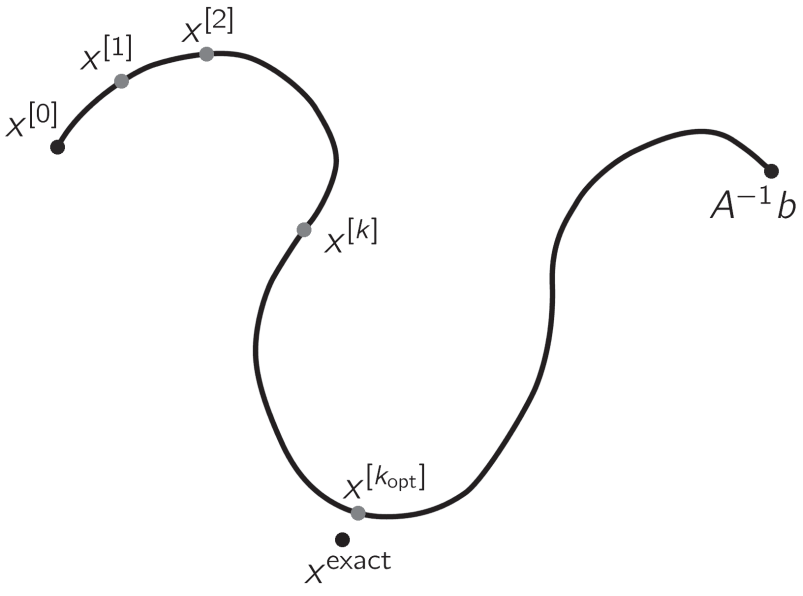

Figure 6.1. *The basic concept of semiconvergence. During the first iterations, the iterates $x^{[k]}$ tend to be better and better approximations to the exact solution x^{exact} (the optimal iterate is obtained at iteration k_{opt}), but at some stage they start to diverge again and instead converge toward the "naive" solution $A^{-1}b$.*

To also satisfy the second requirement we must rely on iterative methods that avoid a predetermined regularization parameter, and where instead the number of iterations plays the role of a regularization parameter. Recall that iterative methods always start with a user-specified starting vector $x^{[0]}$ (often the zero vector) and produce a sequence of iterates $x^{[1]}$, $x^{[2]}$, ... that converge to some solution. For certain iterative methods, the iterates $x^{[k]}$ initially resemble regularized (filtered) solutions, and for $k = 1, 2, \ldots$ they tend to be better and better approximations to the exact solution x^{exact}. In later stages, however, the iterates will start to diverge again from x^{exact} and instead converge to the "naive" solution. This kind of convergence is called *semiconvergence* (see Figure 6.1), and if we can stop the iterations at the right time, then, in principle, we have a large-scale regularization method.

This chapter should be considered as a gentle and simplified introduction to the large area of regularizing iterations. We present the main ideas and techniques, but many methods as well as important details and technicalities are left out (for example, we do not venture into the finite-precision aspects of the Krylov-subspace methods).

6.1 A Few Stationary Iterative Methods

We start our treatment of iterative regularization methods with a short discussion of a few classical stationary iterative methods that exhibit semiconvergence. While these

methods are not suited as general-purpose iterative regularization methods because they can exhibit very slow convergence, they have certain specialized applications, e.g., in tomography applications, where they are competitive with other methods. A nice overview of these methods can be found in Chapter 7 of [48].

6.1.1 Landweber and Cimmino Iteration

One of the best known methods that exhibits semiconvergence is usually referred to as *Landweber iteration* (although many scientists have independently discovered this simple method). In its basic form the method looks as follows:

$$\boxed{x^{[k+1]} = x^{[k]} + \omega\, A^T(b - A\,x^{[k]}), \qquad k = 1, 2, \ldots,} \tag{6.1}$$

where ω is a real number that must satisfy $0 < \omega < 2\,\|A^T A\|_2^{-1} = 2/\sigma_1^2$. That's it; all that is needed in each iteration is to compute the residual vector $r^{[k]} = b - A\,x^{[k]}$ and then multiply with A^T and ω; this correction is then added to the iterate $x^{[k]}$ to obtain the next iterate.

The iterate $x^{[k]}$ has a simple expression as a filtered SVD solution, similar to the TSVD and Tikhonov solutions; cf. (5.1). Specifically, we can write the kth iterate as

$$x^{[k]} = V\,\Phi^{[k]}\,\Sigma^{-1}U^T b,$$

where the elements of the diagonal matrix $\Phi^{[k]} = \mathrm{diag}(\varphi_1^{[k]}, \ldots, \varphi_n^{[k]})$ are the filter factors for $x^{[k]}$, which are given by

$$\varphi_i^{[k]} = 1 - (1 - \omega\,\sigma_i^2)^k, \qquad i = 1, 2, \ldots, n. \tag{6.2}$$

Moreover, for small singular values σ_i we have that $\varphi_i^{[k]} \approx k\,\omega\,\sigma_i^2$, i.e., they decay with the same rate as the Tikhonov filter factors $\varphi_i^{[\lambda]}$ (4.11). (The verification of these results is left as Exercise 6.1.) Figure 6.2 shows how these filters "evolve" as the number of iterations k increases. We see that as k increases, the filters are shifted toward smaller singular values, and more SVD components are effectively included in the iterate $x^{[k]}$.

If we, quite arbitrarily, define the break-point in the filter factors as the value $\sigma_{\mathrm{break}}^{[k]}$ of σ_i for which $\varphi_i^{[k]} = 0.5$, then it is easy to show that

$$\sigma_{\mathrm{break}}^{[k]} = \sqrt{\frac{1 - (\frac{1}{2})^{\frac{1}{k}}}{\omega}}.$$

The break-points are indicated in Figure 6.2 by the circles. To see how the break-point varies with k, let us consider the ratio $\sigma_{\mathrm{break}}^{[k]}/\sigma_{\mathrm{break}}^{[2k]}$ between the break-points at k and $2k$ iterations:

$$\frac{\sigma_{\mathrm{break}}^{[k]}}{\sigma_{\mathrm{break}}^{[2k]}} = \sqrt{\frac{1 - (\frac{1}{2})^{\frac{1}{k}}}{1 - (\frac{1}{2})^{\frac{1}{2k}}}} = \sqrt{\frac{\left(1 + (\frac{1}{2})^{\frac{1}{2k}}\right)\left(1 - (\frac{1}{2})^{\frac{1}{2k}}\right)}{1 - (\frac{1}{2})^{\frac{1}{2k}}}}$$

$$= \sqrt{1 + (\tfrac{1}{2})^{\frac{1}{2k}}} \to \sqrt{2} \quad \text{for} \quad k \to \infty.$$

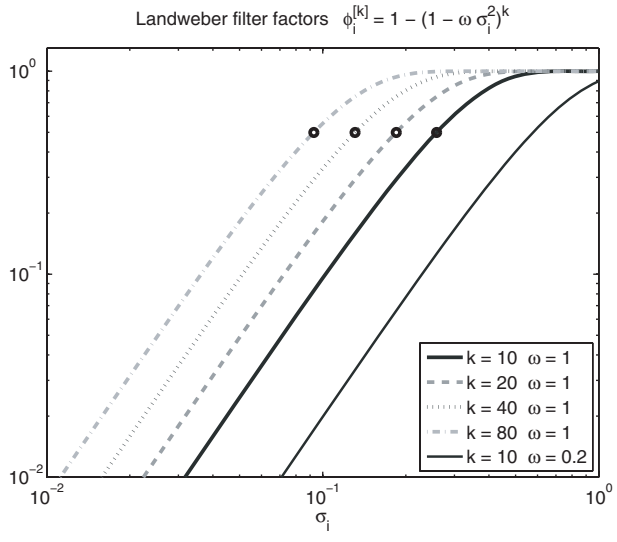

Figure 6.2. *The Landweber filter factors* $\varphi_i^{[k]} = 1 - (1 - \omega\,\sigma_i^2)^k$ *are functions of the number of iterations* k *and the parameter* ω*. The circles indicate the break-points* $\sigma_{\text{break}}^{[k]}$ *in the filter factors where they attain the value* 0.5*.*

Hence, as k increases, the break-point tends to be reduced by a factor $\sqrt{2} \approx 1.4$ each time the number of iterations k is doubled. The consequence is that the semiconvergence toward the exact solution is quite slow.

Another classical iterative method with semiconvergence, known as *Cimmino iteration*, takes the basic form

$$x^{[k+1]} = x^{[k]} + \omega\, A^T D\,(b - A\,x^{[k]}), \qquad k = 1, 2, \ldots,$$

in which $D = \text{diag}(d_i)$ is a diagonal matrix whose elements are defined in terms of the rows $a_i^T = A(i,:)$ of A as

$$d_i = \begin{cases} \dfrac{1}{m}\,\dfrac{1}{\|a_i\|_2^2}, & a_i \neq 0, \\ 0, & a_i = 0. \end{cases}$$

Landweber's and Cimmino's methods are two special cases of a more general class of stationary Landweber-type methods, sometimes referred to as simultaneous iterative reconstruction techniques [71]. These methods take the form

$$x^{[k+1]} = x^{[k]} + \omega\, A^T M\,(b - A\,x^{[k]}), \qquad k = 1, 2, \ldots,$$

where M is a symmetric and positive (semi)definite matrix. The study of this class of methods is similar to that of Landweber's method, except that in the analysis we substitute $M^{1/2}A$ and $M^{1/2}b$ for A and b.

The Landweber-type methods can be augmented to take the form

$$x^{[k+1]} = \mathsf{P}\left(x^{[k]} + \omega\, A^T M\,(b - A\,x^{[k]})\right), \tag{6.3}$$

where the operator P represents so-called hard constraints by which we can incorporate additional a priori knowledge about the regularized solution. Exercise 6.2 illustrates this for the case where P represents a nonnegativity constraint.

6.1.2 ART, a.k.a. Kaczmarz's Method

Another classical method that is worth mentioning here is Kaczmarz's method, also known in computerized tomography as the algebraic reconstruction technique (ART); cf. Section 5.3.1 in [56]. In this so-called row action method, the kth iteration consists of a "sweep" through the rows of A, in which the current solution vector $x^{[k]}$ (for $k = 0, 1, 2, \ldots$) is updated as follows:

$$\begin{aligned}
&x^{[k^{(0)}]} = x^{[k]} \\
&\text{for } i = 1, \ldots, m \\
&\qquad x^{[k^{(i)}]} = x^{[k^{(i-1)}]} + \frac{b_i - a_i^T x^{[k^{(i-1)}]}}{\|a_i\|_2^2}\, a_i \\
&\text{end} \\
&x^{[k+1]} = x^{[k^{(m)}]}
\end{aligned}$$

Here b_i is the ith component of the right-hand side b, and a_i is the ith row turned into a column vector. The row norms $\|a_i^T\|_2 = \|a_i\|_2$ are, of course, precomputed once.

All experience shows that the ART method always converges quite quickly in the first few iterations, after which the convergence may become very slow. It was probably the fast initial convergence that made it a favorite method in the early days of computerized tomography, where only a few iterations were carried out.

It can be shown that Kaczmarz's method is mathematically equivalent to applying Gauss–Seidel iterations to the problem $x = A^T y$, $A\,A^T y = b$. The method can be written in the form of a Landweber-type method, but with a matrix M that is nonsymmetric. This severely complicates the analysis of the method, and the iterates cannot be written in the form (5.1) with a diagonal filter matrix.

Let us for a moment study Kaczmarz's method from a geometric perspective. As shown in [54], each iterate $x^{[k^{(i)}]}$ is obtained by projecting the previous iterate $x^{[k^{(i-1)}]}$ orthogonally onto the hyperplane $\mathcal{H}_i = \{\, x \,|\, a_i^T x = b_i \,\}$ defined by the ith row a_i^T and the corresponding element b_i of the right-hand side. Also, it is proved in [54] that if A is square and nonsingular, then $\|A^{-1}b - x^{[k^{(i)}]}\|_2^2 = \|A^{-1}b - x^{[k^{(i-1)}]}\|_2^2 - (b_i - a_i^T x^{[k^{(i-1)}]})^2$; i.e., the residual norm is nonincreasing.

The above geometric interpretation of ART gives some insight into the regularizing properties of this method. Let θ_{ij} denote the angle between the vector a_i and the jth right singular vector v_j. Then, using that $a_i^T = e_i^T A$, where e_i is the ith unit vector, we get

$$\cos\theta_{ij} = \frac{a_i^T}{\|a_i\|_2} v_j = \frac{e_i^T\, A\, v_j}{\|a_i\|_2} = \frac{\sigma_j\, e_i^T u_j}{\|a_i\|_2} = \frac{\sigma_j\, u_{ij}}{\|a_i\|_2}.$$

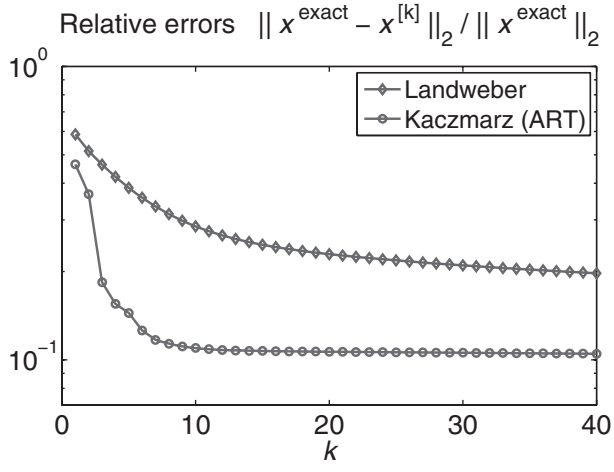

Figure 6.3. *Example of the convergence histories for the Landweber and Kaczmarz (ART) methods applied to the* `shaw` *test problem of size* $n = 128$. *Both methods converge slowly, but the initial converge of ART (during the first, say,* 6 *iterations) is quite fast.*

We can conclude that the vectors a_i are rich in the directions v_j that correspond to large singular values σ_j, while they are almost orthogonal to those singular vectors v_j that correspond to the small singular values. Hence, the Kaczmarz (ART) iterates $x^{[k^{(i)}]}$ (which always lie on the hyperplanes $\mathcal{H}_i$ and thus in the direction of the vectors a_i) are guaranteed to have large components in the direction of the right singular vectors corresponding to the larger singular values. On the other hand, components in directions that correspond to small singular values are hardly present. In other words, the iterates tend to behave as regularized solutions.

Figure 6.3 shows an example of the error histories, i.e., the relative errors $\|x^{\text{exact}} - x^{[k]}\|_2 / \|x^{\text{exact}}\|_2$ as functions of k, for the Landweber and Kaczmarz (ART) methods. The Landweber parameter was chosen as $\omega = \|A\|_{\text{F}}^{-2}$ (using the Frobenius norm as an overestimate of σ_1). We see that both methods exhibit semiconvergence: they actually converge (during the first many iterations) toward the exact solution x^{exact}. Moreover, both methods converge very slowly. During the first few iterations the convergence of ART is quite good; but the method essentially stagnates after about 10 iterations. Clearly, if better accuracy is required, then there is a need for faster general-purpose iterative methods.

6.2 Projection Methods

Before we discuss more modern—and faster—iterative regularization methods, we sidetrack a bit into the area of projection methods, partly because they are interesting in their own right, and partly because they provide a means for understanding the behavior of the iterative methods based on Krylov subspaces.

As we have already seen in the analysis of discrete ill-posed problems, it is impossible to compute a perfect reconstruction of x^{exact}. Our SVD analysis informs us that the best we can do is to compute an approximation to x^{exact} that effectively lies in a (low-dimensional) subspace of $\mathbb{R}^n$. For TSVD, this subspace is spanned by the first k right singular vectors $v_1, \ldots, v_k$, and we know that this subspace mainly captures the low-frequency information in x^{exact}, because these singular vectors have few sign changes. For Tikhonov's method, and for a suitably chosen value of λ, all the significant SVD components lie approximately in the same (low-dimensional) subspace.

For large-scale problems it is infeasible—or impossible—to compute the SVD of A. But we still know from the SVD analysis that the regularized solution should be dominated by low-frequency components. In fact, we might be able to provide, a priori, a set of basis vectors $w_1, \ldots, w_k$ that have the same overall features as the singular vectors, namely, being dominated by low-frequency components.

As an example, we could choose the orthonormal basis vectors of the discrete cosine transform (DCT) as our basis vectors $w_1, \ldots, w_k$. These vectors are given by

$$w_1 = \sqrt{1/n}\,(1, 1, \ldots, 1)^T, \tag{6.4}$$

and for $i = 2, 3, \ldots$

$$w_i = \sqrt{2/n}\,\left(\cos\left(\frac{i\,\pi}{2n}\right), \cos\left(\frac{3\,i\,\pi}{2n}\right), \ldots, \cos\left(\frac{(2n-1)\,i\,\pi}{2n}\right)\right)^T. \tag{6.5}$$

Some of these vectors are shown in Figure 6.4; we see that they are scaled samples of cosine functions with increasing frequency. We emphasize that these basis vectors are not needed explicitly; instead all computations that involve these vectors are performed by means of efficient numerical algorithms based on the FFT. For example, if $W_k = (w_1, \ldots, w_k)$ denotes the $n \times k$ matrix consisting of the first k DCT basis vectors, and if we need to compute the vector $y = W_k^T x$ (of length k), then we can do this in MATLAB by means of the `dct` function:

```
y = dct(x); y = y(1:k);
```
(6.6)

i.e., we first compute the DCT of x and then we extract the first k elements. Similarly, if we need to compute $x = W_k\,y = \sum_{i=1}^{k} w_i\,y_i$ (of length n), then we will use the `idct` function which implements the inverse DCT:

```
x = idct([y;zeros(n-k,1)]).
```
(6.7)

Obviously, the same techniques apply to many other sets of basis functions associated with fast transforms, such as wavelets.

Let us now assume that we have chosen a suitable basis for a low-dimensional subspace, spanned by the columns of the matrix $W_k = (w_1, \ldots, w_k) \in \mathbb{R}^{n\times k}$, and that we want a solution expressed in this basis that fits the right-hand side b. We can formulate this as a constrained least squares problem:

$$\boxed{\min_x \|A\,x - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{W}_k = \text{span}\{w_1, \ldots, w_k\}.} \tag{6.8}$$

We note in passing that the Galerkin discretization method from Section 3.1 is also a projection method, where we find a solution to (2.2) in the subspace spanned by the basis functions $\phi_1, \ldots, \phi_n$.

We can reformulate the constraint in (6.8) as the requirement that $x = W_k\, y$, where $y \in \mathbb{R}^k$ is the new unknown. This leads to a regularized solution expressed in the more computation-friendly formulation

$$x^{(k)} = W_k\, y^{(k)}, \qquad y^{(k)} = \operatorname{argmin}_y \|(AW_k)\, y - b\|_2. \tag{6.9}$$

We refer to the least squares problem $\|(AW_k)\, y - b\|_2$ in (6.9) as the *projected problem*, because it is obtained by projecting the original problem onto the k-dimensional subspace $\operatorname{span}\{w_1, \ldots, w_k\}$. If k is not too large, then we can explicitly compute the matrix $AW_k \in \mathbb{R}^{n\times k}$ and then solve the projected problem, i.e., the least squares problem, for y.

As a special case of the projection problem, if $W_k = (v_1, \ldots, v_k)$, i.e., if the basis vectors are the first k right singular vectors of A, then the projected problem takes the form

$$\begin{aligned} \|U\Sigma V^T W_k\, y - b\|_2 &= \left\| U\Sigma \begin{pmatrix} I_k \\ 0 \end{pmatrix} y - b \right\|_2 \\ &= \left\| \begin{pmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{pmatrix} y - b \right\|_2 . \end{aligned}$$

It follows immediately that the elements of $y^{(k)}$ are $y_i^{(k)} = u_i^T b/\sigma_i$, and therefore the projected solution $x^{(k)} = (v_1, \ldots, v_k)\, y^{(k)}$ is the well-known TSVD solution. Hence, TSVD is actually a projection method based on the singular subspace $\operatorname{span}\{v_1, \ldots, v_n\}$. We note that, in general, the solution computed via projection does not have the form of a filtered SVD expansion (4.1).

Assume once again that the columns of W are the DCT basis vectors. To compute the matrix $\widehat{A}_k = AW_k$ we note that it consists of the first k columns of the matrix $AW = (W^T A^T)^T$. In principle we can therefore compute $\widehat{A}_k$ in MATLAB as `Ahat = dct(A'); Ahat = Ahat(1:k,:)';` but this is inefficient since we produce a large submatrix which is immediately discarded. We obtain a more storage-efficient implementation by computing $\widehat{A}_k$ one row at at time (assuming the preallocation `Ahat = zeros(m,k);`):

```
for i=1:m, z = dct(A(i,:)'); Ahat(i,:) = z(1:k); end
```

Once the solution y to the projected problem has been computed, we compute x by means of (6.7).

Figure 6.4 illustrates the use of a projection method with the DCT basis, applied to the `shaw` test problem of size $n = 128$ and with a small noise level of $\|e\|_2/\|b^{\text{exact}}\|_2 \approx 4 \cdot 10^{-5}$. We show the solutions computed by projecting the original problem onto the basis vectors $w_1, \ldots, w_k$ for $k = 1, 2, \ldots, 10$. These solutions are clearly different from the TSVD solutions, and they inherit the characteristics of the

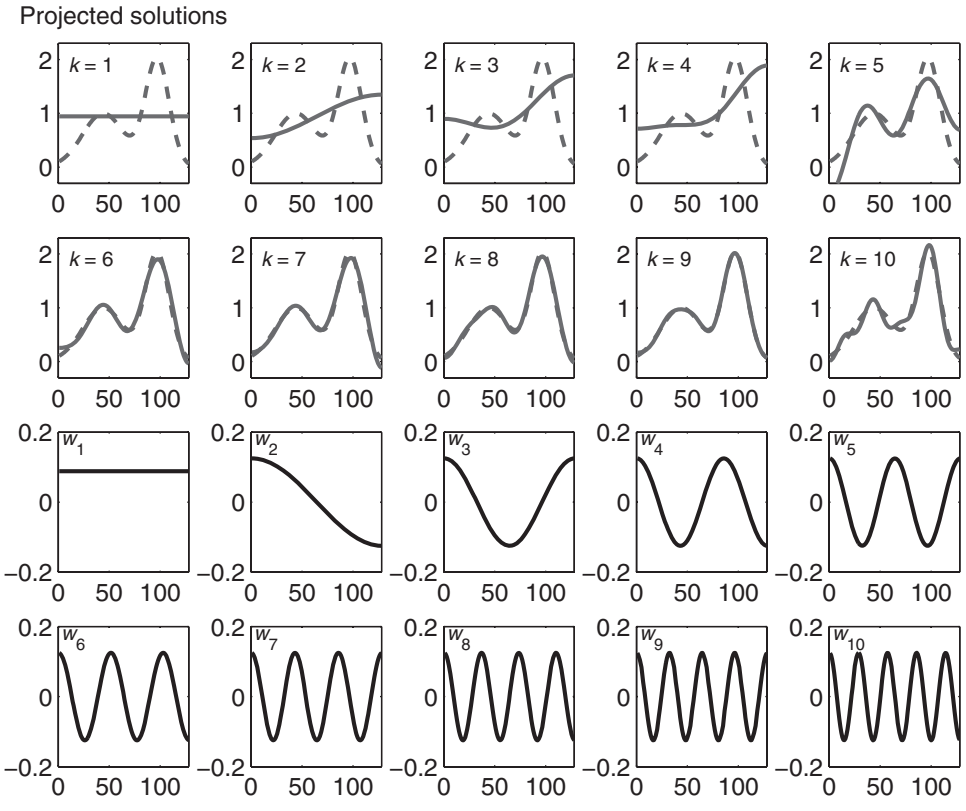

Figure 6.4. *Example of the use of a projection method based on the DCT basis. The figures in the two bottom rows show the DCT basis vectors* $w_1, \ldots, w_{10}$*, which are sampled cosines. The figures in the two top rows show the exact solution* x^{exact} *(dashed line) and the projected solution* $x^{(k)}$ *computed by projecting the original problem onto the basis vectors* $w_1, \ldots, w_k$ *for* $k = 1, 2, \ldots, 10$.

DCT basis vectors. We see that the projection onto a low-dimensional subspace indeed has a regularizing effect on the solution, and that $k = 9$ seems to be the optimal dimension of the projection subspace.

If the solution is known a priori to have certain properties, then we may be able to take advantage of this knowledge in choosing the basis vectors. For example, if we know that the exact solution is periodic or symmetric, then we should choose periodic or symmetric basis vectors. As an illustration of this adaptive feature of the projection method, consider the `phillips` test problem from *Regularization Tools* (cf. Section 3.2.2) whose exact solution is symmetric, i.e., $x_i^{\text{exact}} = x_{n-i+1}^{\text{exact}}$ for $i = 1, 2, \ldots, n/2$. Figure 6.5 shows projected solutions with two different bases, namely, the full DCT basis $w_1, w_2, w_3, \ldots$ and the basis of odd-numbered DCT vectors $w_1, w_3, w_5, \ldots$, the latter consisting purely of symmetric vectors. As shown in

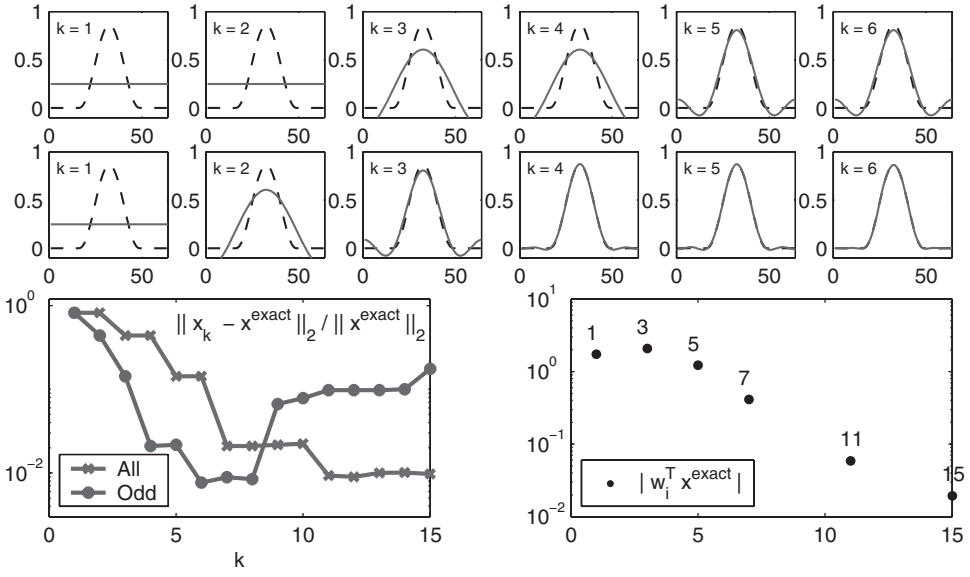

Figure 6.5. *The* `phillips` *test problem has a symmetric solution with* $x_i^{\text{exact}} = x_{n-i+1}^{\text{exact}}$ *for* $i = 1, 2, \ldots, n/2$. *The top row shows solutions* $x^{(k)}$ *computed with a projection method using the full DCT basis* $w_1, w_2, w_3, \ldots$, *while the second row shows solutions using the basis of odd-numbered DCT vectors* $w_1, w_3, w_5, \ldots$; *clearly, a smaller number of basis vectors is needed in the second case to obtain a good approximate solution that is guaranteed to be symmetric. The bottom left plot of the relative errors in* $x^{(k)}$ *confirms this. The bottom right plot shows the coefficients* $w_i^T x^{\text{exact}}$ *of the exact solution in the DCT basis, showing that only (some) odd-numbered basis vectors contribute to* x^{exact}.

the bottom right plot of Figure 6.5, the exact solution's coefficients to the even-numbered basis vectors are all zero. Hence, we can safely exclude these components in the projected solution to the noisy problem and thus obtain a good approximate and guaranteed symmetric solution for a smaller value of k using the odd basis vectors only.

6.3 Regularizing Krylov-Subspace Iterations

The advantage of the projection approach described above, with a fixed set of basis vectors, is that the operations involving these basis vectors can often be performed quickly and without the need to explicitly compute and store the basis. At the same time, the fixed basis is also the main disadvantage of this approach; the basis vectors are not adapted to the particular problem. On the other hand, the "optimal" set of basis vectors for a particular matrix A, namely, the singular vectors, are out of computational reach for large-scale problems.

6.3.1 The Krylov Subspace

To our rescue comes the *Krylov subspace* associated with A and b, defined as the span of powers of the matrix $A^T A$ applied to $A^T b$:

$$\boxed{\mathcal{K}_k \equiv \text{span}\{A^T b, (A^T A)\, A^T b, (A^T A)^2 A^T b, \ldots, (A^T A)^{k-1} A^T b\}.} \qquad (6.10)$$

The dimension of this subspace is at most k. The vectors in the definition of $\mathcal{K}_k$ are the iteration vectors of the power method for computing the largest eigenpair of a matrix, and they become increasingly richer in the direction of the principal eigenvector of $A^T A$, which is really the principal right singular vector v_1. Therefore, these vectors are not immediately useful as basis vectors for a practical implementation.

However, the subspace $\mathcal{K}_k$ itself does carry important information about the problem, and it is well suited to play the role of the subspace $\mathcal{W}_k$ in the projection method (6.8) for discrete inverse problems. We just need a better representation of its basis, and we could obtain such a basis by orthonormalizing the vectors in (6.10), starting from $A^T b$:

$$\begin{aligned}
&w_1 \leftarrow A^T b; \quad w_1 \leftarrow w_1/\|w_1\|_2;\\
&w_2 \leftarrow A^T A\, w_1; \quad w_2 \leftarrow w_2 - w_1^T w_2\, w_1; \quad w_2 \leftarrow w_2/\|w_2\|_2;\\
&w_3 \leftarrow A^T A\, w_2; \quad w_3 \leftarrow w_3 - w_1^T w_3\, w_1; \quad w_3 \leftarrow w_3 - w_2^T w_3\, w_2; \quad w_3 \leftarrow w_3/\|w_3\|_2;
\end{aligned}$$

and so forth. When we assume that the discrete Picard condition is satisfied, then the first basis vector w_1 is already rich in the direction of v_1. Due to the orthogonalization, this direction is purged from the second basis vector and, therefore, still due to the discrete Picard condition, w_2 is rich in the next few directions v_2, v_3, etc. The argument continues for all the orthonormal basis vectors w_i, and hence we obtain a basis which tends to be rich in the directions of $v_1, v_2, \ldots$, the principal right singular vectors.

Figure 6.6 illustrates this. We see that the vectors $A^T b$, $A^T A\, A^T b$, $(A^T A)^2 A^T b$, $\ldots, (A^T A)^{k-1} A^T b$ which define the Krylov subspace, become increasingly dominated by v_1. However, the orthonormal basis vectors $w_1, w_2, \ldots, w_k$ for the same subspace behave quite differently: each vector w_i is dominated by a few singular vectors v_j with index $j \approx i$. This illustrates that the Krylov subspace $\mathcal{K}_k$ indeed carries important information about the principal k right singular vectors. Hence, the Krylov subspace is a good choice for use in a projection method, and we emphasize that this fact hinges on the special property of the right-hand side for the ill-posed problem, namely, that it satisfies the discrete Picard condition.

The above approach to computing the basis vectors w_i is equivalent to the modified Gram–Schmidt algorithm for computing a QR factorization. It is natural to ask whether we can compute the vectors w_i more efficiently. The fact is that the well-known Lanczos tridiagonalization process, applied to the matrix $A^T A$ with starting vector $A^T b$, produces precisely these vectors in the form of the Lanczos vectors! Specifically, after k iterations the Lanczos algorithm has produced the matrix $W_k = (w_1, \ldots, w_k)$ and a symmetric tridiagonal matrix T_k such that

$$W_k^T\, A^T A\, W_k = T_k \qquad \text{for} \qquad k = 1, 2, \ldots. \qquad (6.11)$$

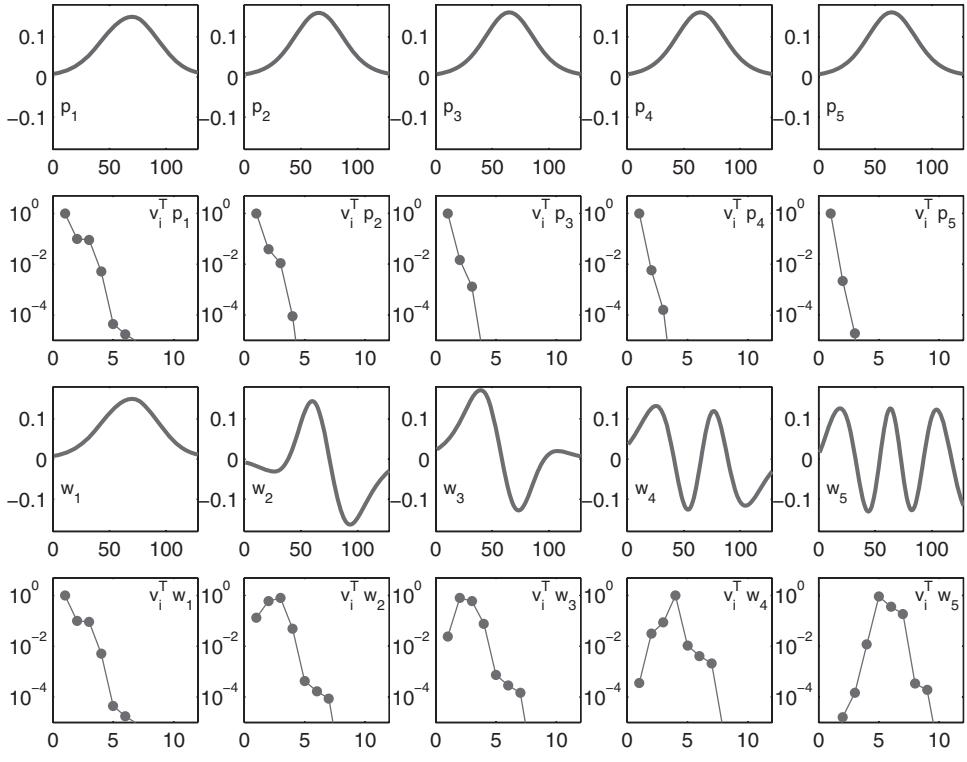

Figure 6.6. *Illustration of the vectors associated with the Krylov subspace* $\mathcal{K}_k = \text{span}\{A^T b, (A^T A)\, A^T b, (A^T A)^2 A^T b, \ldots, (A^T A)^{k-1} A^T b\}$. *The top row shows the first five normalized vectors* $p_i = (A^T A)^{i-1} A^T b / \|(A^T A)^{i-1} A^T b\|_2$ *that define the Krylov subspace, and the second row shows their expansion in the SVD basis; the vectors become increasingly dominated by* v_1. *The third row shows the first five orthonormal basis vectors* w_i *for the same subspace, and the fourth row shows these vectors' expansion in the SVD basis. Clearly,* w_1 *is dominated by* v_1, *while both* w_2 *and* w_3 *are dominated by* v_2 *and* v_3. *Then* w_4 *is dominated by* v_4, *and* w_5 *is dominated by* v_5, v_6, *and* v_7. *The overall behavior is that each vector* w_i *is dominated by a few singular vectors* v_j *with index* $j \approx i$. *We emphasize that this behavior of the Krylov subspace hinges on the problem satisfying the discrete Picard condition.*

Strictly speaking, this is true only if the computations are done in infinite precision; but we refrain from discussing the finite-precision aspects here. When applied to a symmetric positive definite matrix (here, $A^T A$), it can be shown that the Lanczos algorithm produces a tridiagonal matrix T_k whose larger eigenvalues converge to the larger eigenvalues of the matrix (here, the squared singular values of A). Therefore, this algorithm is typically used for computing some of the large eigenvalues and corresponding eigenvectors of a large sparse or structured matrix. Here we use it to produce the orthonormal basis vectors for the Krylov subspace $\mathcal{K}_k$.

6.3.2 The CGLS Algorithm

But there is more! The Lanczos process is closely related to the classical method of conjugate gradients (CG) for solving a system of linear equations with a symmetric positive definite coefficient matrix. In our case, this system is the so-called normal equations $A^T A x = A^T b$ associated with the unregularized least squares problem $\min_x \|Ax - b\|_2$, and one can show that the solution $x^{(k)}$ obtained after applying k steps of the CG algorithm to the normal equations $A^T A x = A^T b$, with the zero starting vector, is precisely the solution to the projected problem (6.8) with $\mathcal{W}_k = \mathcal{K}_k$ (in infinite precision). The key to the efficiency is that the CG algorithm computes this solution without the need to orthonormalize and store all the basis vectors $w_1, w_2, \ldots$ explicitly; only a few auxiliary vectors are needed.

The most stable way to implement the CG algorithm for the normal equations is known as the *CGLS algorithm;* see, e.g., [5, Section 7.4]. This algorithm goes back to the original CG paper by Hestenes and Stiefel [42], and it takes the following form:

$x^{(0)} = 0$ (starting vector)
$r^{(0)} = b - A x^{(0)}$
$d^{(0)} = A^T r^{(0)}$
for $k = 1, 2, \ldots$
  $\bar{\alpha}_k = \|A^T r^{(k-1)}\|_2^2 / \|A d^{(k-1)}\|_2^2$
  $x^{(k)} = x^{(k-1)} + \bar{\alpha}_k d^{(k-1)}$
  $r^{(k)} = r^{(k-1)} - \bar{\alpha}_k A d^{(k-1)}$
  $\bar{\beta}_k = \|A^T r^{(k)}\|_2^2 / \|A^T r^{(k-1)}\|_2^2$
  $d^{(k)} = A^T r^{(k)} + \bar{\beta}_k d^{(k-1)}$
end

It is possible to use other starting vectors, but the zero vector is certainly the most common for discrete inverse problems. In this case we have the following important characterization of the CGLS solution:

$$\boxed{x^{(k)} = \operatorname{argmin}_x \|Ax - b\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{K}_k.} \tag{6.12}$$

Note that the algorithm requires one multiplication with A and one multiplication with A^T per iteration (to compute $A d^{(k-1)}$ and $A^T r^{(k)}$, respectively). Except for a normalization and perhaps a sign change, the "search vectors" $d^{(1)}, \ldots, d^{(k)}$ of the CG algorithm are equal to the orthonormal basis vectors $w_1, \ldots, w_k$ for the Krylov subspace $\mathcal{K}_k$. The solution norm $\|x^{(k)}\|_2$ and residual norm $\|Ax^{(k)} - b\|_2$ are monotonic functions of k,

$$\|x^{(k)}\|_2 \geq \|x^{(k-1)}\|_2, \qquad \|Ax^{(k)} - b\|_2 \leq \|Ax^{(k-1)} - b\|_2, \qquad k = 1, 2, \ldots,$$

allowing us to use the L-curve method as a stopping criterion for this method.

A word on our notation: Since CGLS is an iterative method, we could also use the notation $x^{[k]}$ with brackets for the iteration vectors; instead we have chosen to use $x^{(k)}$ with parentheses to emphasize the connection to the projection methods.

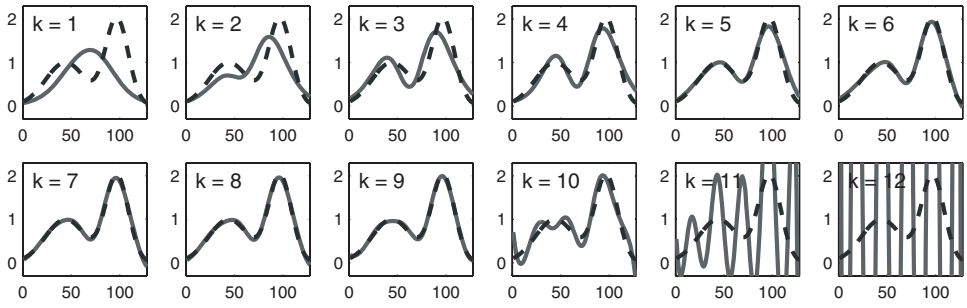

Figure 6.7. *Solid curves: The first* 12 *CGLS solutions* $x^{(k)}$ *for* $k = 1, \ldots, 12$*, for the same test problem as in Figure* 6.4*. Dashed curves: The exact solution. A good approximation to* x^{exact} *is achieved at* 9 *iterations.*

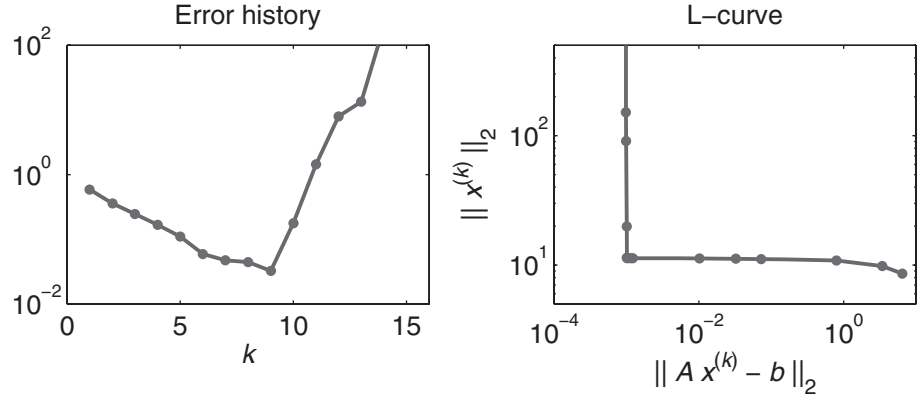

Figure 6.8. *Left: The error history* $\|x^{(k)} - x^{\text{exact}}\|_2 / \|x^{\text{exact}}\|_2$ *for the CGLS solutions shown in Figure* 6.7*. The semiconvergence is obvious: In the first* 9 *iterations the error decreases, from which point it increases dramatically. Right: The L-curve for the CGLS solutions.*

To illustrate the regularizing effects of the CGLS algorithm, we applied this algorithm to the same test problem as in the previous section, namely, the `shaw` problem with $n = 128$. Figure 6.7 shows the CGLS iterates $x^{(k)}$ for $k = 1, 2, \ldots, 12$. We see that the Krylov subspace $\mathcal{K}_k$ provides a good basis for this problem, and the best approximation to x^{exact} is achieved at 9 iterations.

To further illustrate the semiconvergence of the CGLS method, Figure 6.8 shows the error history, that is, the relative error $\|x^{(k)} - x^{\text{exact}}\|_2 / \|x^{\text{exact}}\|_2$ as a function of the number k of iterations. This is a clear example of semiconvergence: the error decreases (in this case, monotonically) until it reaches a minimum at 9 iterations, after which the error increases dramatically when the inverted noise starts to dominate the solution. Figure 6.8 also shows the discrete L-curve for the CGLS solutions, again reflecting the semiconvergence.

We mentioned in the previous section that the solution to a projection problem cannot, in general, be expressed as a filtered SVD solution. To see that the CGLS solution $x^{(k)}$ indeed has a filtered SVD expansion, note that since $x^{(k)}$ is bound to lie in the Krylov subspace $\mathcal{K}_k$ (6.10), it must be a linear combination of the Krylov basis vectors $A^T b, (A^T A)\, A^T b, \ldots, (A^T A)^{k-1} A^T b$, and hence there exist constants $c_1, c_2, \ldots, c_k$ such that

$$x^{(k)} = c_1\, A^T b + c_2\, (A^T A)\, A^T b + c_3\, (A^T A)^2 A^T b + \cdots + c_k\, (A^T A)^{k-1} A^T b.$$

We now insert the SVD of A into this expansion and use that $A^T A = V \Sigma\, U^T U \Sigma\, V^T = V \Sigma^2 V^T$:

$$\begin{aligned} x^{(k)} &= \big(c_1 + c_2\, V \Sigma^2 V^T + c_3\, V \Sigma^4 V^T + \cdots + c_k\, V \Sigma^{2(k-1)} V^T\big) V \Sigma\, U^T b \\ &= V \left(c_1\, \Sigma^2 + c_2\, \Sigma^4 + c_3\, \Sigma^6 + \cdots + c_k\, \Sigma^{2k}\right) \Sigma^{-1} U^T b \\ &= V\, \Phi^{(k)}\, \Sigma^{-1} U^T b, \end{aligned}$$

in which $\Phi^{(k)} = \operatorname{diag}\big(\varphi_1^{(k)}, \ldots, \varphi_n^{(k)}\big)$ is a diagonal matrix with the CGLS filter factors

$$\varphi_i^{(k)} = c_1\, \sigma_i^2 + c_2\, \sigma_1^4 + c_3\, \sigma_1^6 + \cdots + c_k\, \sigma_i^{2k}, \qquad i = 1, \ldots, n.$$

This confirms that the CGLS solution has a filtered SVD expansion, and it shows that the CGLS filter factors are polynomials in the squared singular values σ_i^2. These polynomials can be expressed in terms of the eigenvalues of the symmetric tridiagonal matrix T_k in (6.11), and there are also (potentially unstable) recursion formulas to evaluate the polynomials, which are used in *Regularization Tools*'s implementation `cgls`. It is beyond the scope of this book to go into these details, and we refer the reader to [32, Chapter 6] and [71] for details and references.

6.3.3 CGLS Focuses on the Significant Components

We have seen that the TSVD method, by definition, includes all the first k SVD components in the regularized solution x_k, and that the Tikhonov solution x_λ also includes all the leading SVD components until the filter factors $\varphi_i^{[\lambda]}$ becomes effective. All these SVD components are included whether they are needed or not. The SSVD method (4.6), on the other hand, includes only those SVD components that are needed, and we demonstrated in Section 4.3 that this can be advantageous.

This important feature carries over to the CGLS method, due to the facts that its basis vectors are those of the Krylov subspace, and not the singular vectors. Recall that the Krylov subspace $\mathcal{K}_k$ (6.10) is defined in terms of both the matrix A and the right-hand side b, while the SVD depends on A only. Thus the Krylov subspace $\mathcal{K}_k$ will adapt itself in an optimal way to the specific right-hand side, while the SVD basis is optimal if no information about b is given.

To illustrate this feature of the Krylov subspace we applied the CGLS algorithm to the `phillips` test problem (which was also used in Section 6.2). Due to the symmetries in both the matrix and the exact solution, many of the SVD coefficients

$v_i^T x^{\text{exact}}$ of the exact solution are zero. The Picard plot in the top part of Figure 6.9 reveals that about every second SVD component $v_i^T x^{\text{exact}}$ is zero. Similarly, the Picard plot for the noisy problem (shown in the bottom half of Figure 6.9) shows that the corresponding solution coefficients $u_i^T b/\sigma_i$ are quite small—of the order 10^{-3}.

For this reason, the TSVD method with truncation parameter k, which insists on including *all* SVD components from 1 to k, includes about twice the number of components in x_k that are necessary, because half of them are so small. This is reflected in the error histories for the TSVD solutions: about every second increment of k leaves the error unchanged, when a small SVD component is included, for both the noise-free and the noisy problem.

The Krylov subspace, on the other hand, is constructed from the starting vector $A^T b$ which has the SVD expansion

$$A^T b = A^T (A x^{\text{exact}} + e) = \sum_{i=1}^{n} \sigma_i^2 \, (v_i^T x^{\text{exact}}) \, v_i + \sum_{i=1}^{n} \sigma_i \, (u_i^T e) \, v_i.$$

Hence, about half of the SVD coefficients of this vector are solely due to the noise components, and therefore these components are small (they are zero for the ideal noise-free problem). Consequently, the Krylov space $\mathcal{K}_k$ is dominated by precisely those singular vectors that contribute the most to the solution, and only the needed SVD directions are well represented in the Krylov subspace. For this reason, we can expect that the CGLS solution $x^{(k)}$ is a better approximation to x^{exact} than the corresponding TSVD solution with the same value of k (because the latter includes small and unnecessary components).

All these observations are confirmed by the results in Figure 6.9, which shows both the TSVD and the CGLS solutions for a noise-free and a noisy problem. For the noise-free problem, we see that the error decreases much more rapidly for the CGLS solution than the TSVD solution. The same overall behavior is observed for the noisy problem; but the decreases of the error in the CGLS solution is slower here, because some unnecessary SVD components are "picked up" in the Krylov subspace, due to the noise, for $k = 5$, 6 and 7.

We emphasize that for a realistic large-scale problem, the situation is typically more complicated than we have illustrated here with a small test problem. However, the overall behavior is the same; the Krylov subspace $\mathcal{K}_k$ tends to be a good subspace for projecting the solution, and the subspace tends to "pick up" the most important SVD directions first.

6.3.4 Other Iterations—MR-II and RRGMRES*

With the success of regularizing iterations in the form of the CGLS algorithm, it is natural to seek iterative regularization methods based on other Krylov subspaces. For example, the matrix A may be symmetric, in which case there is obviously no need for its transpose. Or A may represent the discretization of a linear operator for which it is difficult or inconvenient to write a black-box function for multiplication with A^T. Hence there is an interest in iterative methods that do not use multiplication with the transposed matrix.

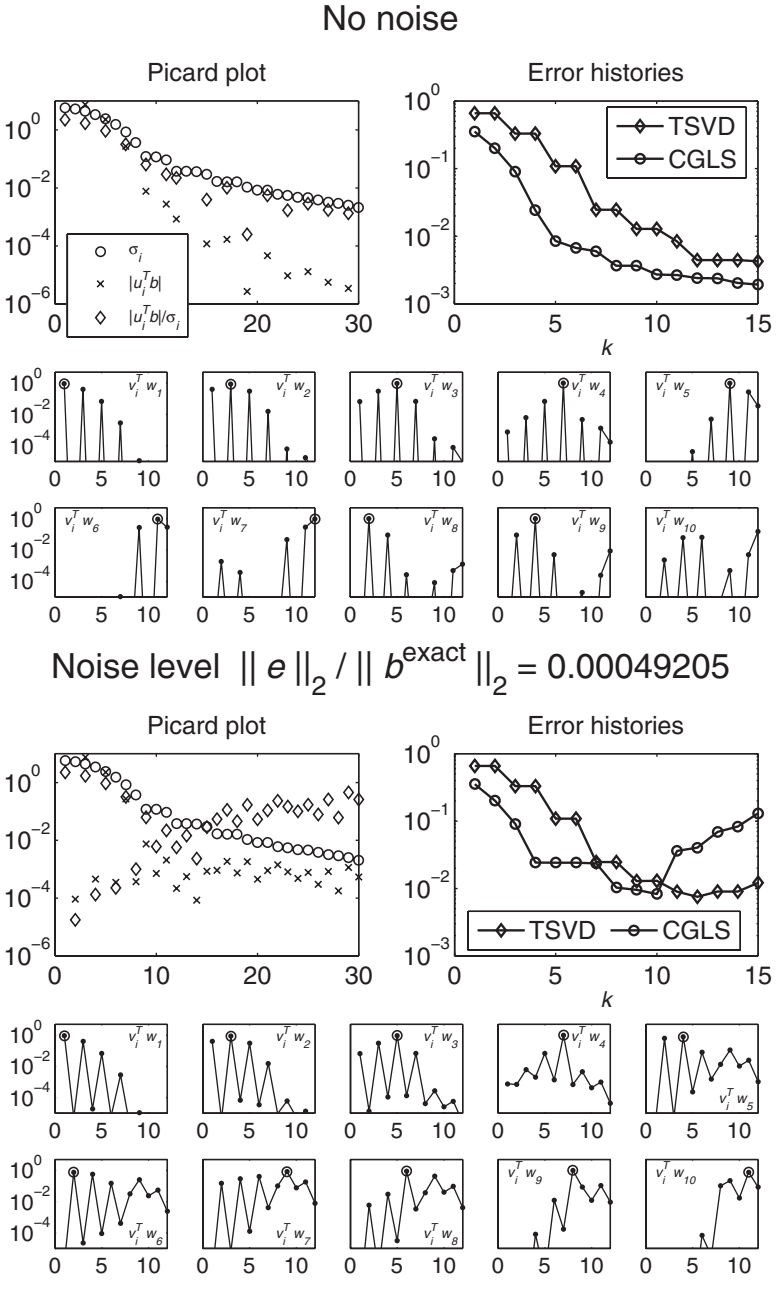

Figure 6.9. *Insight into the behavior of TSVD and CGLS applied to a problem with many coefficients $v_i^T x^{\text{exact}} = 0$, with no noise and with a small noise level. For both situations we show the Picard plot, the error histories for TSVD and CGLS, and the coefficients of the Krylov/CGLS basis vectors w_k in the SVD basis; the largest value in absolute values is marked by a circle.*

Two well-known methods that immediately come to mind are MINRES, designed for symmetric indefinite matrices, and GMRES, designed for nonsymmetric square matrices; both are based on the Krylov subspace $\text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$. Unfortunately this subspace has a big disadvantage: it includes the noisy right-hand side $b = b^{\text{exact}} + e$ and thus the noise component e. This means that the solution, obtained as a linear combination of the vectors $b, Ab, A^2b, \ldots$, is likely to include a large noise component.

The solution to this problem is to work with the "shifted" Krylov subspace that starts with the vector Ab, i.e.,

$$\vec{\mathcal{K}}_k = \text{span}\{Ab, A^2b, \ldots, A^kb\}. \tag{6.13}$$

The advantage of this subspace is that the noise component is now multiplied with A, which, according to our analysis, has a smoothing effect and thus dampens the high-frequency components of the noise. The associated iterative algorithms based on $\vec{\mathcal{K}}_k$ are called MR-II and RRGMRES; they are included in *Regularization Tools* as functions `mr2` and `rrgmres`.

The performance and the regularizing properties of MR-II and RRGMRES are carefully studied in [46], where details can be found. Here we summarize the main results of the analysis.

- The absence of the vector b in the Krylov subspace $\vec{\mathcal{K}}_k$ (6.13) is essential for the use of MR-II and RRGMRES for discrete inverse problems.
- MR-II is a spectral filtering method. Negative eigenvalues of A do not inhibit the regularizing effect, but they can deteriorate the convergence rate.
- RRGMRES mixes the SVD components in each iteration, and thus it does not provide a filtered SVD solution. The method can work well
  - if the mixing of components is weak (e.g, if A is nearly symmetric), or
  - if the Krylov basis vectors are well suited for the problem.

  Otherwise RRGMRES fails to produce regularized solutions, either due to the undesired mixing of the SVD components or due to an unfavorable null space.

In conclusion, MR-II is a competitive alternative to CGLS for symmetric indefinite matrices, while for nonsymmetric matrices there is currently no transpose-free iterative regularization algorithm that can be used as a black-box method, in the same fashion as CGLS. Exercise 6.7 demonstrates that RRGMRES can perform well but is not guaranteed to do so.

6.4 Projection + Regularization = Best of Both Worlds*

In principle the Krylov subspace $\mathcal{K}_k$ should always provide a good basis for a projection method, but in practice, this is not always the case. The main reason is that the

Krylov subspace is generated on the basis of the given, noisy right-hand side b (and not the exact data b^{exact}), and for this reason the Krylov subspace tends to include both the desired basis vectors and, due to the noise, some basis vectors that are not so important for the regularized solution. The fact that we always work with finite-precision arithmetic has essentially the same effect.

The consequence is that we cannot be sure that all the important basis vectors have emerged in the Krylov subspace before some undesired basis vectors appear. In other words, while we experience semiconvergence, there is no guarantee that we have obtained the minimum error in $x^{(k)}$ at the end of the semiconvergence stage of the iterations. As a result, the projected solution may not be a good regularized solution, because it may include noisy components in the directions of the undesired vectors while missing some important components.

The solution to this dilemma is to combine the Krylov method with a regularization method applied to the low-dimensional projected problem. In the setting of Section 6.2, we apply regularization to the least squares problem in (6.8) and (6.9) with $\mathcal{W}_k = \mathcal{K}_k$. In the important case of Tikhonov regularization, the regularized projected problem then takes the form

$$x_\lambda^{(k)} = W_k\, y_\lambda^{(k)}, \qquad y_\lambda^{(k)} = \operatorname{argmin}_y \left\{ \|(A W_k)\, y - b\|_2^2 + \lambda^2 \|y\|_2^2 \right\}. \tag{6.14}$$

It is interesting to note that if the matrix W_k has orthonormal columns, which is always recommended for reasons of numerical stability, then the solution $x_\lambda^{(k)}$ to the Tikhonov-regularized projected problem in (6.14) is identical to the solution obtained by applying the same projection method to the Tikhonov problem (4.9), i.e.,

$$x_\lambda^{(k)} = \operatorname{argmin}_x \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 \qquad \text{s.t.} \qquad x \in \mathcal{W}_k.$$

To see this, simply note that if we insert $x = W_k\, y$, then the problem becomes

$$\begin{aligned} \min_y \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} W_k\, y - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 &= \min_y \left\| \begin{pmatrix} A W_k \\ \lambda W_k \end{pmatrix} y - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 \\ &= \min_y \left\{ \|(A W_k)\, y - b\|_2^2 + \lambda^2 \|W_k\, y\|_2^2 \right\}, \end{aligned}$$

which is equivalent to (6.14) because $\|W_k\, y\|_2 = \|y\|_2$ when $W_k^T W_k = I$. Hence, there is no distinction here between the two approaches "first-regularize-then-project" and "first-project-then-regularize"; see Figure 6.10. The latter is better suited for large-scale problems, because the choice of the regularization parameter is associated with the small projected problem.

Due to the adaptive nature of the Krylov subspace $\mathcal{K}_k$, we want to use the orthonormal basis vectors for this space as the columns of the matrix W_k. There are three ways to generate these vectors:

- We can normalize the "search vectors" $d^{(k)}$ of the CGLS algorithm and set $w_k = d^{(k)} / \|d^{(k)}\|_2$.
- We can apply the iterative Lanczos tridiagonalization algorithm to the matrix $A^T A$ to obtain the matrix W_k; cf. (6.11).

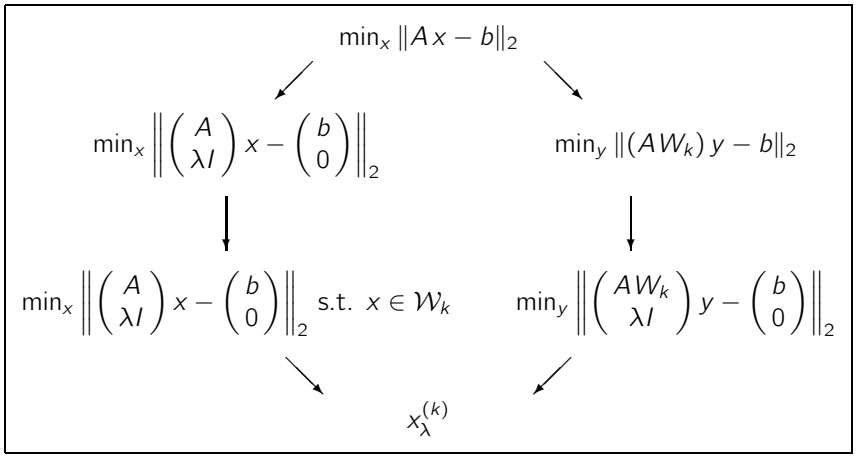

Figure 6.10. *The two approaches "first-regularize-then-project" (left) and "first-project-then-regularize" (right) produce the same solution.*

- We can apply a similar iterative bidiagonalization algorithm directly to the matrix A to again obtain W_k.

The third approach, which is the most elegant and also the most stable, is called Golub–Kahan bidiagonalization or Lanczos bidiagonalization (see Section 7.6.1 in [5]), and when started with the right-hand side b this iterative method takes the following simple form, where the quantities α_k and β_k are chosen such that the corresponding vectors w_k and z_k are normalized:

$w_0 = 0$
$\beta_1 z_1 = b$
for $k = 1, 2, \ldots$
  $\alpha_k w_k = A^T z_k - \beta_k w_{k-1}$
  $\beta_{k+1} z_{k+1} = A w_k - \alpha_k z_k$
end

After k iterations, this remarkably simple algorithm has produced two matrices $W_k \in \mathbb{R}^{n\times k}$ and $Z_k \in \mathbb{R}^{m\times(k+1)}$ with orthonormal columns,

$$W_k = (w_1, w_2, \ldots, w_k), \qquad Z_{k+1} = (z_1, z_2, \ldots, z_{k+1}),$$

and a lower bidiagonal matrix $B_k \in \mathbb{R}^{(k+1)\times k}$,

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix},$$

such that

$$AW_k = Z_{k+1}\, B_k. \tag{6.15}$$

The columns of W_k are the desired basis vectors for the Krylov subspace $\mathcal{K}_k$, and if we insert these relations into (6.11), then we get $T_k = B_k^T B_k$.

When we insert the relation (6.15) into the projected problem (6.9), then we see that the least squares residual can be reformulated as follows:

$$\begin{aligned}\|(AW_k)\, y - b\|_2^2 &= \|Z_{k+1}\, B_k\, y - b\|_2^2 \\ &= \|Z_{k+1}\, B_k\, y - Z_{k+1}\, Z_{k+1}^T b + (I - Z_{k+1}\, Z_{k+1}^T)\, b\|_2^2.\end{aligned}$$

The two vectors $Z_{k+1}\, B_k\, y - Z_{k+1}\, Z_{k+1}^T b$ and $(I - Z_{k+1}\, Z_{k+1}^T)\, b$ are orthogonal. Thus, with $\rho_0 = \|(I - Z_{k+1}\, Z_{k+1}^T)\, b\|_2$, we get

$$\begin{aligned}\|(AW_k)\, y - b\|_2^2 &= \|Z_{k+1}\, (B_k\, y - Z_{k+1}^T b)\|_2 + \rho_0^2 \\ &= \|B_k\, y - \beta_1 e_1\|_2^2 + \rho_0^2.\end{aligned}$$

Here, $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{k+1}$, and we have used that $z_1 = b/\beta_1$. Hence, via the bidiagonalization algorithm, we arrive at the alternative formulation of (6.14):

$$x_\lambda^{(k)} = W_k\, y_\lambda^{(k)}, \qquad y_\lambda^{(k)} = \operatorname{argmin}_y \left\{ \|B_k\, y - \beta_1 e_1\|_2^2 + \lambda^2\, \|y\|_2^2 \right\}. \tag{6.16}$$

The projected Tikhonov problem $\|B_k\, y - \beta_1 e_1\|_2^2 + \lambda^2\, \|y\|_2^2$ can be solved in $O(k)$ operations, due to the bidiagonal structure of B_k, and thus we can afford to solve this problem for several values of the regularization parameter λ in search of a good value. See the discussion in [49] for more details about this approach.

Another important feature of (6.16) is that the solution norm $\|x_\lambda^{(k)}\|_2$ and the residual norm $\|A\, x_\lambda^{(k)} - b\|_2$ can be obtained directly from the Tikhonov solution $y_\lambda^{(k)}$ to the projected problem. Specifically, we have

$$\|x_\lambda^{(k)}\|_2 = \|W_k\, y_\lambda^{(k)}\|_2 = \|y_\lambda^{(k)}\|_2, \qquad \|A\, x_\lambda^{(k)} - b\|_2 = \|B_k\, y_\lambda^{(k)} - \beta_1 e_1\|_2.$$

The key observation here is that the solution and residual norms associated with $x_\lambda^{(k)}$ can be computed directly from $y_\lambda^{(k)}$ and B_k. Hence, we can afford to use a parameter-choice method based on these norms, such as the discrepancy principle or the L-curve criterion, by applying it to the regularized projected problem. The back-transformation, via the multiplication with W_k, is only done once.

A complete algorithm based on these ideas—occasionally referred to as a *hybrid method*—could take the following generic form. Run the bidiagonalization algorithm for $k = 1, 2, 3, \ldots$ and store the matrices B_k and W_k. For each 10th iteration, say, form the projected problem and apply Tikhonov regularization with a parameter-choice rule to this problem to arrive at the parameter $\lambda^{(k)}$. Once the regularization parameter tends to "settle down" and stay almost the same over several value of k, this is a sign that the dimension of the Krylov subspace is large enough to have captured all relevant information and that the appropriate amount of regularization has been determined. Then, finally, compute the solution via multiplication with W_k.

For very large problems it may be intractable to store the matrix W_k. For such problems one must skip the last step of the algorithm outlined above and instead use

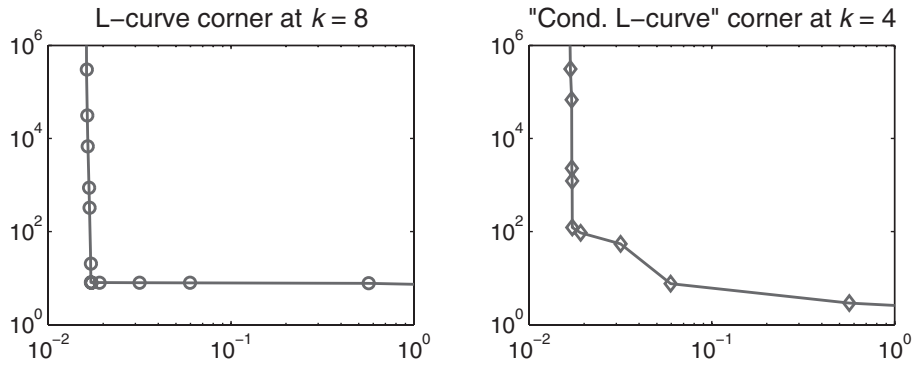

Figure 6.11. *Left: The L-curve for CGLS solutions computed via bidiagonalization. Right: The "condition L-curve"* $(\log\|b - Ax^{(k)}\|_2, \log \mathrm{cond}(B_k))$ *for the same solutions; the corner is not located at the optimal* k*.*

the CGLS algorithm to solve the original Tikhonov problem, using the regularization parameter $\lambda^{(k)}$ determined above.

For hybrid methods it has been suggested to use a "condition L-curve" criterion that locates the corner of the curve $(\log\|b - Ax^{(k)}\|_2, \log \mathrm{cond}(B_k))$. This is obviously a bad idea, since the condition number of B_k increases steadily, while the solution norm $\|x^{(k)}\|_2$ starts to increase only when the inverted noise starts to dominate. Figure 6.11 shows that the "condition L-curve" gives much too small a value of the number of iterations.

The precise details of this hybrid approach depend, of course, on the particular problem. Also, it appears that computing a good regularization parameter via the projected problem may not be straightforward. A successful algorithm based on weighted GCV (5.12), which adaptively determines a good weighting factor, is described in [10]; MATLAB software is also available. More insight into the way that the noise appears in the Krylov subspace and the projected problem can be found in [43].

6.5 The Story So Far

For many large-scale problems the use of iterative methods is the only way to compute a regularized solution (exceptions are structured problems such as those we present in the next chapter). In these problems, the matrix may be sparse, or it may be implicitly available through a function that computes matrix-vector products. A straightforward approach is to use an iterative method to solve the Tikhonov least squares problem, but this can be a cumbersome approach if we need to try many values of the regularization parameter.

This chapter focuses on iterative regularization methods where the number of iterations plays the role of the regularization parameter. These methods exhibit semiconvergence: initially the iterates approach the exact solution, after which they start to deviate from it again and instead converge to the undesired naive solution.

Two classes of methods are covered here: classical stationary iterative methods and Krylov subspace methods, the latter based on the concept of projection methods. Several of these iterative methods are spectral filtering methods and thus fall into the general framework established in the previous chapters; but a few (namely, ART and RRGMRES) are not, which severely complicates the analysis of semiconvergence.

We find that the CGLS method (a dedicated implementation of the CG algorithm for linear least squares problems) can be considered the general-purpose "workhorse" for large-scale problems. It always exhibits semiconvergence, and it tends to focus on the significant components of the solution (similar to the SSVD method). Other iterative methods may be successful for specific problems.

Hybrid methods combine the ideas of projection and regularization, and they make the Krylov-subspace paradigm underlying CGLS even more useful and robust. This is achieved by utilizing the adaptive nature of the Krylov subspace to produce a projected problem of lower dimensions which seeks to capture the essential information in the data, and which can then be treated by any of the medium-scale methods from Chapters 4 and 5.

Exercises

6.1. **Verification of the Landweber Filter Factors**
Use the SVD of A to verify, by insertion, that (6.1) is satisfied when the Landweber iterate $x^{[k]}$ is given by $x^{[k]} = V\,\Phi^{[k]}\,\Sigma^{-1}U^T b$ with filter factors $\varphi_i^{[k]} = 1 - (1 - \omega\,\sigma_i^2)^k$. Also show that for $\omega\,\sigma_i^2 \ll 1$ we have $\varphi_i^{[k]} \approx k\,\omega\,\sigma_i^2$ (and thus the same decay as the Tikhonov filter factors).

6.2. **Landweber Iteration for Nonnegative Solutions**
The purpose of this exercise is to illustrate the use of the augmented Landweber iteration (6.3) where the operator $\mathcal{P}$ represents a nonnegativity constraint. In other words, if x and $y = \mathcal{P}(x)$ are vectors, then

$$y_i = \begin{cases} x_i, & x_i \geq 0, \\ 0, & x_i < 0. \end{cases}$$

The test problem to be used here is a slightly modified version of the `phillips` problem from *Regularization Tools*:

```
[A,b,x] = phillips(n);
x = x - 0.5;
x(x<0) = 0;
b = A*x;
```

This produces an exact solution `x` which has a smooth "blob" in the middle, surrounded by zeros.

Implement two versions of Landweber's method: the standard version (6.1) and the augmented version (6.3) with the above nonnegativity projection $\mathcal{P}$. Use both methods to solve the modified `phillips` problem; you should

perform several hundred iterations, and we suggest that you use a value of ω slightly smaller than $2/\sigma_1^2$. Compute the error histories for the two methods, study the regularized solutions, and comment on the results.

6.3. **Illustration of the CGLS Algorithm**
The purpose of this exercise is to illustrate the use of regularizing iterations in the form of the CGLS algorithm from Section 6.3.2. This algorithm is implemented in *Regularization Tools* as the function `cgls`. The exercise also previews material coming up in Section 7.5.

The model problem is a small image deblurring problem which is generated by means of the function `blur` from the same package. Both the exact image and the blurred image are $N \times N$, and they are represented by N^2-vectors by stacking their columns. Use `reshape(x,N,N)` to get from the "stacked" representation of the image to the image itself. To display[5] the images, you can use the function `imagesc` together with the commands `axis image` and `colormap gray`.

Choose $N = 64$ and generate the test problem with `blur(N,10,1.4)` (the second parameter controls the sparsity of A, and the third parameter controls the amount of blurring). Plot the sharp and the blurred $N \times N$ images. At this stage, do not add noise to the problem. Perform a number of CGLS iterations; note that the CGLS iterates are returned as columns of the output matrix from `cgls`. Plot some of the iterates as images, and notice how the reconstruction gets sharper as the number of iterations increases.

Now add noise e to the blurred image with relative noise level $\|e\|_2/\|b\|_2 = 0.1$, and repeat the CGLS computations. You should notice that after a certain number of steps, the noise starts to dominate. This illustrates that the number of iterations indeed plays the role of the regularization parameter for the CGLS algorithm.

6.4. **The L-Curve Criterion for Regularizing Iterations**
This exercise illustrates the use of the L-curve criterion applied to an iterative regularization method. We use the `blur` test problem and the CGLS method from the previous exercise.

Plot the discrete L-curve for the CGLS solutions; you should note that the residual and solution norms vary slowly with the number k of iterations. Why do you think this is so?

Try to locate the corner of the L-curve, either by visual inspection or using the function `l_corner` from *Regularization Tools*. Compare with that solution which is closest to x^{exact}. Does the L-curve criterion work for this problem and this method?

How does the performance of the L-curve criterion depend on the relative noise level and the amount of blurring (controlled by the third input parameter to `blur`)?

6.5. **CGLS Versus TSVD**
In this example we consider the test problem that was used in Section 4.3 to illustrate the SSVD method. The matrix is from the `deriv2` test problem, and

[5] If the Image Processing Toolbox is available, use `imshow(X,[])` to display image `X`.

the exact solution is a "sawtooth function" $\wedge\!\vee$ constructed such that only each fourth SVD component is nonzero. The following code constructs this solution and the corresponding right-hand side, assuming that the problem dimension `n` is a multiple of 4:

```
nq = n/4;
x1 = (0:nq-1)' + 0.5;
x2 = (nq:-1:1)' - 0.5;
xex = [x1;x2;-x1;-x2];
bex = A*xex;
```

Your task is to compare the solutions computed by means of TSVD and CGLS, for $k = 1, \ldots, 50$, using the functions `tsvd` and `cgls` from *Regularization Tools*. Use the same noise level $\eta = 10^{-5}$ as in Section 4.3. For CGLS, try to use both the standard implementation with three input parameters and an augmented implementation with four input parameters that specifically reorthogonalize each vector $d^{(k)}$ to the previous vectors. The latter implementation, which is computationally expensive, simulates the performance of CGLS in infinite precision.

Plot the error histories for the three methods, and comment on the results. Which method gives the most accurate solution?

6.6. **The NCP Criterion for Regularizing Iterations**
The implementation of the NCP criterion in *Regularization Tools*, in the function `ncp`, computes the residual vectors given the right-hand side b and the SVD of A. This is not suited for iterative regularization methods, and your task is to write an implementation of the NCP criterion, based on `ncp`, that takes a number of residual vectors as input (e.g., in the form of a matrix whose columns are the residual vectors). Your function must find that residual vector whose NCP is closest to a straight line.

Use your function on the image deblurring test problem from the previous exercise, and compare with that solution which is closest to x^{exact}. Does the NCP criterion work for this problem? How does the performance of this criterion depend on the relative noise level and the amount of blurring (controlled by the third input parameter to `blur`)?

6.7. **Which is Better: CGLS or RRGMRES?***
This exercise illustrates some of the results in [46] regarding the regularizing properties of CGLS and RRGMRES. Your task is to compare the two methods for solving the two test problems `baart` and `i_laplace` from *Regularization Tools*. We suggest you use the problem dimension $n = 64$, the very small noise level $\eta = 10^{-8}$, and a total of `kmax = 16` iterations for both methods.

Since RRGMRES always orthogonalizes the basis vectors for the Krylov subspace $\vec{\mathcal{K}}_k$ (6.13), the comparison is most fair if you also use reorthogonalization in CGLS (set `cgls`'s fourth input parameter to `1`).

Compare the error histories for the two methods. For each of the two test problems, which iterative method is better?

If you want to explain the convergence histories in detail, you need to study the expansion of the exact solution x^{exact} in the orthonormal basis vectors of

the two Krylov subspaces $\mathcal{K}_k$ (6.10) and $\vec{\mathcal{K}}_k$ (6.13). You can compute the first basis as the third output argument from `lanc_b(A,b,16,2)`, where the fourth input argument enforces accurate reorthogonalization of these vectors. The following code computes an accurate basis for the second basis:

```
Khat = A*b; Khat = Khat/norm(Khat);
for i=2:kmax
    v = A*K(:,i-1);
    for j=1:i-1, v = v - (K(:,j)'*v)*K(:,j); end
    for j=1:i-1, v = v - (K(:,j)'*v)*K(:,j); end
    K(:,i) = v/norm(v);
end
```

Two orthogonalizations are needed in each loop to obtain the necessary accuracy.

Chapter 7

Regularization Methods at Work: Solving Real Problems

The purpose of this chapter is to introduce several realistic problems and, along with these, to discuss various issues related to solving real problems with regularization algorithms: deconvolution, matrix structure, fast algorithms, boundary conditions, inverse crime, and resolution. We finish with a more formal discussion of what we understand by a working regularization algorithm. Don't miss this chance to learn more about the practical treatment of discrete inverse problems.

7.1 Barcode Reading—Deconvolution at Work

A barcode is the series of alternating black and white stripes that can be found on almost every item sold in stores today. In these high-contrast images, the relative widths of the bars encode information, typically a string of letters and/or numbers that identifies the item. For example, in the Universal Product Code used on grocery items, five digits represent the manufacturer's number, while another five digits represent the product number assigned by the manufacturer.

A barcode reader (or scanner) is a device for reading the printed barcodes, and it consists of a light source, a lens, and a photo conductor translating optical impulses into electrical ones. A mathematical model of the barcode reading process takes its basis in a piecewise constant function $f(t)$ that represents the intensity of the printed barcode along a line perpendicular to the black and white bars. Ideally this function takes the form shown in the top of Figure 7.1.

Due to imperfections in the scanner (which is a mass-produced device) we cannot record the exact signal $f(t)$ but rather a blurred version $g(s)$, such as the one shown in the middle of Figure 7.1. In our model, we assume that $f(t)$ and $g(s)$ are related by

$$\int_0^1 \exp\left(-\frac{(s-t)^2}{\varsigma^2}\right) f(t)\,dt = g(s), \qquad 0 \le s \le 1, \tag{7.1}$$

where the s axis represents the one-dimensional photo conductor inside the scanner. We emphasize that this is not the correct model of the blurring inside a barcode reader,

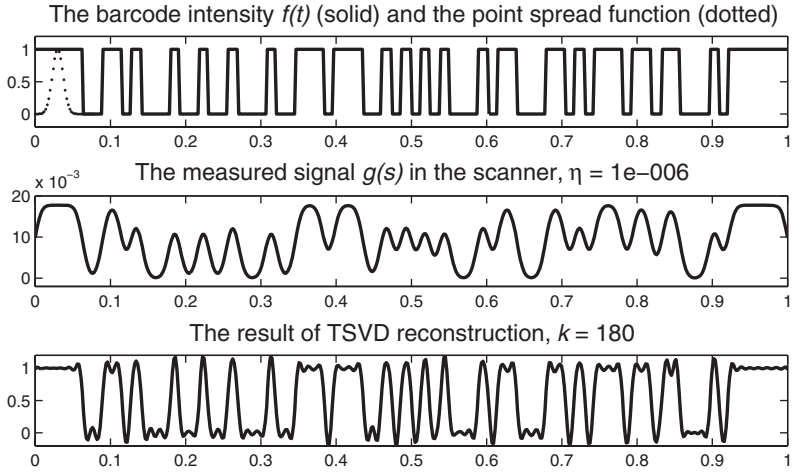

Figure 7.1. *Barcode reading example with $n = 500$ and $\varsigma = 0.01$. Top: The intensity of the printed barcode, together with the point spread function. Middle: The recorded signal. Bottom: TSVD solution for $k = 180$.*

but it serves as a good approximation. Obviously, (7.1) is a special instance of the Fredholm integral equation (2.2), with the kernel given by

$$K(s,t) = \exp\left(-\frac{(s-t)^2}{\varsigma^2}\right), \qquad s,t \in [0,1]. \tag{7.2}$$

This function represents the blurring of the image that takes place inside the scanner, and the parameter ς controls the amount of blurring (the larger the ς the narrower the Gaussian peak in (7.2) and thus less blurring). Note that we have omitted an arbitrary scaling in our definition of $K(s,t)$.

Kernels of this form, that depend only on the difference $s - t$ between the two independent variables, are called convolution kernels, and we say that the measured signal $g(s)$ is computed as the *convolution*

$$\int_0^1 h(s-t)\,f(t)\,dt = g(s), \qquad 0 \le s \le 1, \tag{7.3}$$

between the exact signal $f(t)$ and the point spread function $h(t)$, in this case given by

$$h(t) = \exp(-t^2/\varsigma^2), \qquad t \in \mathbb{R}.$$

Point spread functions typically peak at $t = 0$ and decay away from this peak such that we can think of $h(t)$ as practically located near the peak. While this point spread function is never identically zero, for $\varsigma = 0.01$ it satisfies $h(t) < 10^{-4}$ for $|t| > 0.03$, and thus for all practical purposes we can consider it to be zero here. On top of Figure 7.1 we also show the point spread function of the problem considered, and we

recognize the overall width and shape of this function in the measured signal in the middle of the figure.

For obvious reasons, the process of computing the function $f(t)$ given the measured signal $g(s)$ and the point spread function $h(t)$ is called *deconvolution*. Such problems arise in an abundance of technical applications. As we shall see, it is advantageous to discretize deconvolution problems by means of the midpoint quadrature rule with n quadrature and collocation points given by

$$s_i = (i - \tfrac{1}{2})/n, \qquad t_j = (j - \tfrac{1}{2})/n, \qquad i, j = 1, \ldots, n.$$

Then the elements of the matrix A are given by

$$a_{ij} = K(s_i, t_j) = \frac{1}{n} \exp\left(-\frac{(i-j)^2}{\varsigma^2 n^2} \right), \qquad i, j = 1, \ldots, n. \tag{7.4}$$

The bottom of Figure 7.1 shows a TSVD solution x_k computed with the truncation parameter $k = 180$. While a lot of the details of $f(t)$ are lost in $g(s)$, we are actually able to recover the intensity quite well, although with some oscillations due to the discontinuous nature of the function.

7.1.1 Discrete Convolution

When we look at the linear system $Ax = b$ obtained from discretization of the deconvolution problem, we note that if we define the quantities $h_{i-j} = a_{ij}$, then we can also write this system in the form

$$\boxed{\sum_{j=1}^{n} h_{i-j}\, x_j = b_i, \qquad i = 1, \ldots, n.} \tag{7.5}$$

Note that the quantities h_{i-j} are allowed to have zero and negative indices; they form a sequence from g_{-n+1} through h_0 to h_{n-1}. This is obviously a discrete analog of the continuous convolution in (7.3), and we refer to (7.5) as *discrete convolution*. Discrete convolution problems often arise from the discretization of convolution equations (7.3), but they may also arise directly in their discrete form, e.g., in digital signal processing.

When discrete convolution equations are written in the matrix form $Ax = b$, then we note that the matrix elements $a_{ij} = h_{i-j}$ are always a function of the difference $i - j$ only, such that $a_{ij} = a_{i+\ell,j+\ell}$ for all relevant integers i, j, and ℓ. Such a matrix A, which has constant values along all its diagonals, is called a *Toeplitz matrix*. Figure 7.2 shows an example of a symmetric Toeplitz matrix, which is completely specified by its first column. In MATLAB, the symmetric Toeplitz matrix A in (7.4) is computed as `toeplitz(-((0:n-1)/(sigma*n)).^2)`.

In some problems, the function h in (7.3) is periodic with period 1, that is,

$$h(t) = h(t + p), \qquad t \in \mathbb{R}, \quad p = \mathbb{N}.$$

It is easy to see that in the discrete case this corresponds to the condition that

$$h_{i-j} = h_{(i-j) \bmod n}, \qquad i, j \in \mathbb{N}.$$

$$\begin{pmatrix} 5 & 4 & 3 & 2 & 1 & 0 \\ 4 & 5 & 4 & 3 & 2 & 1 \\ 3 & 4 & 5 & 4 & 3 & 2 \\ 2 & 3 & 4 & 5 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 & 4 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix} \quad \begin{pmatrix} 5 & 0 & 1 & 2 & 3 & 4 \\ 4 & 5 & 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 0 & 1 & 2 \\ 2 & 3 & 4 & 5 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}$$

Figure 7.2. *Examples of structured matrices that arise in discrete deconvolution problems. From left to right: Symmetric Toeplitz, circulant, and Hankel.*

For these periodic convolution problems, we need only to specify the function $h(t)$ in the interval $t \in [0, 1]$, and we need only to specify the quantities $h_0, h_1, \ldots, h_{n-1}$. Moreover, it follows that the elements in the coefficient matrix A satisfy $a_{ij} = a_{(i+\ell) \bmod n, (j+\ell) \bmod n}$. A Toeplitz matrix, where the elements also "wrap around" the borders of the matrix, is called a *circulant matrix*; see Figure 7.2 for an example. A circulant matrix is completely specified by its first column.

Discrete deconvolution problems are thus special instances of discrete inverse problems with Toeplitz or circulant matrices, and they can be solved very efficiently by means of methods that take into account the structure of the matrix, such as FFT-based methods of complexity $O(n \log n)$. See, e.g., [35] for a discussion of these algorithms in the context of regularization algorithms.

7.1.2 Condition Number of a Gaussian Toeplitz Matrix

Let us consider the $n \times n$ symmetric Toeplitz matrix T_n whose first row and column have elements $h_0, h_1, \ldots, h_{n-1}$ given by

$$h_i = \exp\bigl(-(i/\varsigma)^2\bigr), h_{n/2-1}, \ldots, n-1. \tag{7.6}$$

This matrix resembles the one in (7.4) for the barcode problem; but note that the dimension n does not appear in the denominator in (7.6). We will show that the condition number of the matrix T_n is practically independent of n.

To make the analysis simpler, assume that n is even, and let C_n denote the circulant matrix whose first column has the elements

$$h_0, h_1, \ldots, h_{n/2}, 0, h_{n/2}, h_{n/2-1} \ldots, h_1.$$

Asymptotically the eigenvalues of T_n and C_n are identical [21], and they are quite close even for moderate values of n. This simplifies matters because the eigenvalues of C_n are given by the discrete Fourier transform of the first column of C_n.

Now consider the Toeplitz matrix T_{2n} whose elements are also given by (7.6), as well as the corresponding circulant matrix C_{2n}. Since the elements $h_0, h_1, h_2, \ldots$ decay quickly for the Gaussian kernel, the first columns of T_{2n} and C_{2n} can practically be considered as zero-padded versions of those of T_n and C_n. We know that if we pad a length-n signal with n additional zeros, then every even Fourier coefficient with

index $2i$ of the padded signal is identical to the ith Fourier coefficient of the original signal. Hence, the smallest eigenvalue of C_{2n} is approximately equal to the smallest eigenvalue of C_n, and consequently the same holds approximately for the smallest eigenvalues of T_{2n} and T_n.

Since the largest eigenvalue of C_{2n} is identical to that of C_n, we can conclude that the condition number—the ratio between the largest and smallest eigenvalues of a symmetric matrix—is approximately independent of the problem size n.

7.2 Inverse Crime—Ignoring Data/Model Mismatch

When we solve an inverse problem with real (and noisy) data, it is very important that there is consistency between the mathematical model—represented by the kernel K and the matrix A—and the data represented by the function $g(s)$ and the right-hand side b. This sounds so obvious that you may wonder why we mention it, yet it is occasionally ignored when developing, testing, and using regularization algorithms.

We illustrate the issue of model/data mismatch with an example using the gravity test problem from Section 2.1. We use the quadrature discretization of this problem, which was derived in Exercise 2.1 and implemented as test problem `gravity` in *Regularization Tools*. The underlying model assumes that the source, i.e., the density distribution $f(t)$ at depth d, is confined to the interval $[0, 1]$ along the t axis; outside this interval the function $f(t)$ is implicitly assumed to be zero, because the integration interval is $[0, 1]$.

In this example we use the call `gravity(64,3)` leading to a piecewise constant source function located at depth $d = 0.25$ and given by

$$f(t) = \begin{cases} 2, & 0 < t < \frac{1}{3}, \\ 1 & \frac{1}{3}, < t < 1. \end{cases}$$

The noise-free data $g(s)$ from this source is shown as the solid blue line in the top of Figure 7.3 for $-0.5 \leq s \leq 1.5$, and we see that the function $g(s)$ rapidly approaches zero for s outside the interval $[0, 1]$—a natural consequence of the fact that the source is confined to $t \in [0, 1]$. The blue circles show the values of the right-hand side b, which are samples of the function $g(s)$ in the interval $[0, 1]$. This noise-free problem is so well conditioned that we can safely compute the naive solution, which is shown as the solid line in the bottom of Figure 7.3.

As already mentioned, the model incorporates the implicit assumption about $f(t)$ being zero outside the interval $[0, 1]$. Hence there is perfect agreement between the model and the actual data in the right-hand side b, and thus we compute a perfect reconstruction (within rounding errors).

Such a perfect agreement between the model and the data in the discretized inverse problem arises when precisely the same model is used to generate the test data and to compute the reconstruction, and this situation is often referred to as *inverse crime* [11, p. 133]. Real data are obviously not generated by a computer model but come from a physical (or some other) phenomenon. Thus we must be very careful about the assumptions used when setting up the model!

To continue with the example, we modified the function `gravity` to generate data from a longer source which is nonzero in the interval $[-0.5, 1.5]$, obtained by

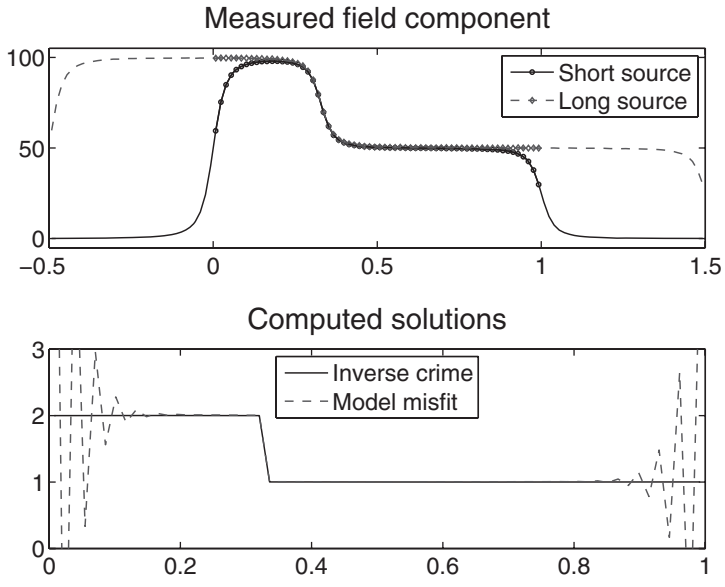

Figure 7.3. *Illustration of "inverse crime." When the data is generated by precisely the same model as that used in the inversion, then in the noise-free case we obtain perfect reconstruction (solid lines). This kind of test does not reveal the effects of a data/model mismatch (dashed lines).*

extending the two constant values of $f(t)$. This longer source produces a new data function $g(s)$, shown as the dashed line in the top of Figure 7.3, with a quite different behavior. In particular, the elements of the corresponding right-hand side, shown as the diamonds, are now almost constant for s close to 0 and 1. When we use this new right-hand side together with the *same matrix* A as before, then we compute the reconstruction shown as the dashed line in the bottom of Figure 7.3. Clearly, the computed reconstruction is severely "damaged" by oscillations, caused by the mismatch between the data and the assumption in the inversion model that $f(t) = 0$ for t outside $[0, 1]$.

The term "inverse crime" is used to signal the danger of testing a working regularization algorithm using only data generated with the same model as that used in the inversion. Of course, any good regularization algorithm must work under the conditions of inverse crime, and the first test of an algorithm should check this. However, a rigorous testing of a working regularization algorithm must also reveal how sensitive the algorithm is to model/data mismatch.

7.3 The Importance of Boundary Conditions

The incorporation of *boundary conditions* on the regularized solution is a convenient mechanism for avoiding severe effects of model/data mismatch, as illustrated in the previous section. Boundary conditions should be considered whenever the measured

data b come from a function $g(s)$ that can be influenced by values of the source function $f(t)$ outside the integration interval $[0, 1]$ used in the generic integral equation (2.2).

We emphasize that if the function $f(t)$ is known to be zero outside the integration interval, and if the data always come from a right-hand side function $g(s)$ obtained in this way, then there is obviously no need to worry about boundary conditions, and you may skip the rest of this section.

Here we treat only the important case of *reflexive boundary conditions*, where the underlying assumption is that the behavior of the function $f(t)$ outside $[0, 1]$ is a "reflection" of its behavior inside the interval. Specifically, we define the extended function f_{BC} in the interval $[-1, 2]$ as follows:

$$f_{\mathrm{BC}}(t) = \begin{cases} f(-t), & -1 < t < 0, \\ f(t), & 0 \le t \le 1, \\ f(2-t), & 1 < t < 2, \end{cases}$$

and we consider data $g_{\mathrm{BC}}(s)$ generated by the extended model

$$\begin{aligned} g_{\mathrm{BC}}(s) &= \int_{-1}^{2} K(s,t)\, f_{\mathrm{BC}}(t)\, dt \\ &= \int_{-1}^{0} K(s,t)\, f_{\mathrm{BC}}(t)\, dt + \int_{0}^{1} K(s,t)\, f_{\mathrm{BC}}(t)\, dt + \int_{1}^{2} K(s,t)\, f_{\mathrm{BC}}(t)\, dt \\ &= \int_{0}^{1} K(s,-t)\, f(t)\, dt + \int_{0}^{1} K(s,t)\, f(t)\, dt + \int_{0}^{1} K(s,2-t)\, f(t)\, dt \\ &= \int_{0}^{1} \big(K(s,-t) + K(s,t) + K(s,2-t) \big)\, f(t)\, dt. \end{aligned}$$

We see that reflexive boundary conditions simply correspond to working with a modified integral equation of the same generic form as (2.2), except that the kernel $K(s, t)$ is replaced by the extended kernel

$$K_{\mathrm{BC}}(s,t) = K(s,-t) + K(s,t) + K(s,2-t). \tag{7.7}$$

When we discretize this modified integral equation, then we obtain a modified coefficient matrix A_{BC} that is the sum of the original matrix A plus correction terms coming from discretization of the terms $K(s,-t)$ and $K(s,2-t)$ in $K_{\mathrm{BC}}(s,t)$.

We conclude that in order to incorporate boundary conditions into the reconstruction model, we must solve a new system of linear equations, obtained by *modifying* the matrix A in such a way that it takes into account our assumptions about the behavior of $f(t)$ outside the integration interval. The precise form of the modification depends on the assumed boundary conditions. If no boundary conditions are incorporated, then it follows from the integral equation formulation that we implicitly assume zero boundary conditions, i.e., that $f(t) = 0$ for t outside the interval $[0, 1]$.

To illustrate this technique, we return to the example from the previous section. This time, to be more realistic, we add noise e to the right-hand side, with the noise level $\|e\|_2/\|b\|_2 = 10^{-3}$. We compute regularized solutions with the original matrix

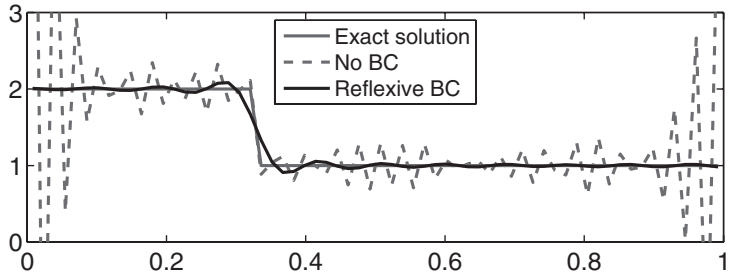

Figure 7.4. *Illustration of zero and reflexive boundary conditions (BC), using the test problem* `gravity`*. The regularized solution computed by a model that incorporates reflexive boundary conditions is much better than that with zero boundary conditions.*

A, corresponding to zero boundary conditions, and with the modified matrix A_{BC} that incorporates reflexive boundary conditions. Figure 7.4 shows the exact solution and the two regularized solutions. The incorporation of reflexive boundary conditions removes the oscillations due to model/data mismatch. There are still small oscillations in the solution due to the discontinuity in $f(t)$ at $t = \frac{1}{3}$.

In the next section we show how the modified matrix A_{BC} can be constructed easily in the case of deconvolution problems.

7.4 Taking Advantage of Matrix Structure

For the barcode problem in Section 7.1, where $f(t)$ represents black bars on a white background, it is advantageous to use the reflexive boundary conditions introduced in Section 7.3 in order to avoid boundary effects in the regularized solution. According to (7.7), the incorporation of the reflexive boundary conditions means that we must add correction terms A^{l} and A^{r} for the left and right boundary conditions, respectively. The elements of these two matrices are, for $i, j = 1, \ldots, n$,

$$a^{\text{l}}_{ij} = K(s_i, -t_j) = \exp\left(-\frac{(s_i + t_j)^2}{\varsigma^2}\right) = \exp\left(-\frac{(i + j - 1)^2}{\varsigma^2\, n^2}\right)$$

and

$$a^{\text{r}}_{ij} = K(s_i, 2 - t_j) = \exp\left(-\frac{(s_i + t_j - 2)^2}{\varsigma^2}\right) = \exp\left(-\frac{(i + j - 2n - 1)^2}{\varsigma^2\, n^2}\right).$$

For both matrices we see that element a_{ij} is a function of $i+j$, such that $a_{ij} = a_{i+\ell,j-\ell}$ for all relevant integers i, j, and ℓ. A matrix with this structure, where the elements are constant along all the antidiagonals, is called a *Hankel matrix*; see Figure 7.2 for an example. To completely specify a Hankel matrix, we must know its first column and its bottom row.

We shall now show that we can express the elements of the two Hankel matrices A^{l} and A^{r} in terms of the elements of the Toeplitz matrix A. For the first $n - 1$

elements of the first column of A^{l} we obtain

$$a^{\mathrm{l}}_{i1} = \exp\left(-\frac{((i+1)-1)^2}{\varsigma^2 n^2}\right) = a_{i+1,1}, \qquad j = 1, 2, \ldots, n-1,$$

while the last element a^{l}_{n1}, and, in fact, all the elements of the bottom row, are undetermined. Since the point spread function $h(t)$ can be considered as zero away from the peak, we can safely define these remaining elements of A^{l} to be zero. Similarly, the last $n-1$ elements of the bottom row of A^{r} are given by

$$a^{\mathrm{r}}_{nj} = \exp\left(-\frac{(1-(2+n-j))^2}{\varsigma^2 n^2}\right) = a_{1,2+n-j}, \qquad j = 2, 3, \ldots, n,$$

while we can define the remaining elements to be zero. Thus, there is no overlap between the nonzero elements of the two Hankel matrices A^{l} and A^{r}. Moreover, the matrix $A^{\mathrm{l}} + A^{\mathrm{r}}$ is also Hankel, and its first column is a shifted version of the first column of A, while its bottom row is a shifted version of the first row of A in reverse order. In MATLAB, `hankel([A(2:n,1);0],[0,A(1,n:-1:2)])`.

The resulting matrix $A_{\mathrm{BC}} = A + A^{\mathrm{l}} + A^{\mathrm{r}}$ is a symmetric Toeplitz-plus-Hankel matrix, with the Hankel part derived from the Toeplitz part. As an illustration,

$$A = \begin{pmatrix} 3 & 2 & 1 & 0 & 0 \\ 2 & 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 1 & 2 & 3 & 2 \\ 0 & 0 & 1 & 2 & 3 \end{pmatrix} \Rightarrow A^{\mathrm{l}} + A^{\mathrm{r}} = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}, \; A_{\mathrm{BC}} = \begin{pmatrix} 5 & 3 & 1 & 0 & 0 \\ 3 & 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 1 & 2 & 3 & 3 \\ 0 & 0 & 1 & 3 & 5 \end{pmatrix}.$$

We refer to a matrix with this particular structure as a *symmetric Toeplitz-plus-Hankel matrix* or STH matrix.

Now let $W_n = (w_1, \ldots, w_n)$ denote the $n \times n$ "DCT matrix" such that $\mathrm{dct}(x) = W_n^T x$ is the discrete cosine transform (DCT) of x (with MATLAB's Image Processing Toolbox, W_n = `dctmtx(n)'`). The columns w_i of W_n were introduced in (6.4)–(6.5). In Appendix B we prove that any STH matrix has the eigenvalue decomposition

$$A_{\mathrm{BC}} = W_n \, D \, W_n^T, \qquad D = \mathrm{diag}(d_1, \ldots, d_n),$$

where the eigenvalues d_i are given by

$$d_i = [\mathrm{dct}(A_{\mathrm{BC}}(:,1))]_i \,/\, [\mathrm{dct}(e_1)]_i, \qquad i = 1, \ldots, n,$$

where $e_1 = (1, 0, \ldots, 0)^T$ and $[\cdot]_i$ denotes the ith component. Note that the ordering if the eigenvalues in D is determined by fixed ordering of the columns of W_n.

Until this point we have always expressed our regularized solutions in terms of the SVD. The good news is that the SVD of A_{BC} is very closely related to its eigenvalue decomposition, since we have

$$A_{\mathrm{BC}} = (W_n \, \Omega) \, \mathrm{diag}(|d_1|, \ldots, |d_n|) \, W_n^T,$$

in which the diagonal matrix $\Omega = \mathrm{diag}(\mathrm{sign}(d_1), \ldots, \mathrm{sign}(d_n))$ holds the signs of the eigenvalues, such that $W_n \, \Omega = (\,\mathrm{sign}(d_1)\, w_1\,,\, \mathrm{sign}(d_2)\, w_2\,,\, \ldots\,)$. This is almost the

SVD of A_{BC}, except that the singular values in $\mathrm{diag}(|d_1|, \ldots, |d_n|)$ are not guaranteed to appear in nonincreasing order. However, since large singular values are typically associated with low-frequency singular vectors, the eigenvalues d_i tend to appear in order of decreasing magnitude.

Now it follows immediately that the Tikhonov solution to a discrete inverse problem with an STH matrix takes the form

$$x_\lambda = W_n \, \Phi^{[\lambda]} \, D^{-1} W_n^T \, b$$

and similarly for other regularized solutions, with the Tikhonov filter matrix $\Phi^{[\lambda]}$ replaced by the appropriate filter matrix. Since the multiplications with W_n and W_n^T represent the DCT and its inverse, it follows immediately that we can compute the Tikhonov regularized solution in $O(n \log n)$ complexity with the following MATLAB code, where `Abc` holds the matrix A_{BC}:

```
d = dct(Abc(:,1)) ./ dct(eye(n,1));
q = d ./ (d.^2 + lambda^2);
x_reg = idct( q .* dct(b) );
```

For TSVD, replace the second line with `q = [1./d(1:k);zeros(n-k,1)];` where `k` is the truncation parameter. Note that for large problems there is no need to store the full A and A_{BC} matrices, since the first column of each matrix suffices for the computations we need to carry out.

7.5 Deconvolution in 2D—Image Deblurring

Image deblurring is the science of computing a sharper reconstruction of a digital image from a blurred and noisy one, on the basis of a mathematical model of the blurring process. Figure 7.5 shows an example of image deblurring by means of Tikhonov regularization; the reconstruction was computed by means of the MATLAB software from [39]. The basic problem is very similar to the barcode reading problem from Section 7.1, where a clean signal (the sharp image) is deteriorated by blurring in the optical system. The main difference is that everything now involves two-dimensional signals, namely, the sharp and blurred images, as well as the point spread function.

In the continuous setting of Chapter 2, image deblurring is a first-kind Fredholm integral equation of the generic form

$$\int_0^1 \int_0^1 K(\mathbf{s}, \mathbf{t}) \, f(\mathbf{t}) \, dt_1 \, dt_2 = g(\mathbf{s}), \qquad \mathbf{s} \in [0,1] \times [0,1],$$

in which the two functions $f(\mathbf{t})$ and $g(\mathbf{s})$ that represent the sharp and blurred images are both functions of two spatial variables $\mathbf{s} = (s_1, s_2)$ and $\mathbf{t} = (t_1, t_2)$. Discretization of this problem, e.g., by means of the midpoint quadrature rule, then leads to the discrete image deblurring problem encountered in a variety of applications involving digital images.

For clarity we restrict this discussion to square gray-level images, but all our methods and results also hold for rectangular images as well as color images; see, e.g.,

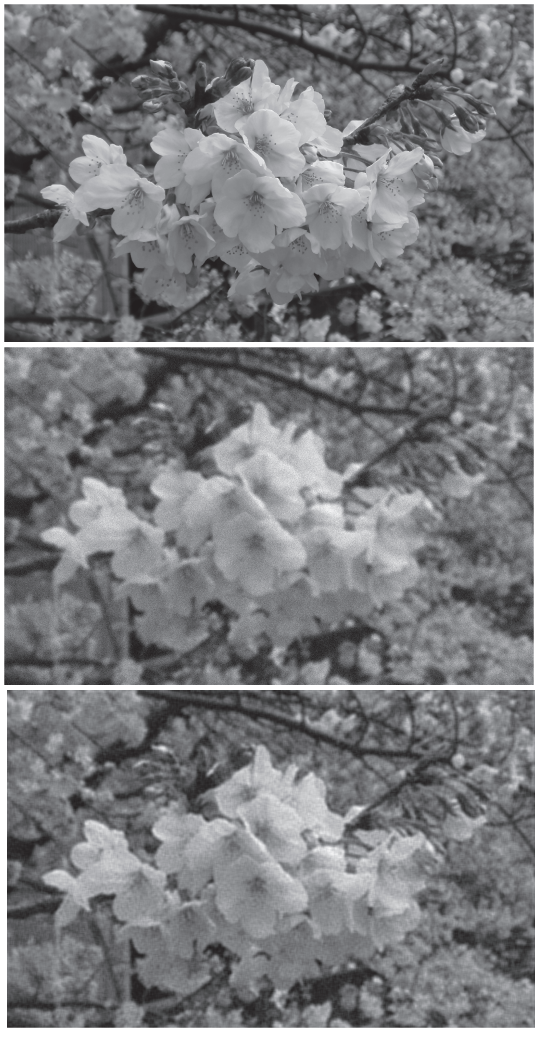

Figure 7.5. *Top to bottom: Sharp image, blurred and noisy image, and Tikhonov reconstruction.*

[39] for the necessary extensions. Let the two $N \times N$ arrays X and B represent the unknown, sharp image and the recorded blurred and noisy image, respectively, and let X_{ij} and B_{ij} denote the array elements, also known as the pixels in the two images. Also let the vectors x and b, of length $n = N^2$, consist of the "stacked" columns of X and B, i.e.,

$$\left.\begin{array}{rcl} x_\ell &=& X_{ij} \\ b_\ell &=& B_{ij} \end{array}\right\}, \qquad \text{where} \quad \ell = (j-1)N + i. \tag{7.8}$$

Figure 7.6. *The point spread function is the image of a single bright pixel.*

Then a good model for the blurring is very often of the form $Ax = b$, where A is an $N^2 \times N^2$ given matrix.[6]

In the rest of the discussion we will assume that the blurring is spatially invariant, i.e., the blurring is the same everywhere in the image. While this seems like a strong requirement, it is often a good approximation to the actual blurring, and it makes the deblurring problem much simpler than the general case.

7.5.1 The Role of the Point Spread Function

To arrive at the model $Ax = b$, let us first inspect the point spread function for image blurring, which prescribes how the light from a single pixel in the the sharp image X is spread over a number of pixels in the blurred image. Specifically, we define the array P—called the *point spread function* (PSF)—as the (possibly small) image of a single white pixel, as illustrated in Figure 7.6. We will use the convention that the elements of P sum to one, which means that the energy is preserved in the blurring process.

The point spread function may arise from a mathematical model of the blurring process, or from actual measurements of the point spread function. The dimensions of the computed or measured PSF array are often much smaller than N, due to the fact that the spreading of the light takes place in a small region around the white pixel. We can always conceptually augment such a small array with zeros to make the PSF array P as large as needed in the equations below.

From the assumption of spatial invariance and the definition of P it now follows that if $X_{ij} = 1$ for a certain index (i, j), and otherwise zero, then the blurred image B is given by

$$B_{kl} = P_{k-i,l-j}, \qquad i, j, k, l = 1, \ldots, N.$$

With this convention of indices, the elements of the PSF array P are allowed to have zero and negative indices, and the center of P is the element P_{00}.

Underlying the integral equation formulation is an assumption that the blurring is a linear process. This implies that given any image X, the corresponding blurred image B is the sum of the blurred images of all the pixels in X, as illustrated in Figure 7.7. Hence, the pixels in X and B are related through a *discrete* $2D$ *convolution* that takes

[6] In the literature we often find the notation $x = \mathrm{vec}(X)$ and $b = \mathrm{vec}(B)$ for (7.8).

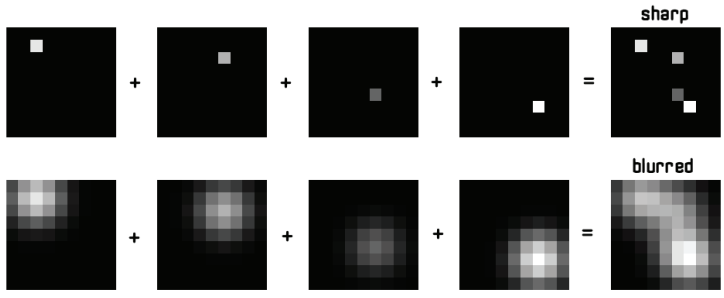

Figure 7.7. *The blurred image is the sum of all the blurred pixels.*

the form

$$\sum_{i,j=1}^{N} P_{k-i,l-j}\, X_{ij} = B_{kl}, \qquad k, l = 1, \ldots, N. \tag{7.9}$$

We immediately notice the similarities with the discrete one-dimensional convolution in (7.5). With this definition, where all indices to X and B are in the range $1, \ldots, N$, we are implicitly assuming zero boundary conditions; cf. the discussion in Section 7.3.

In MATLAB, the discrete 2D convolution in (7.9) is computed by means of the call `B = conv2(X,P,'same')`, where the array `P` that represents the point spread function is allowed to have smaller dimensions than the image X, and the input parameter `'same'` ensures that the output image `B` has the same dimensions as the input image `X`. The center of the PSF array, represented by the element P_{00}, is by definition located at the center of the $p \times q$ MATLAB array `P`.

Since the model (7.9) describes a linear relationship between the elements of the sharp and blurred images, it is clear that the linear relationship is preserved when the elements of X and B are rearranged into the vectors x and b according to (7.8). Hence, there is a unique—but complicated—relationship between the elements of the PSF P and the elements of the matrix A in the linear system $Ax = b$. This relationship is worked out in detail in [39], and in light of the discussion in Section 7.1, it should not come as a surprise that we will encounter Toeplitz structure. More precisely, the matrix $N^2 \times N^2$ A is block Toeplitz with Toeplitz blocks (BTTB), meaning that A consists of $N \times N$ blocks arranged in Toeplitz structure, and each block is an $N \times N$ Toeplitz matrix. The test problem `blur` in *Regularization Tools* creates such a matrix.

Due to the size of the coefficient matrix A, it is necessary to avoid forming it explicitly, and instead we must be able to perform all computations with A via (7.9) and knowledge about the PSF P. Typically, we are given two functions `Amult(P,x)` and `ATmult(P,x)` that compute the matrix-vector products Ax and $A^T x$ through the 2D convolution in (7.9) involving the point spread function stored in `P` and the image represented by `x`. For the square matrices considered here, the two functions could take the form

```
function y = Amult(P,x)
N = sqrt(length(x));
X = reshape(x,N,N);
Y = conv2(X,P,'same');
y = Y(:);
```

```
function y = ATmult(P,x)
N = sqrt(length(x));
X = reshape(x,N,N);
P = flipud(fliplr(P));
Y = conv2(X,P,'same');
y = Y(:);
```

In the function `ATmult`, the command `P = flipud(fliplr(P))` rotates the point spread function 180 degrees, and it can be shown that this corresponds to a multiplication with A^T. With these two functions at our hands, it is easy to use the CGLS algorithm for image deblurring, as shown in Exercise 7.5.

7.5.2 Rank-One PSF Arrays and Fast Algorithms

For certain point spread functions we obtain more structure in the matrix A, which immediately allows us to compute its SVD, similar to the technique developed in Section 7.4. Several such cases are discussed in [39]; here we restrict our discussion to the case where the matrix P is a rank-one matrix, i.e., it can be written as

$$P = a\,\bar{a}^T \qquad \rightarrow \qquad P_{ij} = a_i\,\bar{a}_j, \tag{7.10}$$

where a and $\bar{a}$ are two vectors whose indices, like those of P, can be zero and negative. We also assume that both a and $\bar{a}$ are symmetric about the "middle" element a_0 and $\bar{a}_0$, i.e., $a_i = a_{-i}$ and $\bar{a}_i = \bar{a}_{-i}$ for all relevant i. The resulting PSF P is doubly symmetric: it is symmetric across both the row and the column passing through its center. The following PSF arrays are doubly symmetric:

$$\begin{pmatrix} 1 \\ 2 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 4 \\ 5 \\ 4 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 4 & 8 & 12 & 8 & 4 \\ 5 & 10 & 15 & 10 & 5 \\ 4 & 8 & 12 & 8 & 4 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}.$$

If we revisit the 2D convolution in (7.9) and insert the expression (7.10) for the elements of P, then for $k, l = 1, \ldots, N$ we get

$$B_{kl} = \sum_{i,j=1}^{N} a_{k-i}\,\bar{a}_{l-j}\,X_{ij} = \begin{pmatrix} a_{k-1}, \ldots, a_{k-N} \end{pmatrix} X \begin{pmatrix} \bar{a}_{l-1} \\ \vdots \\ \bar{a}_{l-N} \end{pmatrix}.$$

It follows immediately that we can write the blurred image B as

$$B = A\,X\,\bar{A}^T,$$

where A and $\bar{A}$ are two symmetric Toeplitz matrices defined as

$$A = \begin{pmatrix} a_0 & a_{-1} & \cdots & a_{-(N-1)} \\ a_1 & a_0 & \ddots & a_{-(N-2)} \\ \vdots & \ddots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & \cdots & a_0 \end{pmatrix}, \quad \bar{A} = \begin{pmatrix} \bar{a}_0 & \bar{a}_{-1} & \cdots & \bar{a}_{-(N-1)} \\ \bar{a}_1 & \bar{a}_0 & \ddots & \bar{a}_{-(N-2)} \\ \vdots & \ddots & \ddots & \vdots \\ \bar{a}_{N-1} & \bar{a}_{N-2} & \cdots & \bar{a}_0 \end{pmatrix}.$$

As already mentioned, this model corresponds to the assumption of zero boundary conditions; if we want to use the reflexive boundary conditions introduced in Section 7.3, then we must replace A and $\bar{A}$ with their STH counterparts A_{BC} and $\bar{A}_{\text{BC}}$, as defined in the previous section.

To arrive at a fast regularization algorithm we proceed as before, using that A_{BC} and $\bar{A}_{\text{BC}}$ are both diagonalized by the DCT matrix W_n,

$$A_{\text{BC}} = W_n \, D \, W_n^T, \qquad \bar{A}_{\text{BC}} = W_n \, \bar{D} \, W_n^T,$$

leading to relations

$$B = W_n \, D \, W_n^T X \, W_n \, \bar{D} \, W_n^T \quad \Leftrightarrow \quad X = W_n \left(D^{-1} \left(W_n^T B \, W_n \right) \bar{D}^{-1} \right) W_n^T.$$

Here we recognize $\widehat{B} = W_n^T B \, W_n = \text{dct2}(B)$ as the 2D DCT of B, while $X = W_n \, \widehat{C} \, W_n^T = \text{idct2}(\widehat{C})$ is the inverse 2D DCT of $\widehat{C} = D^{-1} \widehat{B} \, \bar{D}^{-1}$. Thus, again following the ideas presented in the previous section, we arrive at the following efficient MATLAB code for computing a Tikhonov regularized solution (assuming that `Abc` and `Abc_bar` hold the matrices A_{BC} and $\bar{A}_{\text{BC}}$):

```
c = dct(eye(n,1));
d = dct(Abc(:,1)) ./ c;
d_bar = dct(Abc_bar(:,1)) ./ c);
Q = (d*d_bar') ./ ((d*d_bar').^2 + lambda^2);
X_reg = idct2( Q .* dct2(B) );
```

We recall that for large problems there is no need to store the full A_{BC} and $\bar{A}_{\text{BC}}$ matrices, since the first column of each matrix suffices for our computations. Extensions of this approach to other PSF arrays and other boundary conditions are described (including MATLAB algorithms) in [39].

7.6 Deconvolution and Resolution*

Given a problem with noisy data, and given a specific regularization method, we are interested in studying the *resolution*—how small are the details that we can reliably recover in the regularized solution? The analysis of resolution is closely connected to the number of SVD components that can be recovered from the noisy data, as well as the spectral properties of the singular vectors (which we discussed in Section 2.5). For many deconvolution problems this task is simplified by the fact that their singular vectors are samples of simple spectral basis functions. Specifically, for the deconvolution problems with reflexive boundary conditions that we consider in this chapter, the singular vectors are identical to the discrete cosine basis vectors associated with the DCT.

From the analysis in Chapters 4 and 5 we know that spectral properties of a regularized solution are determined by the singular vectors that take part in the solution. This is, of course, obvious for the TSVD method, but it holds for any method that involves a filtered expansion in the SVD basis. Hence, for a deconvolution problem, if we include the first k DCT basis vectors $w_1, \ldots, w_k$, then the size of the

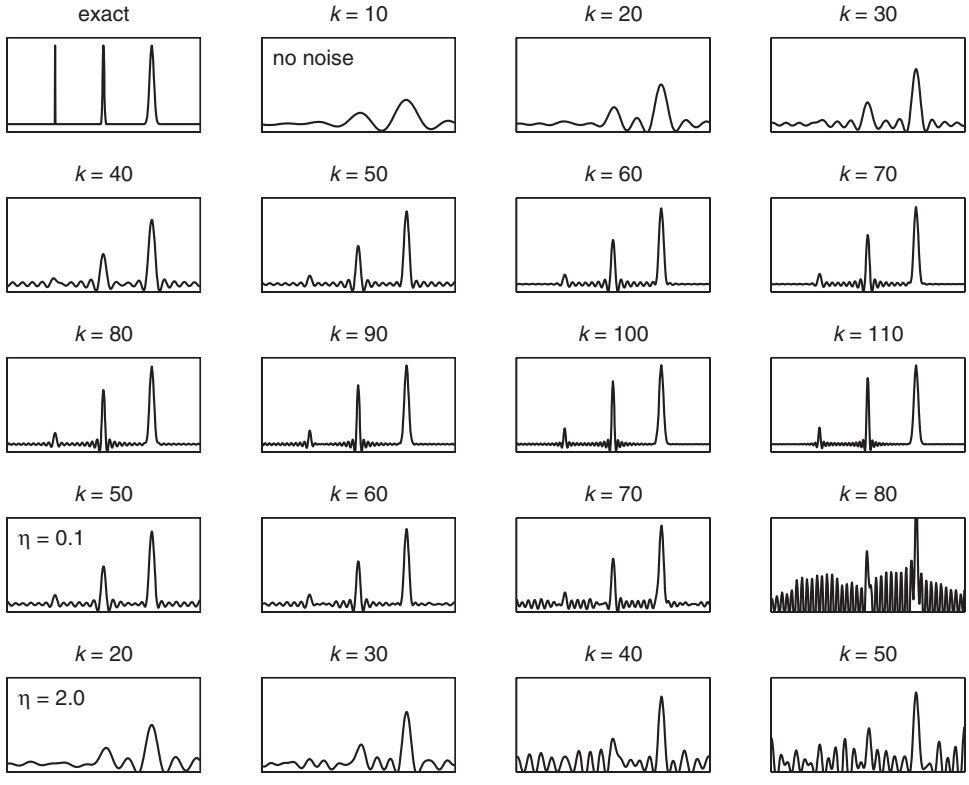

Figure 7.8. *Illustration of the resolution limit for a deconvolution problem with a DCT basis. Top left: The exact solution consisting of three narrow peaks of width* 0.004, 0.03, *and* 0.07, *respectively. Remaining plots in top three rows: TSVD solutions to a noise-free problem, where truncation errors in the form of oscillations are visible for large values of k. Bottom two rows: TSVD solutions to two noisy problems; for the smaller noise level* $\eta = 0.1$ *the two widest peaks can be reliably identified in the solutions, and for the larger noise level* $\eta = 2.0$ *only the widest peak is clearly visible.*

smallest details that we can hope to recover is determined by the width of a single oscillation in the vector w_k with the highest frequency. Assuming the standard interval $[0, 1]$ for the t variable, the width of a half a cosine in w_k is $1/(k-1)$, which we can define as the resolution limit.

Figure 7.8 illustrates the issues of resolution, using an exact solution consisting of three narrow peaks of approximate width 0.004, 0.03, and 0.07, respectively. The coefficient matrix A is a Toeplitz matrix that models a Gaussian kernel, and we impose reflexive boundary conditions leading to the DCT basis. The top part of the figure shows TSVD solutions for increasing values of the truncation parameter k for a noise-free problem, and we see that as k increases we obtain increasingly better

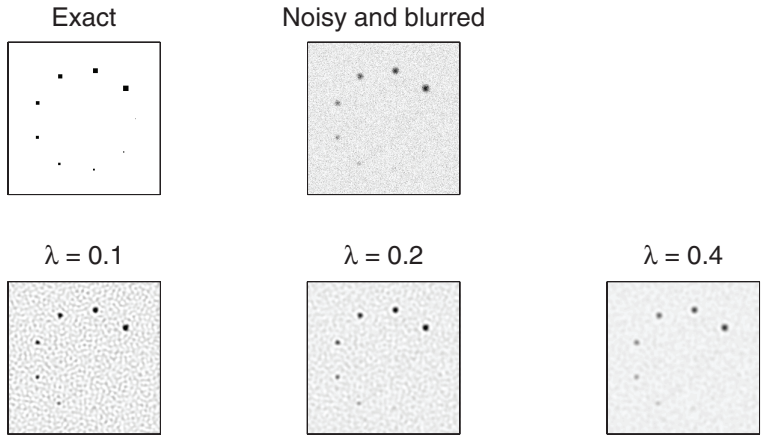

Figure 7.9. *Illustration of the resolution limit for image deblurring in the form of a 2D deconvolution problem with a 2D DCT basis. The color scale is inverted for clarity. Top: The exact image with nine spots of size* $1 \times 1, 2 \times 2, \ldots, 9 \times 9$ *pixels, and the noisy and blurred image. Bottom: Tikhonov solutions for three different values of the regularization parameter; only six (perhaps seven) spots can be identified in the deblurred images.*

approximations to the exact solution. Some oscillations, caused by the truncated expansion, are visible for all values of k.

The second to last row of plots in Figure 7.8 show TSVD solutions for a problem with some noise. The Picard plot (not shown) reveals that $k = 60$ is a good truncation parameter for this problem, leading to an expected resolution limit of about $1/60 \approx 0.017$, which indicates that we should be able to recover the two widest peaks only. The solution plots confirm this: in the regularized solutions we can identify the two widest peaks (although the amplitude of the second is incorrect), while there is no significant visible indication of the thinnest peak.

The four plots in the bottom row of Figure 7.8 show TSVD solutions for a problem with higher noise, where the Picard plot indicates that only about 40 SVD components can be included in the regularized solution. The resolution is therefore about $1/40 = 0.025$, meaning that we should barely be able to detect the middle peak. Again the solution plots confirm this: only the widest peak can be reliably identified (and its amplitude is incorrect for $k = 40$).

While the situation is conceptually the same for 2D problems, such as image deblurring, the analysis of resolution is complicated by the fact that spatial resolution in 2D is much more complex than in one dimension. It is outside the scope of this book to go into these issues, so we choose to merely illustrate matters with the example shown in Figure 7.9, which shows a similar numerical experiment using a test image with nine small spots of size $1 \times 1, 2 \times 2, \ldots, 9 \times 9$ pixels. Again we use a Gaussian blurring model with reflexive boundary conditions, and the Tikhonov solutions are computed by means of the function `tik_dct` from [39]. The inverted noise (that

appears as small high-frequency oscillations in Figure 7.8) now appears in the form of "freckles" that are visible in all three Tikhonov solutions. We can safely identify six (perhaps seven?) of the nine spots, while the smallest ones are lost in the inverted noise; we conclude that the resolution limit is about 4×4 pixels in this problem.

7.7 Tomography in 2D*

Tomography can be characterized as the science of computing reconstructions in 2D and 3D from projections, i.e., data obtained by integrations along rays (typically straight lines) that penetrate a domain Ω—typically a rectangle in 2D, and a box in 3D.

Here we consider a 2D model problem on the square domain $\Omega = [0,1] \times [0,1]$ (in arbitrary units), in which we are given an unknown function $f(\mathbf{t}) = f(t_1, t_2)$ that we wish to reconstruct. We assume that this function represents some material parameter, in such a way that the damping of a signal penetrating an infinitesimally small part $d\tau$ of a ray at position $\mathbf{t}$ is proportional to the product to $f(\mathbf{t})\, d\tau$. The data in the tomography problem consist of measurements of the damping of signals following well-defined rays through the domain Ω. See, e.g., [7, Section 7.4] for details of the mathematical formulation.

In this model problem, the ith observation b_i, for $i = 1, \ldots, m$, represents the damping of a signal that penetrates Ω along a straight line, which we refer to as ray^i; see Figure 7.10 for an example. All the points $\mathbf{t}^i$ on ray^i are given by

$$\mathbf{t}^i(\tau) = \mathbf{t}^{i,0} + \tau\, \mathbf{d}^i, \qquad \tau \in \mathbb{R},$$

where $\mathbf{t}^{i,0}$ is an arbitrary point on the ray, and $\mathbf{d}^i$ is a (unit) vector that points in the direction of the ray. Due to the above assumption, the damping is proportional to the integral of the function $f(\mathbf{t})$ along the ray. Specifically, for the ith observation, and ignoring a problem-specific constant, the damping associated with the ith ray is given by

$$b_i = \int_{-\infty}^{\infty} f\big(\mathbf{t}^i(\tau)\big)\, d\tau, \qquad i = 1, \ldots, m,$$

where $d\tau$ denotes the integration along the ray.

We can discretize this problem by dividing Ω into an $N \times N$ array of pixels, and in each pixel (k, ℓ) we assume that the function $f(\mathbf{t})$ is a constant $f_{k\ell}$:

$$f(\mathbf{t}) = f_{k\ell} \qquad \text{for} \qquad t_1 \in I_k \text{ and } t_2 \in I_\ell,$$

where we have defined the interval $I_k = [\,(k-1)/N\,,\; k/N\,]$, $k = 1, \ldots, N$ (and similarly for I_ℓ). With this assumption about $f(\mathbf{t})$ being piecewise constant, the expression for the kth measurement takes the simpler form

$$b_i = \sum_{(k,\ell) \in \text{ray}^i} f_{k\ell}\, \Delta L_{k\ell}^{(i)}, \qquad \Delta L_{k\ell}^{(i)} = \text{length of ray}_i \text{ in pixel } (k, \ell)$$

for $i = 1, \ldots, m$. Again, see Figure 7.10 for clarification, for the case $N = 6$.

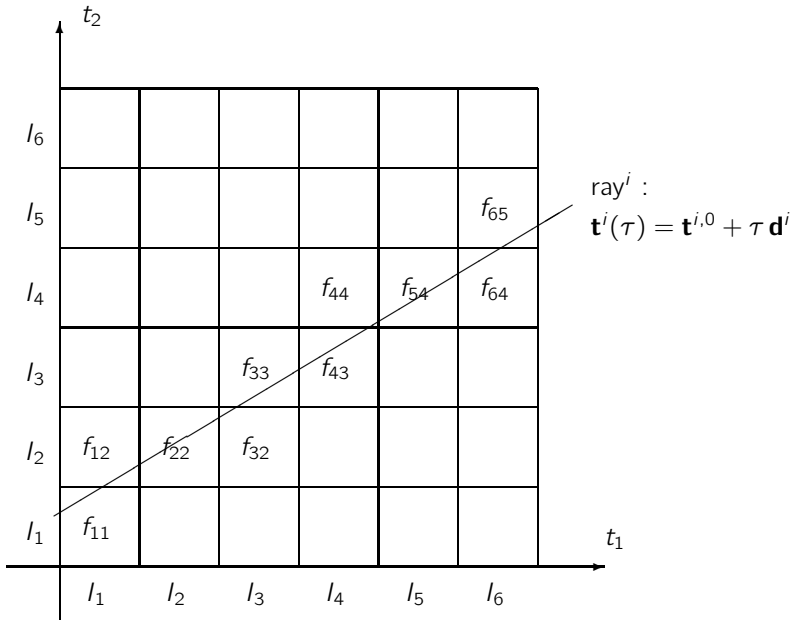

Figure 7.10. *Example of a discretized tomography problem with* $N = 6$*. The* i*th ray intersects a total of* 10 *pixels, and thus the* i*th row of the matrix* A *has* 10 *nonzero elements (in columns* 1, 2, 8, 14, 15, 21, 22, 28, 34, *and* 35*).*

The above equation is, in fact, a linear system of equations in the N^2 unknowns $f_{k\ell}$. To arrive at a more convenient notation for this system, we introduce the vector x of length $n = N^2$ whose elements are the (unknown) function values $f_{k\ell}$, ordered as follows:

$$x_j = f_{k\ell}, \qquad j = (\ell - 1)\,N + k.$$

This corresponds to stacking the columns of the $N \times N$ matrix F whose elements are the values $f_{k\ell}$. Moreover, we organize the measurements b_i into a vector b.

There is clearly a linear relationship between the data b_k and the unknowns in the vector x, meaning that we can always write

$$b_i = \sum_{j=1}^{n} a_{ij}\,x_j, \qquad i = 1, \ldots, m.$$

With the above definitions it then follows that we arrive at a linear system of equations $A\,x = b$ with an $m \times n$ matrix whose elements are given by

$$a_{ij} = \begin{cases} \Delta L_{k\ell}^{(i)}, & (k,\ell) \in \text{ray}_i \\ 0 & \text{else.} \end{cases}$$

We recall that index i denotes the ith observation (corresponding to ray$_i$) and j denotes the pixel number in an ordering with $j = (\ell - 1)\,N + k$. The matrix A is very sparse,

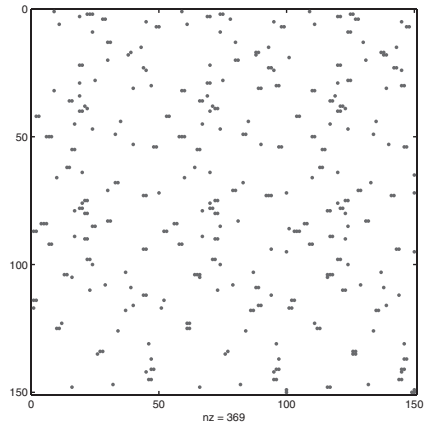

Figure 7.11. *An arbitrary* 150×150 *submatrix of a sparse tomography matrix* A *for the case* $N = 50$ *leading to a matrix of dimensions* 2500×2500.

and the number of nonzero elements in any row is bounded above by $2N$. Figure 7.11 shows a typical example of a 150×150 submatrix of A for the case $N = 50$.

In the image deblurring problem described in the previous section, one would never form the coefficient matrix explicitly. On the other hand, for tomography problems it is often feasible and convenient to explicitly form the very sparse matrix A. For both problems one will use an iterative method, and the classical methods from Section 6.1 have proved to be quite efficient for tomography problems (while they may be very slow when applied to general discrete inverse problems with a dense coefficient matrix). Exercise 7.8 illustrates this.

7.8 Depth Profiling and Depth Resolution*

We have already seen a variety of examples of how inverse problems can be used to reveal hidden information, that would somehow involve opening or destroying the object. Therefore, inverse problems play an important role in nondestructive testing, a common analysis technique used in science and industry to evaluate the properties of a material, component, or system without causing damage to it.

The example in this section, which acts as a prelude to the geophysical prospecting problem considered in the next section, deals with reconstruction of hidden layers of paint in an image via solving an inverse problem that involves X-ray measurements. The technique is called Particle-Induced X-ray Emission Spectroscopy (PIXE) depth profiling; see, e.g., [55] for details.

At a specific position, an X-ray source sends a beam at an angle s into the material, and the energy penetrates to a depth of $d\cos(s)$, where d depends on the energy in the beam. The beam that penetrates the material is partially reflected according to the material parameter $f(t)$, where t denotes the depth into the material, with $0 \leq t \leq d$. A detector located perpendicular to the surface records the reflected signal $g(s)$, which depends on the incidence angle s and the material parameter $f(t)$

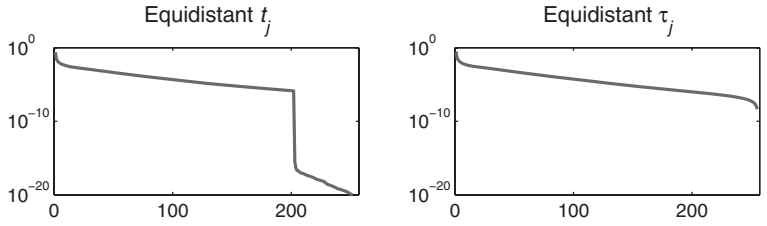

Figure 7.12. *Left: The singular values σ_i of the matrix corresponding to a discretization of the depth profiling problem with equidistant abscissas t_j. Right: The singular values for a discretization based on the variable transformation $t_j = \sin(\tau_j)$ with equidistant τ_j. The variable transformation removes the artificial singular values at or below machine precision.*

via the following simplified model:

$$g(s) = \int_0^{d\cos(s)} \exp(-\mu\, t)\, f(t)\, dt, \qquad 0 \le s \le \tfrac{1}{2}\pi.$$

Thus, reconstruction of $f(t)$ is an inverse problem given by the first-kind Fredholm integral equation (2.2) with kernel

$$K(s,t) = \begin{cases} \exp(-\mu\, t), & 0 \le t \le d\, \cos(s), \\ 0\ , & d\, \cos(s) \le t \le d, \end{cases} \tag{7.11}$$

with s and t in the intervals $s \in [0, \frac{1}{2}\pi]$ and $t \in [0, d]$. In this example we set $d = 1$ and $\mu = 8$, and we use an artificial solution given by

$$f(t) = \exp\bigl(-30(t-0.25)^2\bigr) + 0.4\exp\bigl(-30(t-0.5)^2\bigr) + 0.5\exp\bigl(-50(t-0.75)^2\bigr),$$

which models three layers of paint at depths $t = 0.25$, 0.5, and 0.75.

If we discretize this problem by means of the midpoint quadrature rule (3.1) with equidistant abscissas t_j, and with equidistant sampling points s_i for $g(s)$, then we obtain a coefficient matrix with elements

$$a_{ij} = \begin{cases} n^{-1}\exp(-\mu\, t_j), & t_j \le \cos(s_i), \\ 0 & \text{else.} \end{cases}$$

The singular values of this matrix are shown in the left part of Figure 7.12 for an example with $n = 256$. We see that due to approximation errors and finite-precision arithmetic, this matrix has a large cluster of singular values at or below machine precision. This is unfortunate, because the integral operator has only the trivial null space.

We can improve the numerical model by using a different quadrature rule with nonequidistant points, which we obtain by means of the variable transformation $t = \sin(\tau)$, leading to the transformed model

$$g(s) = \int_0^{\arcsin(d\cos(s))} \exp\bigl(-\mu\, \sin(\tau)\bigr)\, \cos(\tau)\, f(\tau)\, d\tau, \qquad 0 \le s \le \tfrac{1}{2}\pi.$$

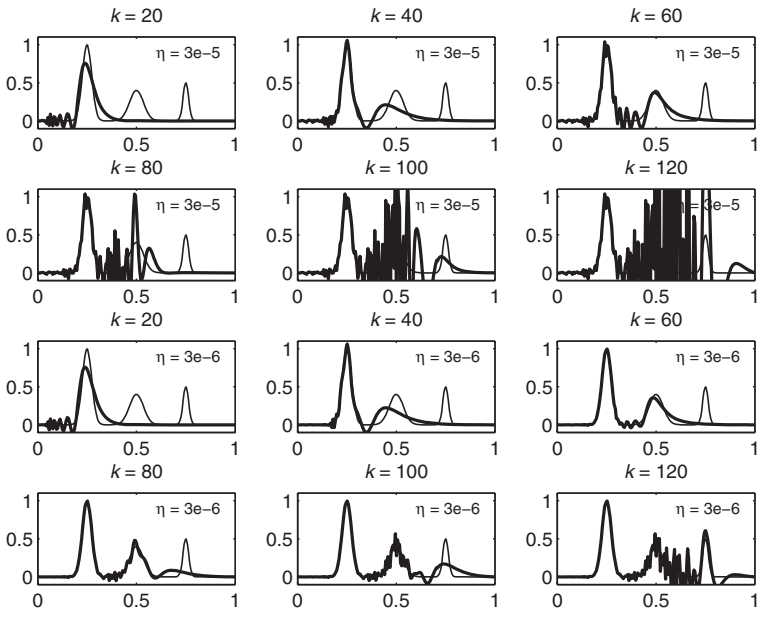

Figure 7.13. *Selected TSVD solutions (thick lines) to the depth profiling problem, for two different noise levels, together with the exact solution (thin lines). For the larger noise level $\eta = 3 \cdot 10^{-5}$ the noise starts to dominate the reconstruction for $k \approx 80$, and consequently we can recover only the two top layers of paint. For the smaller noise level $\eta = 3 \cdot 10^{-6}$ we are able to recover all three layers with some confidence.*

Using again the midpoint quadrature rule, this time with equidistant $\tau_j \in [0, \frac{1}{2}\pi]$ (and again with $d = 1$), we obtain a new coefficient matrix with elements

$$a_{ij} = \begin{cases} (\pi/2n) \exp\bigl(-\mu \sin(\tau_j)\bigr) \cos(\tau_j), & \sin(\tau_j) \le s_i, \\ 0 & \text{else.} \end{cases} \tag{7.12}$$

The singular values of this matrix, shown in the right part of Figure 7.12, show a much more satisfactory behavior free from rounding error artifacts.

Figure 7.13 shows TSVD reconstructions for the latter discretization model, for increasing values of k, and using two different noise levels $\eta = 3 \cdot 10^{-5}$ and $\eta = 3 \cdot 10^{-6}$. We see that as k increases we obtain information about deeper structures, until the noise starts to dominate the reconstruction, and with a lower noise level (where the noise starts to dominate for a larger value of k) the deep part of the reconstruction is more reliable. Only with the small noise level do we have a reliable reconstruction of the bottom layer of paint.

This illustrates the important question of *depth resolution:* To which depth can we, for a given problem and with a given noise level, reliably resolve details in the solution? It is no surprise that the smaller the noise level, the deeper we can "look into" the material in this problem. This a variant of the classical resolution problem

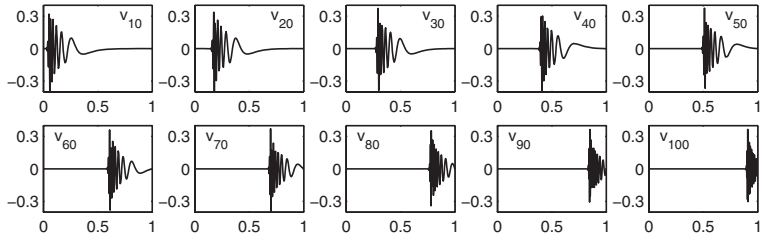

Figure 7.14. *Selected right singular vectors plotted versus the depth t. Each singular vector carries information about the solution in a certain depth interval only, starting from the smallest depths.*

discussed in Section 7.6. The next section gives another example of depth resolution in geophysics.

The key to analyzing and understanding depth resolution often lies in a study of the right singular vectors v_i that provide the basis for the regularized solution. Some of these singular vectors are shown in Figure 7.14 for the depth profiling problem; for clarity we plot the elements of the singular vectors versus the abscissas $t_j = \sin(\tau_j)$ to better illustrate the interplay between the singular vectors and the depth. Clearly, each singular vector carries information about the solution in a certain depth interval only, starting from the smallest depths for small indices i. When we include the first k SVD components in a regularized solution, then we can recover details in those depth ranges for which the singular vectors $v_1, \ldots, v_k$ carry information. The combined inspection of the right singular vectors and the Picard plot (which reveals the SVD components that can be recovered from noisy data) is a convenient way to obtain the desired insight about depth resolution.

In Section 8.3 we return to this basis and demonstrate that there is actually a better basis for solving this problem than the SVD basis that underlies the analysis here.

7.9 Digging Deeper—2D Gravity Surveying*

Throughout this book we have used the gravity surveying problem from Section 2.1 as a simple test problem. Real-world gravity surveying problems are, of course, much more complex than this simple one-dimensional problem; in particular, one needs a 3D reconstruction of the mass density distribution below the ground. Such large-scale problems are, however, beyond the scope of this book. As a compromise, in this section we study the inverse potential field problem associated with a 2D gravity surveying problem, which still gives insight into more realistic problems as well as providing another example of the concept of depth resolution.

To set the stage, let $f(t, z)$ denote the mass density distribution as a function of the horizontal and vertical coordinates t and z. We consider the domain $\Omega = [0, 1] \times [0, d]$, which represents a vertical "slice" below the ground, from the surface

at $z = 0$ to the depth $z = d$. We ignore all mass outside[7] this slice. Following the derivation from Section 2.1 it is easy to see that the vertical component of the gravity field due to $f(t, z)$ at the surface is given by

$$g(s) = \int_0^1 \int_0^d \frac{z}{(z^2 + (s - t)^2)^{3/2}} \, dz \, dx, \qquad 0 \le s \le 1. \tag{7.13}$$

Typically the depth d of the domain Ω is much smaller than the width of the domain; here we use $d = 0.1$.

We discretize the domain Ω into a grid of $N_t \times N_z$ pixels, and in the center of each pixel we conceptually place a point source whose magnitude we wish to find from samples of the measured field $g(s)$ at m equidistant points on the surface. This corresponds to a very simple Galerkin discretization with delta functions as basis functions, leading to a system with $n = N_t \cdot N_z$ unknowns (the source magnitudes) organized into a vector $x \in \mathbb{R}^n$ (similar to the organization of the unknowns in the image deblurring and tomography problems above). The coefficient matrix $A \in \mathbb{R}^{m \times n}$ can easily be constructed from submatrices obtained from calls to the function `gravity` in *Regularization Tools*. We also construct an exact solution which consists of two smoothly varying mass concentrations located at $(t, z) = (\frac{1}{3}, \frac{1}{3}d)$ and $(t, z) = (\frac{11}{16}, \frac{2}{3}d)$; see Figure 7.16 for an example with $N_t = 40$ and $N_z = 20$.

Figure 7.15 shows the Picard plots for the exact and the noise-corrupted right-hand side, using $m = n = 800$. We use the noise level $\eta = 10^{-4}$, which is clearly visible as the plateaux where the right-hand side coefficients $|u_i^T b|$ level off for the noisy problem. The solution coefficients do not (overall) increase, showing that the discrete Picard condition is satisfied. From these Picard plots we expect that we can recover about 400 SVD components for the noise-free data (the remaining coefficients are contaminated with rounding errors) and about 240 SVD components for the noisy data. We also see that it makes sense to use TSVD truncation parameters k that are multiples of $N_t = 40$, the number of discretization points in the t direction.

Figures 7.16 and 7.17 show TSVD solutions for the noise-free and noisy problems, respectively. Both figures demonstrate that as more SVD components are included in the regularized solutions, we obtain better reconstructions of the larger source in the left part of the domain Ω. For small values of k the reconstructed source is clearly too shallow, and the more SVD components we include the more correct the localization of this source; about 400 components are needed to compute a reasonably good reconstruction. We also see that the smaller and deeper source in the right part of the domain cannot be reconstructed well—even in the noise-free problem—due to the influence of the rounding errors. For the noisy data, the maximum truncation parameter before the noise sets in is $k = 280$, and the corresponding TSVD solution is not able to produce the correct sources: the bottom part of the larger source is not reconstructed correctly, and the smaller and deeper source is both too shallow and too weak.

When too few SVD components are included in the reconstruction, then the source is artificially placed too close to the surface. This resembles the behavior

[7] An alternative model, sometimes referred to as "$2\frac{1}{2}$-dimensional," assumes that the function $f(t, z)$ extends infinitely in the third dimension; this model has a different kernel than the one considered here.

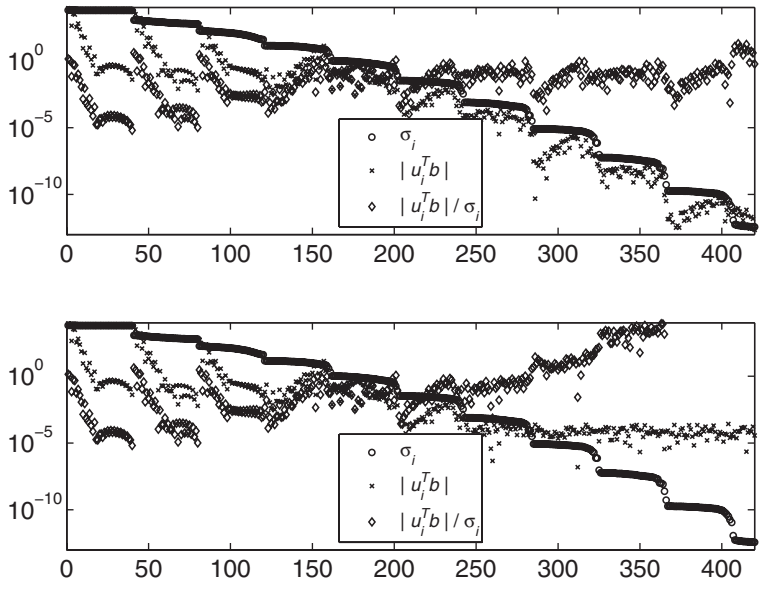

Figure 7.15. *Picard plots for the noise-free problem (top) and the problem with noisy data (below). The singular values beyond* $i = 410$ *level off at the plateaux around* $3 \cdot 10^{-13}$.

of the formulation ambiguity mentioned in Section 2.4; but it is an artifact of the discretization and the oversmoothed regularization.

Clearly, the depth resolution depends on the number of SVD components included in the regularized solution. To see how this is reflected in the right singular vectors v_i, it is convenient to lump information about $N_t = 40$ singular vectors at a time by computing the vectors $w_1, w_2, \ldots$ whose elements are the RMS values of the elements of 40 consecutive singular vectors. For example, the first two vectors have elements given in terms of $v_1, \ldots, v_{40}$ and $v_{41}, \ldots, v_{80}$ by

$$(w_1)_i = \left(\sum_{j=1}^{N_t} v_{ij}^2 \right)^{\frac{1}{2}}, \qquad (w_2)_i = \left(\sum_{j=N_t+1}^{2N_t} v_{ij}^2 \right)^{\frac{1}{2}}, \qquad i = 1, \ldots, n.$$

Figure 7.18 shows the vectors w_l in the form of $N_z \times N_t$ images similar to the SVD solutions, and we see that the large elements in each vector w_l are confined to a few layers in the solution around the lth layer. This implies that each sequence of 40 right singular vectors mainly carries information about the solution at the lth layer (this resembles the localization of the singular vectors in Figure 7.14 for the depth profiling problem). The singular vectors v_i beyond $i = 400$ are heavily influenced by rounding errors, and so are the corresponding vectors w_l.

In conclusion, we see again how a combined inspection of the Picard plot and the right singular vectors—this time in the form of the vectors w_l in Figure 7.18—

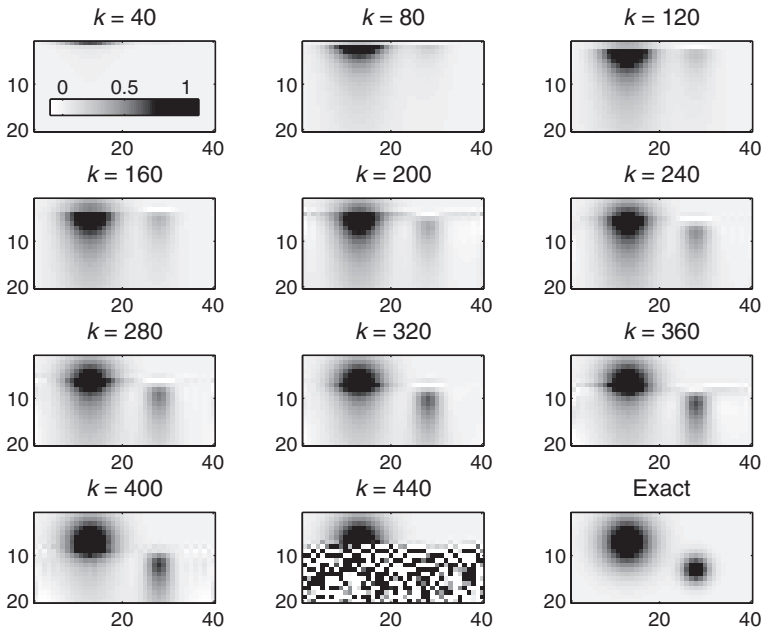

Figure 7.16. *TSVD solutions for the noise-free* 2D *gravity surveying problem, and the exact solution. For* $k = 440$ *the solution is heavily influenced by rounding errors.*

provides valuable insight about the depth resolution available in a given noisy problem. Extensions of these ideas to 3D potential field problems are introduced and discussed in [15].

7.10 Working Regularization Algorithms

With all the insight and algorithms available at this point, the time is right to discuss the necessary ingredients of a regularization algorithm for solving a given class of problems. To make this more concrete, we define a *Working Regularization Algorithm* (WRA) as a consistent combination of three ingredients: a regularization method, a parameter-choice method, and an implementation. As we have demonstrated in this book, the analysis of a given inverse problem is always very important—almost as important as computing the regularized solution. Therefore, it is relevant to consider if the WRA, in addition to computing the solution, should also provide means for computing quantities that provide insight into the problem (such as a Picard plot, an L-curve, or a resolution study). Figure 7.19 summarizes these issues.

For a WRA to be of practical value to solve real problems, it must satisfy some minimum requirements. First, for efficiency the implementation should take into account any available mathematical structure in the problem (e.g., symmetry, sparsity, or Toeplitz/Hankel matrix structure). Also, the algorithm must work for a class of

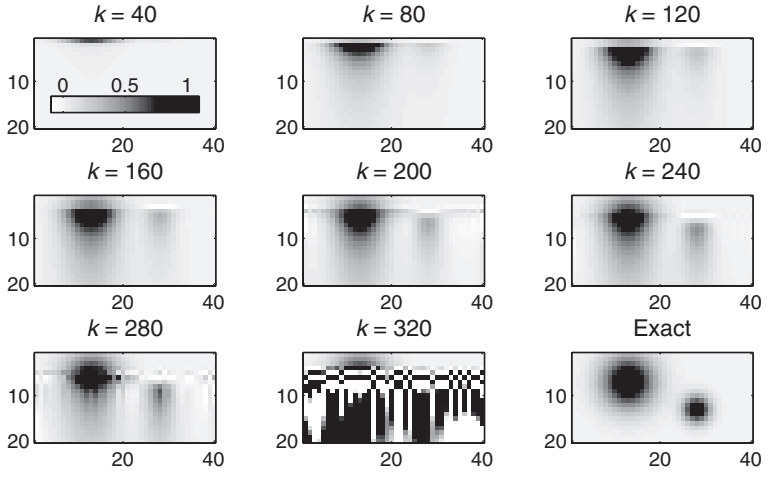

Figure 7.17. *TSVD solutions for the noisy problem, and the exact solution.*

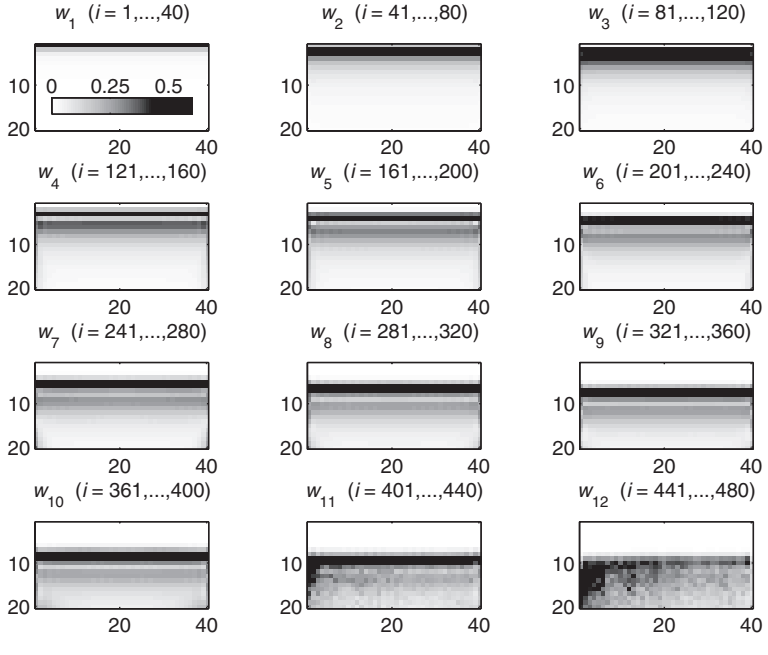

Figure 7.18. *The vectors w_l whose elements are RMS values of the elements of the* 40 *consecutive singular vectors v_i listed for each plot. Each sequence of* 40 *right singular vectors mainly carries information about the solution at the lth layer (until rounding errors start to dominate).*

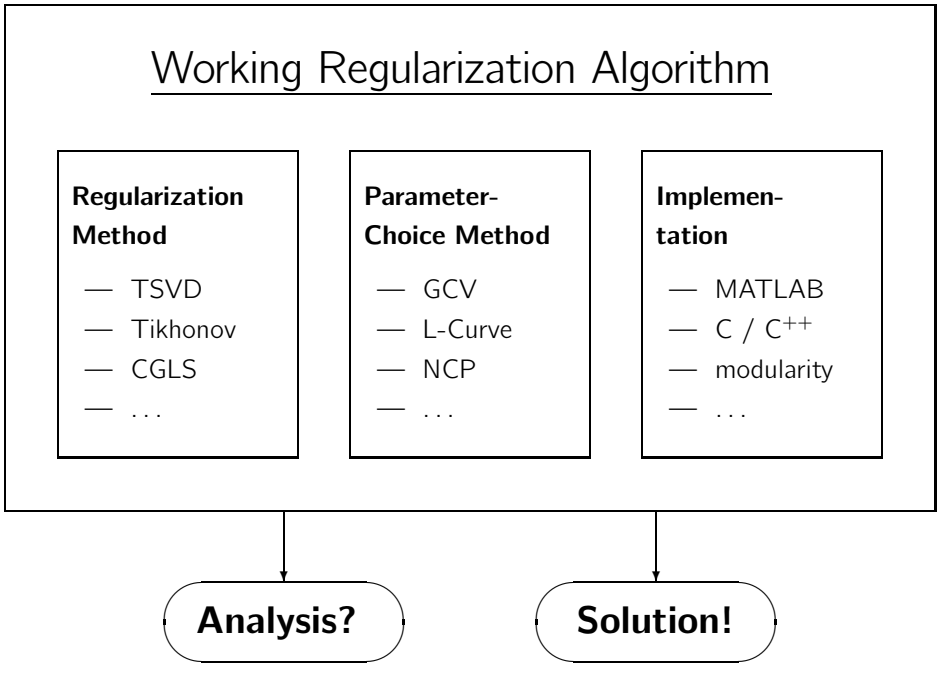

Figure 7.19. *Illustration of the main ingredients of a Working Regularization Algorithm (WRA) for solving discrete inverse problems. See the text for more details.*

representative test problems, generated such that they reflect the properties of the inverse problem to be solved. Finally, it must work for Gaussian white noise of realistic noise levels; if it cannot handle such noise, then there is no hope that it will work for real data with nonideal noise.

The WRA should incorporate everything the user knows about the problem, the data, and the errors. Obtain all available information from the experts who carry out the measurement; data collection experts usually know a lot about their instruments and the noise that they are facing. Also, make sure that the forward model is as accurate as possible, and make precise what is wanted from the solution; should it be symmetric, what boundary conditions are needed, etc. Further requirements may be relevant, and in the next (final) chapter we touch upon how to incorporate some of this a priori information. Do not expect mathematics to compensate for any lack of knowledge of the above; mathematics is just a language for expressing this knowledge in a form useful for algorithm development. Moreover, do not expect that a single parameter-choice method will always work.

The implementation of the WRA as a piece of software is beyond the scope of this book; but you should consider which programming language is appropriate, and you should check if software already exists for core computations such as matrix-vector products in the iterative methods. When designing, implementing, and testing the WRA, we suggest using the following checklist.

1. Focus on a single application, or a specific and narrow class of applications; no WRA is guaranteed to work for a broad class of problems.

2. When implementing the WRA, focus on modularity and clarity of the computer code; it is guaranteed that you need to go back and modify/expand the software at some point in time.

3. Make sure you understand the performance of the implementation, including computing times, storage requirements, etc.

4. When testing the WRA, make sure to generate test problems that reflect as many aspects as possible of real, measured data.

5. When testing, also make sure to model the noise as realistically as possible, and use realistic noise levels.

6. Be aware of the concept of inverse crime:

   (a) As a "proof-of-concept" first use tests that commit inverse crime; if the WRA does not work under such circumstances, it can never work.

   (b) Next, in order to check the robustness to model errors, test the WRA without committing inverse crime.

7. Carefully evaluate the regularized solutions; consider which characteristics are important, and use the appropriate measure of the error (the 2-norm between the exact and regularized solutions is not always the optimal measure).

8. Using the same exact data, create many realizations of the noise and perform a systematic study the robustness of the WRA. Use histograms or other tools to investigate if the distribution of the errors has an undesired tail.

7.11 The Story So Far

In this chapter we introduced several new important inverse problems. These problems are interesting in their own right, and they are also vehicles for introducing new concepts and techniques associated with these problems. The barcode reading problem is an inverse problem from daily life, and it naturally leads to a discussion of deconvolution problems and structured matrices (Toeplitz, Hankel, and circulant), and how we can solve such structured problems efficiently by means of fast transforms (the focus here is on DCT, the discrete cosine transform).

We also used the barcode reading problem to discuss the importance of inverse crime and boundary conditions. Inverse crime is what we commit when we test a regularization method with precisely the same computational model for the forward computations and the regularization method. Preferably, the forward computation should be done with a method that resembles the physics of the problem as well as possible; for example, it should correctly model the behavior of the data measurement near the edges of the domain.

Of course, if a regularization method does not work well under the conditions of inverse crime, then there is probably no hope that it will ever work on real data. So it is important to first test a method under inverse crime. However, one should not rely on a regularization method that has only been tested by committing inverse crime!

The discussion of the influence of boundary conditions clearly illustrates the above issues. At the same time, we saw how we can treat boundary conditions (in our case, reflexive boundary conditions) without sacrificing the computational efficiency of the regularization methods.

We then moved on to discuss 2D image reconstruction, in the form of image deblurring and tomography. In both cases, we showed how these large-scale problems can conveniently be solved by means of iterative regularization methods. We demonstrated how certain image deblurring problems with a structured point spread function and reflective boundary conditions can be treated efficiently by means of the 2D DCT. We also introduced the 2D gravity surveying problem as another example of an imaging problem.

Throughout the chapter we also discussed some issues related to resolution, i.e., the size of details that can be identified in a regularized solutions, and we showed how the resolution depends on the properties of the right singular vectors that take part in the regularized solution. This is true for both spatial resolution in image deblurring and depth resolution in gravity surveying. In particular, for DCT-based problems we can directly relate the resolution limit to the wavelength of the discrete cosine with the highest frequency included in the regularized solution.

Finally, we introduced the framework of a Working Regularization Algorithm (WRA) and summarized the most important aspects of developing a robust regularization algorithm.

Exercises

7.1. **The Point Spread Function and the Toeplitz Matrix**
This exercise relates the one-dimensional point spread function from the barcode reading problem in Section 7.1 to the columns and rows of the Toeplitz matrix. Assume that $\ldots, h_{-2}, h_{-1}, h_0, h_1, h_2, \ldots$ are the elements of the discrete point spread function, with h_0 corresponding to the center. Show that the image of the jth unit vector e_j is the vector

$$(h_{1-j}, \ldots, h_0, \ldots, h_{n-j})^T, \qquad j = 1, \ldots, n. \tag{7.14}$$

Then show that the blurred image of the vector $x = (x_1, x_2, \ldots, x_n)^T$ is given by

$$\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \sum_{j=1}^{n} x_j \begin{pmatrix} h_{1-j} \\ \vdots \\ h_{n-j} \end{pmatrix}$$

and that the jth column of A is the vector in (7.14). Finally, use these results to show that the ith row in A is given by $(h_{i-1}, h_{i-2}, , \ldots, h_{i-n})$, with the

elements of the point spread function appearing in reverse order, and show that this immediately leads to (7.5). This explains why it is natural to use a notation for discrete convolution where the elements of the point spread function appear in reverse order.

7.2. **More Light on Barcode Reading**
Revisit the barcode reading problem with a kernel that is a "top hat" ⎍ function (arising from an out-of-focus model for the blurring in the reader). The corresponding coefficient matric can be generated as

```
A = toeplitz([ones(p,1);zeros(n-p,1)]).
```

Use the SVD to verify that this matrix is rank-deficient, and solve a noisy realization of the problem by means of the TSVD method.

7.3. **Another Deconvolution Problem**
The mathematical model considered here has the form of a deconvolution problem (7.3) with the kernel $K(s,t) = h(s-t)$ given by the function

$$h(t) = \frac{\sin(c\,t)}{\pi\,t},$$

where c is a positive constant and $s, t \in [-1, 1]$. This model arises in signal processing in connection with extrapolation of a band-limited signal, in coherent imaging, and in many other applications. As an example, we will use $c = 15$ and the exact solution

$$f(t) = \exp\left(-\left(\frac{t-0.5}{0.15}\right)^2\right) + 0.85\,\exp\left(-\left(\frac{t+0.7}{0.1}\right)^2\right).$$

This particular problem is constructed as a good test problem for this exercise.

Use a simple method to discretize the problem such that you obtain a coefficient matrix A with Toeplitz structure. You can assume zero boundary conditions. The Toeplitz matrix that you obtain is called the prolate matrix, and it has been studied carefully in the literature; see, e.g., [73], where it is shown that its eigenvalues cluster near zero and one.

Use the same discretization method to obtain the exact solution x^{exact} corresponding to the above $f(t)$, and construct the exact right-hand side as $b^{\text{exact}} = A\,x^{\text{exact}}$. Then write a MATLAB program that implements your discretization of the test problem (let c be an input parameter).

Now choose the dimension $n = 128$ and the relative noise level `rnl` $= 10^{-6}$, and compute regularized solutions by means of the following four methods: TSVD (4.2), Tikhonov (4.8), Landweber with nonnegativity constraints (6.3), and CGLS. Comment on the computed results.

Try to use several parameter-choice methods (such as the L-curve criterion, the GCV method, and the NCP criterion) to compute the regularization parameter. To use the NCP criterion for the iterative methods, you must use the implementation from Exercise 6.6. Again, comment on the results.

7.4. **The NCP Criterion and the DCT Basis**

The NCP criterion as described in Section 5.5 is based in the discrete Fourier transform (DFT), which is probably the most well-known transform method for performing a spectral analysis. However, other transforms will also work; for example, we can use the discrete cosine transform (DCT), which requires us only to change the definition of q in (5.14) to $q = m - 1$. This is a natural choice for deconvolution problems with reflexive boundary conditions.

Implement the NCP criterion in the DCT setting, and test it on a deconvolution problem of your own choice; make sure to use reflexive boundary conditions.

7.5. **Image Deblurring by the CGLS Algorithm**

Your first task is to create a new version `new_cgls` of the function `cgls` from *Regularization Tools*, where the multiplications with A and A^T are replaced by calls to the two functions `Amult` and `ATmult` from Section 7.5. For convenience, in `new_cgls` you should replace the first input parameter `A` with a variable `P` that holds the point spread function, such that the calls to the multiplication routines take the form `Amult(P,...)` and `ATmult(P,...)`. In addition, in line 38 of `cgls`, you must change the statement `[m,n] = size(A)` to `n = length(b)`.

Next you should create a sharp image X^{exact} of dimensions $N \times N$ (do not make N too large), as well as the vector x^{exact} with the columnwise representation of the image. If you choose your own image, make sure that it is a square gray-scale image stored in `double` format. Alternatively, you can create a 480×480 test image by means of the commands

```
load mandrill
clear caption map
N = 480;
X_exact = double(X(:,10+(1:N)));
x_exact = X_exact(:);
```

Then create a 31×31 array `P` with the point spread function by means of the commands

```
[I,J] = meshgrid(0.01*(-15:15));
P = (1 + (I/0.15).^2 + (J/0.1).^2).^(-1);
P = P/sum(P(:));
```

Finally, create a blurred noisy image B, and the corresponding vector representation $b = A x^{\text{exact}} + e$, by means of the commands

```
b_exact = Amult(P,x_exact);
e = randn(size(b_exact)); e = e/norm(e);
e = 0.01*norm(b_exact)*e;
b = b_exact + e;
B = reshape(b,N,N);
```

Then you can use the call `Z = new_cgls(P,b,k)` to compute a matrix `Z` of dimensions $N^2 \times$ `k` in which each column is an iteration vector. You should use

`reshape(Z(:,j),N,N)` to convert the `j`th column of `Z` to the corresponding image.

Now run a total of 150 CGLS iterations (perhaps more, if your image dimension is larger than $N = 480$), and inspect some of the reconstructions. You should observe a gradual improvement of the reconstructions, until the inverted noise starts to appear in the image. When should you stop the iterations, if you want to minimize the error $\|x^{\text{exact}} - x^{(k)}\|_2$? Compare this optimal choice of iterations with the choice obtained by means of the discrepancy principle.

7.6. **Image Deblurring with the 2D DCT**

The 2D DCT deblurring algorithm described in Section 7.5.2 works only for rank-one PSF arrays P. One common PSF that satisfies this requirement is the Gaussian PSF,

$$P_{ij} = \exp\left(-\frac{1}{2}\left(\frac{i}{\varsigma_1}\right)^2 - \frac{1}{2}\left(\frac{j}{\varsigma_2}\right)^2\right),$$

where the parameters ς_1 and ς_2 define the width of the PSF. The center of the PSF is located at $i = j = 0$. The corresponding MATLAB code takes the form, for a 31×31 PSF array,

```
[I,J] = meshgrid(0.01*(-15:15));
P = exp(-0.5*(I/s1).^2 -0.5*(J/s2).^2);
P = P/sum(P(:));
```

To generate a test problem with realistic behavior at the edges, we must blur a larger sharp image and then crop the blurred image to the desired dimensions to simulate how the image is actually recorded in a camera. The following MATLAB code demonstrates how to do this, by cropping a border of width 15 pixels:

```
load mandrill
clear caption map
N = 480;
X = double(X(:,10+(1:N)));
B = conv2(X,P,'same');
B_exact = B(16:N-15,16:N-15);
X_exact = X(16:N-15,16:N-15);
N = 450;
```

Use this approach, with $\varsigma_1 = 1$ and $\varsigma_2 = 1.5$, to generate a blurred noisy image, and then reconstruct it with the 2D DCT algorithm described in Section 7.5 (for reflexive boundary conditions).

7.7. **Image Deblurring by the Richardson–Lucy Algorithm***

The Richardson–Lucy algorithm is an iterative method that is often used in image deblurring problems for computing nonnegative reconstructions. For example, it was used for deblurring of the flawed images produced by the Hubble Space Telescope during its first three years of operation (due to the telescope mirror being slightly out of shape).

Underlying the Richardson–Lucy algorithm is the statistical problem of maximizing a likelihood function given by

$$\mathcal{L}(x) = \|Ax\|_2 - b^T \log(Ax),$$

where the log function is taken elementwise. Here it is assumed that both Ax and b are positive. The Richardson–Lucy algorithm is a particular fixed-point iteration scheme for solving the problem $\nabla \mathcal{L}(x) = 0$. Let d denote a vector whose elements are the column sums in A, i.e., $d_j = \sum_{i=1}^{m} a_{ij}$ for $j = 1, \ldots, n$. Then each iteration takes the form

$$x^{[k+1]} = x^{[k]} \odot A^T \big(b \oslash (A x^{[k]}) \big) \oslash d, \qquad k = 0, 1, 2, \ldots,$$

in which $\odot$ and $\oslash$ denote elementwise multiplication and division, respectively. The initial vector $x^{[0]}$ must be positive. In many presentations it is assumed that the PSF is already scaled such that d is the vector of all 1's and the elementwise scaling is left out.

The Richardson–Lucy iteration is very simple and, similar to the CGLS algorithm, it can take advantage of efficient black-box functions that compute the multiplications with A and A^T. As noted in [1] the method is identical to a weighted steepest descent method with positivity constraints, and hence it always produces a reconstruction with positive elements.

Implement the Richardson–Lucy algorithm as stated above, using the two functions `Amult` and `ATmult` from Section 7.5. Then apply it to the test problem described in the previous exercise, and comment on the convergence rate and the results. If you implemented the function `new_cgls` in the previous exercise, then you should also compare the performance of the two iterative methods (note that the cost of one iteration in both methods is dominated by the two matrix multiplications).

7.8. **Tomography Reconstructions***

Use the function `tomo` from *Regularization Tools* to construct a 2D tomography problem of size $N \times N$, according to the model from Section 7.7. This function uses random numbers to generate the test problem, so to ensure that you can reproduce your results you should initialize `rand` in the same way before each time you call `tomo`. Use `X = reshape(x,N,N)` to get from the "stacked" representation of the image `x` to the image itself.

For the image dimension we suggest you start with the value $N = 50$ (larger values are more realistic, but the computing times for the ART algorithm may become overwhelming in MATLAB). The problem has $n = N^2$ unknowns (the pixel values), and you should use the same number of measurements (i.e., use `f` $= 1$ in `tomo`). Add Gaussian white noise with $\eta = 0.5$.

Run 50 CGLS and ART iterations, using the implementations from *Regularization Tools*. (If you implemented Landweber's method in Exercise 7.3, you can also use this method here.) Plot the errors $\|x^{\text{exact}} - x^{[x]}\|_2$ and $\|x^{\text{exact}} - x^{(x)}\|_2$ versus k, and comment on these error histories. What is the optimal number of iterations for each of the two methods?

Also plot the prediction errors $\|b^{\text{exact}} - A\,x^{[k]}\|_2$ and $\|b^{\text{exact}} - A\,x^{(k)}\|_2$ versus k. What is the optimal number of iterations for the two methods if the goal is to minimize the prediction error?

Finally, we turn to possible methods for choosing k. For both methods plot the solution norm versus the residual norm. Why is it not possible to use the L-curve criterion for the tomography problem? Then try to use the discrepancy principle for both iterative methods. Is this parameter-choice method able to produce a good reconstruction?

7.9. **The Impact of the Boundary Condition***

The purpose of this exercise is to illustrate the influence of the boundary conditions on the regularized solutions, through a study of the TSVD method for a deconvolution problem. Recall that the Toeplitz matrix A in the discretized problem actually corresponds to an assumption about zero boundary conditions, and also recall that we can write the corresponding TSVD solutions as

$$x_k = A_k^\dagger b \qquad \text{with} \qquad A_k^\dagger = \sum_{i=1}^{k} v_i\,\sigma_i^{-1}\,u_i^T$$

(cf. Section 4.2). The STH matrix A_{BC} for reflexive boundary conditions is obtained by adding a Hankel matrix to A, as described in Section 7.4. If the SVD of this matrix is $A_{\text{BC}} = \widetilde{U}\,\widetilde{\Sigma}\,\widetilde{V}^T$, then the TSVD solutions with reflexive boundary conditions are given by

$$\tilde{x}_k = \widetilde{A}_k^\dagger b \qquad \text{with} \qquad \widetilde{A}_k^\dagger = \sum_{i=1}^{k} \tilde{v}_i\,\tilde{\sigma}_i^{-1}\,\tilde{u}_i^T.$$

Thus, the difference between the two solutions obtained with the two different assumptions about boundary conditions is given by

$$x_k - \tilde{x}_k = (A_k^\dagger - \widetilde{A}_k^\dagger)\,b,$$

and we can therefore study the influence of the boundary conditions on the regularized solution by studying the matrix difference

$$\Delta_k = A_k^\dagger - \widetilde{A}_k^\dagger.$$

In particular, large elements in this matrix indicate a possibility that elements in the corresponding positions of b lead to a significant difference between x_k and $\tilde{x}_k$.

Use the gravity problem or the barcode problem (both lead to deconvolution problems) to investigate the above: generate a discrete problem, compute the SVDs of A and A_{BC}, and create the matrix Δ_k for a range of truncation parameters k. You should see that the largest elements of Δ_k are always located in its top left and bottom right corners, showing that the influence of changing the boundary conditions has the largest effect near the ends of the regularized solution, and this influence is small only if the right-hand side b has small elements near its ends.

You should also see that the matrix Δ_k has nonzero (although somewhat smaller) elements in its middle rows and columns, showing that the influence of the change of boundary conditions is not restricted to effects at the ends of the regularized solution.

Chapter 8

Beyond the 2-Norm: The Use of Discrete Smoothing Norms

This final chapter covers a few slightly more advanced topics that are very important when solving real problems, and a basic understanding of these topics is a must for anyone who wants to solve inverse problems. The demands on the reader are higher than in the previous chapters, however, and therefore the chapter may not be suited for a basic course in discrete inverse problems.

The main topic is the use of smoothing norms other than the 2-norm $\|x\|_2$ for obtaining regularity of the solution. The smoothing norms we cover here involve a derivative of the function to be reconstructed, and the main focus is on discrete smoothing norms that take the form $\|L\,x\|_2$, where the matrix L is a discrete approximation to a derivative operator. We finish with a brief introduction to total variation regularization, which uses the smoothing norm $\|L\,x\|_1$; this is the only time you will encounter the 1-norm in this book.

Throughout the chapter we assume that the solution lives on an equidistant grid with grid points $t_j = (j - \frac{1}{2})/n$, $j = 1, \ldots, n$, as samples in a quadrature method. This simplifies the discussion of the discrete derivative matrix L. Various issues in choosing the matrix L are discussed in [12], [57], and Section 4.3 in [32]. The choice of smoothing norms can also be set in a statistical setting [8].

8.1 Tikhonov Regularization in General Form

The presentation so far has focused on the 2-norm as a means for controlling (or suppressing) the error in the regularized solution. The norm $\|x\|_2$ enters explicitly in the Tikhonov formulation (4.8), and it is also part of the alternative formulation (4.5) of the TSVD method.

While the norm $\|x\|_2$ is indeed useful in a number of applications, it is not always the best choice. In order to introduce a more general formulation, let us return to the continuous formulation of the first-kind Fredholm integral equation from Chapter 2. In this setting, the residual norm for the generic problem in (2.2) takes the form

$$R(f) = \left\| \int_0^1 K(s,t)\, f(t)\, dt - g(s) \right\|_2 .$$

In the same setting, we can introduce a *smoothing norm* $S(f)$ that measures the regularity of the solution f. Common choices of $S(f)$ belong to the family given by

$$S(f) = \|f^{(d)}\|_2 = \left(\int_0^1 \left(f^{(d)}(t) \right)^2 dt \right)^{1/2}, \qquad d = 0, 1, 2, \ldots,$$

where $f^{(d)}$ denotes the dth derivative of f. Then we can write the Tikhonov regularization problem for f in the form

$$\min \left\{ R(f)^2 + \lambda^2 S(f)^2 \right\}, \tag{8.1}$$

where λ plays the same role as in the discrete setting. Clearly, the Tikhonov formulation in (4.8) is merely a discrete version of this general Tikhonov problem with $S(f) = \|f\|_2$.

Returning to the discrete setting, we can also formulate a general version of the Tikhonov problem by replacing the norm $\|x\|_2$ with a discretization of the smoothing norm $S(f)$. The general form of the discrete smoothing norm takes the form $\|L\,x\|_2$, where L is a discrete approximation of a derivative operator, and the associated Tikhonov regularization problem in *general form* thus takes the form

$$\boxed{\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2 \right\} \ .} \tag{8.2}$$

The matrix L is $p \times n$ with no restrictions on the dimension p.

If L is invertible, such that L^{-1} exists, then the solution to (8.2) can be written as $x_{L,\lambda} = L^{-1}\bar{x}_\lambda$, where $\bar{x}_\lambda$ solves the standard-form Tikhonov problem

$$\min_{\bar{x}} \{ \|(A\,L^{-1})\,\bar{x} - b\|_2^2 + \lambda^2 \|\bar{x}\|_2^2 \}.$$

The multiplication with L^{-1} in the back-transformation $x_\lambda = L^{-1}\bar{x}_\lambda$ represents integration, which yields additional smoothness in the Tikhonov solution, compared to the choice $L = I$. The same is also true for more general rectangular and noninvertible smoothing matrices L.

Similar to the standard-form problem obtained for $L = I$, the general-form Tikhonov solution $x_{L,\lambda}$ is computed by recognizing that (8.2) is a linear least squares problem of the form

$$\min_x \left\| \begin{pmatrix} A \\ \lambda L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2. \tag{8.3}$$

The general-form Tikhonov solution $x_{L,\lambda}$ is unique when the coefficient matrix in (8.3) has full rank, i.e., when the null spaces of A and L intersect trivially:

$$\mathcal{N}(A) \cap \mathcal{N}(L) = \emptyset.$$

Since multiplication with A represents a smoothing operation, it is unlikely that a smooth null vector of L (if L is rank-deficient) is also a null vector of A.

Various choices of the matrix L are discussed in the next section; two common choices of L are the rectangular matrices

$$L_1 = \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times n}, \tag{8.4}$$

$$L_2 = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix} \in \mathbb{R}^{(n-2)\times n}, \tag{8.5}$$

which represent the first and second derivative operators, respectively (in *Regularization Tools* use `get_l(n,1)` and `get_l(n,2)` to compute these matrices). Thus, the discrete smoothing norm $\|L\,x\|_2$, with L given by I, L_1, or L_2, represents the continuous smoothing norms $S(f) = \|f\|_2$, $\|f'\|_2$, and $\|f''\|_2$, respectively. We recall the underlying assumption that the solution f is represented on a regular grid, and that a scaling factor (related to the grid size) is absorbed in the parameter λ.

To illustrate the improved performance of the general-form formulation, consider a simple ill-posed problem with missing data. Specifically, let x be given as samples of a function, and let the right-hand side be given by a subset of these samples, e.g.,

$$b = A\,x, \qquad A = \begin{pmatrix} I_{\text{left}} & 0 & 0 \\ 0 & 0 & I_{\text{right}} \end{pmatrix},$$

where I_{left} and I_{right} are two identity matrices. Figure 8.1 shows the solution x (consisting of samples of the sine function), as well as three reconstructions obtained with the three discrete smoothing norms $\|x\|_2$, $\|L_1\,x\|_2$, and $\|L_2\,x\|_2$. For this problem, the solution is independent of λ. Clearly, the first choice is bad: the missing data are set to zero in order to minimize the 2-norm of the solution. The choice $\|L_1\,x\|_2$ produces a linear interpolation in the interval with missing data, while the choice $\|L_2\,x\|_2$ produces a quadratic interpolation here.

In certain applications it can be advantageous to use a smoothing norm $S(f)$ that incorporates a combination of several derivatives of the function f, namely, the *weighted Sobolev norm* defined by

$$S(f) = \left(\alpha_0\|f\|_2^2 + \alpha_1\|f'\|_2^2 + \cdots + \alpha_p\|f^{(d)}\|_2^2\right)^{1/2}$$

(the standard Sobolev norm corresponds to $\alpha_0 = \cdots = \alpha_d = 1$). Such smoothing norms are particularly handy in 2D and 3D problems where we need to enforce different regularity on the solution in different directions, e.g., by choosing $S(f) = \left(\|\partial f/\partial x\|_2^2 + \|\partial^2 f/\partial y^2\|_2^2\right)^{1/2}$ in a 2D problem. There is a vast number of choices of combinations, and the best choice is always problem dependent.

In the discrete setting, weighted Sobolev norms are treated by "stacking" the required matrices that represent the derivative operators. For example, if we use the two matrices in (8.4)–(8.5), then the matrix L that corresponds to the Sobolev norm

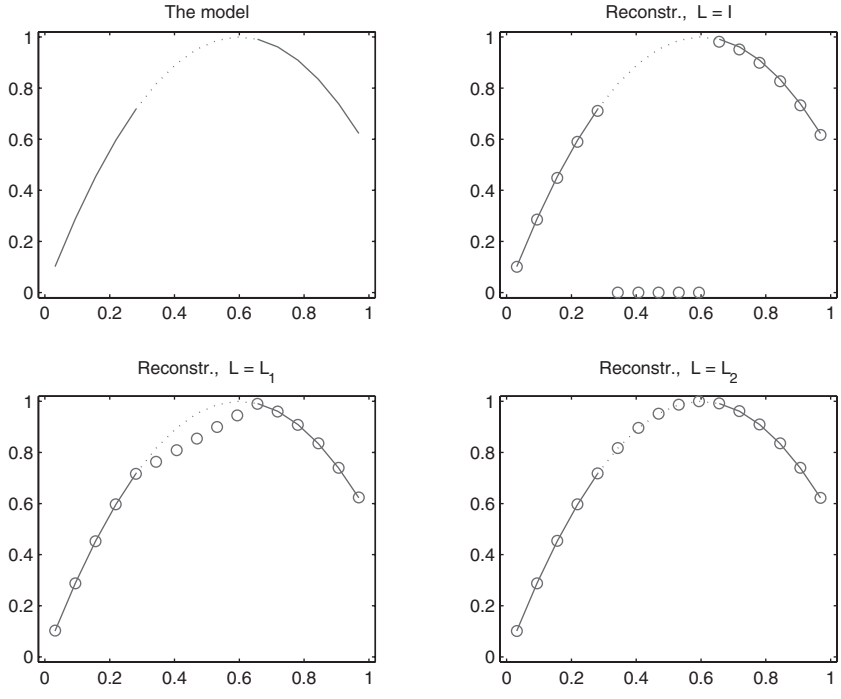

Figure 8.1. *Illustration of the missing data problem. We show Tikhonov solutions for three choices of the discrete smoothing norm* $\|L\,x\|_2$.

with $d = 2$ takes the form

$$L = \begin{pmatrix} I \\ L_1 \\ L_2 \end{pmatrix} \in \mathbb{R}^{(3n-3)\times n}.$$

If we compute the QR factorization of this "tall" matrix, i.e., $L = Q\,R$, then we can replace L with the square, banded, and upper triangular matrix R. To see that this is valid, just notice that

$$\|L\,x\|_2^2 = (L\,x)^T(L\,x) = (Q\,R\,x)^T(Q\,R\,x) = x^T R^T Q^T Q\,R\,x = x^T R^T R\,x = \|R\,x\|_2^2$$

(since the matrix Q has orthonormal columns). Similar techniques may apply to problems in two and three dimensions. For large-scale problems, such factorizations of L may be useful, provided that the triangular factor R remains sparse.

We finish this brief discussion of general-form regularization methods with a discussion of the null space of L. It is often convenient to allow the matrix L to have a nontrivial null space

$$\mathcal{N}(L) \equiv \{x \in \mathbb{R}^n : L\,x = 0\}.$$

For example, the null spaces of the two matrices L_1 and L_2 defined in (8.4) and (8.5) are given by

$$\mathcal{N}(L_1) = \text{span}\{\,(1,1,\ldots,1)^T\,\}, \tag{8.6}$$

$$\mathcal{N}(L_2) = \text{span}\{\,(1,1,\ldots,1)^T\,,\,(1,2,\ldots,n)^T\,\}. \tag{8.7}$$

These null spaces are natural, because they reflect the null spaces of the corresponding smoothing norms $\|f'\|_2$ and $\|f''\|_2$. Specifically, f' is zero when f is any constant function, and f'' is zero if f is any constant or linear function.

Clearly, any component of the regularized solution $x_{L,\lambda}$ in L's null space is unaffected by the smoothing norm $\|L\,x\|_2$, because

$$y \in \mathcal{N}(L) \quad \Rightarrow \quad \|L\,y\|_2 = 0 \quad \text{and} \quad \|A\,y - b\|_2^2 + \lambda^2\,\|L\,y\|_2^2 = \|A\,y - b\|_2^2.$$

If the null-space component $y \in \mathcal{N}(L)$ has components along the right singular vectors v_i associated with the small singular values of A, then we are in trouble because these components will not be filtered. Fortunately this is very unlikely; the null space of L is usually spanned by very smooth vectors that are well represented by the principal singular vectors v_1, v_2, etc. Hence, these are precisely the components that least need any filtering.

8.2 A Catalogue of Derivative Matrices*

In the previous section we already encountered the matrices L_1 and L_2 for the first and second derivatives. Both matrices are intentionally rectangular, with fewer rows than columns, thus endowing them with the null spaces in (8.6) and (8.7). These choices of L are "neutral" in the sense that they incorporate no assumptions about boundary conditions, i.e., about the behavior of the solution outside its domain.

Occasionally, it may be convenient to include boundary conditions (BC) in the derivative matrices in order to better control the regularization of the solution. Here we describe the cases of zero and reflexive boundary conditions, where the solution is assumed to be either zero outside the domain or reflected across the boundary. In both cases, we need to augment the matrices L_1 and L_2 from before. Chapters 6 and 8 in [8] give a thorough discussion of the choice of the matrix L as well as BC from a statistical perspective.

For zero BC, we simply extend the diagonals to obtain the matrices $L_1^z \in \mathbb{R}^{(n+1)\times n}$ and $L_2^z \in \mathbb{R}^{n\times n}$, given by

$$L_1^z = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{pmatrix}, \qquad L_2^z = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}.$$

The null spaces of both matrices are empty, because only the zero function satisfies the zero BC. For reflexive BC, using the technique from Section 7.4, we obtain the

matrices

$$L_1^r = L_1, \qquad L_2^r = \begin{pmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Both matrices have a one-dimensional null space

$$\mathcal{N}(L_1^r) = \mathcal{N}(L_2^r) = \text{span}\{(1, 1, \ldots, 1)^T\}$$

corresponding to the constant function. Alternative versions of the first-derivative matrices are discussed in Exercise 8.1. Which one to use always depends on the application.

We now turn to 2D problems, such as image deblurring, where the $N \times N$ matrix X represents a solution $f(t_1, t_2)$ living on a 2D grid with grid points $((i - \frac{1}{2})/N, (j - \frac{1}{2})/N)$, with $i, j = 1, \ldots, N$. Similar to the previous chapter, we assume that the elements of the solution vector x and the matrix X are related by (7.8), and we recall that $n = N^2$.

To illustrate the basic idea, let us consider how to compute representations of the second-order partial derivatives $\partial^2 f/\partial t_1^2$ and $\partial^2 f/\partial t_2^2$ for the case of zero BC, with the purpose of approximating the Laplacian of f.

With our conventions in (7.8), columns of X correspond to the direction t_1 in the sense that the elements of the column vector $X(:, j)$ represent samples of the function $f(t_1, (j - \frac{1}{2})/N)$ of the single variable t_1. It follows that the column vectors computed as $L_2^z X(:, j)$ are scaled approximations to the second derivatives of the function $f(t_1, (j - \frac{1}{2})/N)$. Thus, the columns of the matrix

$$L_2^z X = \big(L_2^z X(:, 1), L_2^z X(:, 2), \ldots, L_2^z X(:, n) \big)$$

represent the partial derivatives with respect to t_1 for a fixed $t_2 = (j - \frac{1}{2})/N$ for $j = 1, 2, \ldots, N$, and therefore the matrix $L_2^z X$ represents/approximates the partial derivative $\partial^2 f/\partial t_1^2$ sampled on the grid. With a similar argument, it follows that the rows of the matrix

$$X\,(L_2^z)^T = \begin{pmatrix} X(1, :)\,(L_2^z)^T \\ \vdots \\ X(N, :)\,(L_2^z)^T \end{pmatrix}$$

represent the partial derivatives with respect to t_2 for a fixed $t_1 = (i - \frac{1}{2})/N$ for $i = 1, 2, \ldots, N$, and thus the matrix $X\,(L_2^z)^T$ represents/approximates the partial derivative $\partial^2 f/\partial t_2^2$ on the grid. Given the above results, it follows immediately how to compute the approximation to the 2-norm of the Laplacian, given by

$$\|\nabla^2 f\|_2 = \left(\int_0^1 \int_0^1 \left(\frac{\partial^2 f}{\partial t_1^2} + \frac{\partial^2 f}{\partial t_2^2} \right)^2 dt_1\, dt_2 \right)^{1/2}.$$

The discrete smoothing norm with zero BC that corresponds to the Laplacian is thus $\|L_2^z X + X (L_2^z)^T\|_F$. The null space for this discrete smoothing norm is empty.

To use reflexive BC instead of zero BC, simply replace L_2^z with L_2^r in the above discrete smoothing norm. The null space for reflexive BC is spanned by the $N \times N$ matrix of all ones, corresponding to the constant function $f(t_1, t_2) = 1$.

We do not recommend using BC-free discrete derivative matrices for the Laplacian, because any harmonic function (i.e., a function f that satisfies Laplace's equation $\nabla^2 f = 0$) is in the null space of this smoothing norm, and this null space has infinite dimension. This is reflected in the fact that the dimension of the null space of the corresponding discrete smoothing norm is large; it is easy to check that the null space of $\|L_2 X(:, 2:N-1) + X(2:N-1, :) L_2^T\|_F$ has dimension $4(N-1)$. Discrete smoothing norms whose null spaces have large dimensions are not useful, because any component in the null space is not regularized.

For the same reason, one must be careful when using a smoothing norm based on mixed partial derivatives. Consider the smoothing norm $S(f) = \|\partial^2 f / \partial t_1 \partial t_2\|_2$. Any function $f(t_1, t_2)$ that depends only on t_1 or t_2 satisfies $\partial^2 f / \partial t_1 \partial t_2 = 0$, so the null space has infinite dimension. The discrete version of this smoothing norm takes the form $\|L_1 X L_1^T\|_F$ for both the BC-free case and for reflexive BC, and its null space has dimension $2N - 1$. For zero BC, on the other hand, the null space is empty.

The safest approach for incorporating discrete smoothing norms in 2D and 3D problems is perhaps to use the equivalent of the weighted Sobolev norm. As an example, consider the smoothing norm

$$S(f) = \left(\|\partial f / \partial t_1\|_2^2 + \|\partial^2 f / \partial t_2^2\|_2^2 \right)^{1/2},$$

which seeks to enforce a smoother solution in the t_2 direction than in the t_1 direction. With no assumptions about BC, the discrete smoothing norm takes the form $\left(\|L_1 X\|_F^2 + \|X L_2^T\|_F^2\right)^{1/2}$, and the corresponding null space of dimension two is spanned by

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 2 & \cdots & N \\ 1 & 2 & \cdots & N \\ \vdots & \vdots & & \vdots \\ 1 & 2 & \cdots & N \end{pmatrix},$$

which represent the constant and linear functions $f(t_1, t_2) = 1$ and $f(t_1, t_2) = t_2$. If we impose reflective BC, then only the matrix of all ones lies in the null space, and with zero BC the null space is empty.

8.3 The Generalized SVD

The SVD from Section 3.2 plays a central role throughout this book as the main analysis tool for studying the inverse problems and the regularization algorithms. In fact, most of the algorithms considered in the book are spectral filtering algorithms in the SVD basis. The main exception to this rule is Tikhonov regularization in general form (8.2) with a $p \times n$ matrix $L \neq I$. This is still a spectral filtering algorithm but in a different basis.

The *generalized SVD* (GSVD) provides the spectral basis we need for studying general-form Tikhonov regularization. Since two matrices A and L are involved, we need a joint decomposition of both. In this decomposition the matrices A and L must share the same right singular vectors x_i', since both matrices multiply x; cf. (8.2)–(8.3). In addition, we need two sets of left singular vectors u_i' and v_i' for the two matrices. These generalized singular vectors satisfy the following relations (similar to those (3.9) for the SVD):

$$\begin{aligned} p < n: \qquad & A\,x_i' = \sigma_i'\,u_i', \quad & L\,x_i' = \mu_i'\,v_i', \quad & \qquad i = 1, \ldots, p, \\ & A\,x_i' = u_i', & L\,x_i' = 0, & \qquad i = p+1, \ldots, n, \\ p \geq n: \qquad & A\,x_i' = \sigma_i'\,u_i', & L\,x_i' = \mu_i'\,v_i', & \qquad i = 1, \ldots, n. \end{aligned}$$

We emphasize that u_i', σ_i', and v_i' are not the ordinary SVD of A; we use the notation with a prime to indicate the kinship between the SVD and the GSVD.

In the GSVD relations above, σ_i' and μ_i' are nonnegative numbers which satisfy $\sigma_i'^2 + \mu_i'^2 = 1$, and their ratios σ_i'/μ_i' are called the generalized singular values. The two sets of left GSVD vectors u_i' and v_i' are orthonormal, conveniently allowing us to pull them out of the 2-norms in (8.2)–(8.3). The right singular vectors x_i' are linearly independent, but they are neither normalized nor orthogonal. The null space of L is spanned by those right singular vectors x_i' for which $\mu_i' = 0$ and, in the case $p < n$, also by the vectors $x_{p+1}', \ldots, x_n'$.

A word of caution is in order here. For historical reasons, the values $\sigma_1', \sigma_2', \ldots, \sigma_p'$ are ordered in nondecreasing order, i.e., in the *opposite* order of the singular values. Hence, the largest σ_i' and the smoothest GSVD vectors u_i', v_i', and x_i' are those for $i \approx p$. If $p < n$, then $x_{p+1}', \ldots, x_n'$ are null vectors for L and therefore also are very smooth.

The right singular vectors provide the basis for the general-form Tikhonov solution $x_{L,\lambda}$. Using the above relations between the GSVD vectors, it follows that

$$\begin{aligned} p < n: \qquad & x_{L,\lambda} = \sum_{i=1}^{p} \varphi_i^{[L,\lambda]}\, \frac{u_i'^T b}{\sigma_i'}\, x_i' + \sum_{i=p+1}^{n} u_i'^T b\, x_i', \\ p \geq n: \qquad & x_{L,\lambda} = \sum_{i=1}^{n} \varphi_i^{[L,\lambda]}\, \frac{u_i'^T b}{\sigma_i'}\, x_i', \end{aligned} \tag{8.8}$$

where $\varphi_i^{[L,\lambda]}$ are the general-form Tikhonov filter factors, given by

$$\varphi_i^{[L,\lambda]} = \frac{\sigma_i'^2}{\sigma_i'^2 + \lambda^2\,\mu_i'^2}, \qquad i = 1, \ldots, \min(p, n).$$

From the relations $\varphi_i^{[L,\lambda]} = (\sigma_i'/\mu_i')^2 / \big((\sigma_i'/\mu_i')^2 + \lambda^2\big)$ and $\sigma_i'^2 + \mu_i'^2 = 1$ it follows immediately that we have the approximation

$$\varphi_i^{[L,\lambda]} \approx \begin{cases} 1, & \sigma_i'/\mu_i' \gg \lambda, \\ \sigma_i'^2/\lambda^2, & \sigma_i'/\mu_i' \ll \lambda. \end{cases}$$

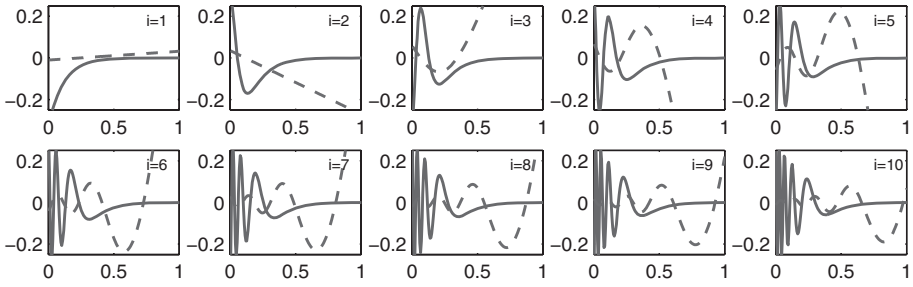

Figure 8.2. *Selected right singular vectors v_i (solid lines) and generalized singular vectors x'_{n-i+1} (dashed lines) for the depth profiling problem plotted versus the depth t. The latter vectors (which are scaled to fit the plot) are smoother than the former.*

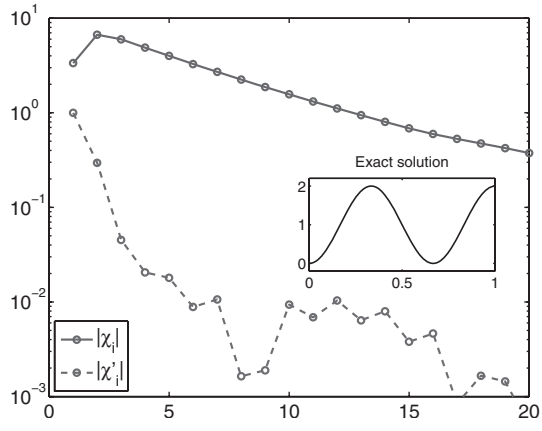

Figure 8.3. *The first* 20 *expansion coefficients for the exact solution (shown in the insert) in the SVD and GSVD bases.*

Like the standard-form filter factors, these filters therefore dampen contributions to the regularized solution from the small σ'_i, which are associated with the high-frequency components. The function `tikhonov` in *Regularization Tools* uses the GSVD and (8.8) for the general-form case. This choice was made for pedagogical reasons; it is not an efficient way to compute $x_{L,k}$.

To illustrate the additional smoothness that general-form Tikhonov regularization can provide in the solution, we return to the depth profiling example from Section 7.8. For $n = 256$ we construct the matrix A in (7.12), and we use the second-derivative smoothing matrix $L = L_2$ from (8.5).

Figure 8.2 shows the first 10 right singular vectors $v_1, v_2, \ldots, v_{10}$ and the last 10 right generalized singular vectors $x'_n, x'_{n-1}, \ldots, x'_{n-9}$. Recall that x'_n and x'_{n-1} span the null space of L. The generalized singular vectors are, indeed, smoother than the ordinary singular vectors. Hence, if we construct a smooth exact solution x^{exact} (see the insert in Figure 8.3), then we can expect that the expansion coefficients for

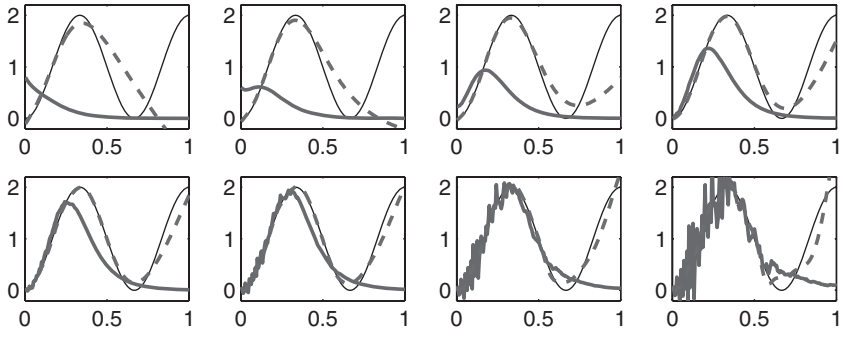

Figure 8.4. *Selected regularized solutions for the depth profiling problem: standard-form Tikhonov solutions x_λ (thick solid lines) and general-form Tikhonov solutions $x_{L,\lambda}$ (dashed lines) with second-derivative smoothing. The thin solid lines show the exact solution. The latter solutions give better reconstructions than the former.*

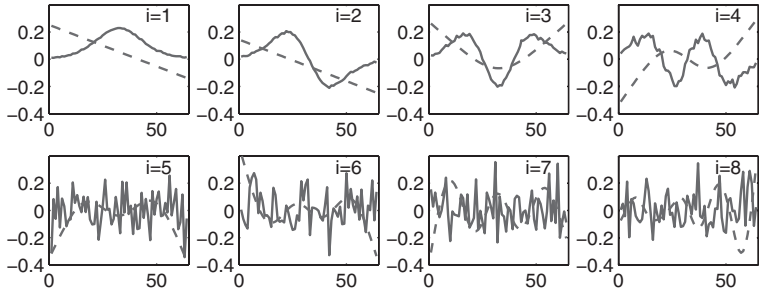

Figure 8.5. *The matrices are* `A = shaw(n) + 0.01*randn(n)` *(a noisy version of the coefficient matrix) and* `L = get_l(n,2)` *(the second derivative smoothing matrix) with $n = 64$. We show the right singular vectors $v_1, v_2, \ldots, v_{10}$ (solid lines) and the right generalized singular vectors $x'_{64}, x'_{63}, \ldots, x'_{57}$ (dashed lines). The latter (scaled to fit the plot) are much smoother than the former.*

x^{exact} in the GSVD basis decay faster than the coefficients in the SVD basis. This is confirmed by Figure 8.3, which shows the first 20 expansion coefficients computed as $\chi = V^T x^{\text{exact}}$ and $\chi' = (\, x'_1 \,, \ldots \,, x'_n \,)^{-1} x^{\text{exact}}$.

Figure 8.4 shows a few regularized solutions, namely, the standard-form Tikhonov solutions (for $L = I$) and the general-form Tikhonov solutions with $L = L_2$. The regularization parameters were chosen to show the progression from oversmoothing to undersmoothing. After the above GSVD analysis it is no surprise that general-form regularization provides a better basis with two advantages for the reconstruction: the noise is better suppressed, and the reconstruction reaches a deeper level before it deteriorates due to the noise. We have already discussed the difficulty of obtaining depth resolution in this problem (cf. Section 7.8), and this difficulty is still reflected in the general-form solutions.

Another example further illustrates the power of using a smoothing norm with $L \neq I$. We use the matrix A from the `shaw` test problem from Examples 3.6 with $n = 64$, and we add Gaussian noise with standard deviation 0.01 to each element of A in order to simulate a situation where the matrix involves noisy measurements (the noise level is exaggerated). Figure 8.5 shows the right singular vectors v_i and the right generalized singular vectors x'_{n-i+1} for $i = 1, \ldots, 8$; the latter are much smoother than the former. If we use standard-form regularization for this problem, then the noise in the coefficient matrix carries over to the reconstructions, and general-form regularization is therefore better suited for reconstruction of smooth solutions.

Exercises 8.3 and 8.4 provide other examples of the advantage of general-form regularization over the standard form.

8.4 Standard-Form Transformation and Smoothing Preconditioning

We have already seen that Tikhonov regularization in general form, with $L \neq I$, leads to the linear least squares problem (8.3) which can be solved with standard numerical methods. The challenge is to figure out how to use regularizing iterations, e.g., in the form of the CGLS method from Section 6.3.2, because the solution norm does not appear explicitly in these methods. What we need is a technique to transform the general-form problem (8.2)–(8.3) into a related Tikhonov problem in standard form

$$\min_{\bar{x}} \|\overline{A}\,\bar{x} - b\|_2^2 + \lambda^2 \|\bar{x}\|_2^2$$

which can be solved by CGLS, and whose solution $\bar{x}$ can easily be transformed back into the general-form setting. The computational overhead in the transformations must be reasonable.

We postpone the derivation of the standard-form transformation to the next section and state the main result here (danger: lots of equations ahead). We consider three cases of increasing "difficulty."

1. L is square and invertible. As already mentioned in Section 8.1, we can use the transformation

$$\overline{A} = A\,L^{-1} \qquad \text{and} \qquad x = L^{-1}\bar{x}. \tag{8.9}$$

   This is useful whenever operations with L^{-1} can be performed computationally efficiently, e.g., when L is banded or sparse.

2. L has full rank and more rows than columns (L is $p \times n$ with $p > n$). In this case we can replace the inverse of L with the Moore–Penrose pseudoinverse $L^\dagger$. A mathematically equivalent approach which is more efficient is obtained by realizing that if we compute the "skinny" QR factorization

$$L = Q_L\,R_L, \qquad \text{where } R_L \text{ is } n \times n \text{ and invertible,}$$

   then we have $\|L\,x\|_2 = \|Q_L\,R_L\,x\|_2 = \|R_L\,x\|_2$. Hence we can use the simpler standard-form transformation

$$\overline{A} = A\,R_L^{-1} \qquad \text{and} \qquad x = R_L^{-1}\,\bar{x}. \tag{8.10}$$

Again this is useful whenever L is banded or sparse or has some other form that allows us to compute the QR factorization efficiently. (The pseudoinverse of L is given by $L^\dagger = R_L^{-1} Q_L^T$.)

3. L has full rank and fewer rows than columns (L is $p \times n$ with $p < n$). In this case L has a nontrivial null space $\mathcal{N}(L)$ which we must handle properly. As explained in the next section, we must now use transformation

$$\overline{A} = A L^\# \qquad \text{and} \qquad x = L^\# \bar{x} + x_{\mathcal{N}}, \tag{8.11}$$

where $L^\#$ is the oblique (or A-weighted) pseudoinverse of L and $x_{\mathcal{N}}$ is the component of the solution in $\mathcal{N}(L)$. To compute these quantities, let the columns of the matrix W span the null space of L, i.e., $\mathcal{R}(W) = \mathcal{N}(L)$, and compute the two "skinny" QR factorizations

$$L^T = Q_{L^T} R_{L^T}, \qquad AW = Q_{AW} R_{AW}.$$

Then

$$L^\# = \left(I - W (AW)^\dagger A\right) L^\dagger = \left(I - W R_{AW}^{-1} Q_{AW}^T A\right) Q_{L^T} R_{L^T}^{-T} \tag{8.12}$$

and

$$x_{\mathcal{N}} = W (AW)^\dagger b = W R_{AW}^{-1} Q_{AW}^T b. \tag{8.13}$$

As long as the dimension of L's null space is low, the matrix AW is very skinny and the additional overhead involved with its QR factorization is negligible compared to the other two cases.

We note that for problems where L has structure (such as Hankel or Toeplitz) there are computationally efficient algorithms for performing the operations with L^{-1} or $L^\dagger$. It is beyond the scope of this book to go into the vast topic of structured matrix computations.

If we are in case 3, and L can be partitioned such that $L = (L_{\text{left}} , L_{\text{right}})$, where L_{left} is square and has full rank, then we have the computationally simpler expression

$$L^\# = \left(I - W T\right) \begin{pmatrix} L_{\text{left}}^{-1} \\ 0 \end{pmatrix}, \qquad T = (AW)^\dagger A = R_{AW}^{-1} Q_{AW}^T A.$$

This relation is used in *Regularization Tools*.

For problems that involve (weighted) Sobolev norms we can encounter rank-deficient L matrices with more rows than columns ($p > n$). For example, if L is obtained by stacking the two matrices L_1 and L_2 from (8.4)–(8.5), then L has the null space $\mathcal{N}(L) = \text{span}\{(1, \ldots, 1)^T\}$. When L is rank-deficient, we cannot rely on the QR factorization, and we must use the more general expressions

$$L^\# = \left(I - W (AW)^\dagger A\right) L^\dagger, \qquad x_{\mathcal{N}} = W (AW)^\dagger b. \tag{8.14}$$

How to deal with the pseudoinverse $L^\dagger$ depends on the problem and the particular structure of L.

To clearly see the role played by the matrix $L^{\#}$ (which may simplify to $L^{\dagger}$ or L^{-1}, depending on the properties of L), we first consider the iterates $\bar{x}^{(k)}$ from applying CGLS to the standard-form least squares problem $\min_{\bar{x}} \|\overline{A}\,\bar{x} - b\|_2$. Since $\bar{x}^{(k)}$ lies in the Krylov subspace $\operatorname{span}\left\{\overline{A}^T b, (\overline{A}^T\overline{A})\,\overline{A}^T b, \ldots, (\overline{A}^T\overline{A})^{k-1}\,\overline{A}^T b\right\}$, it follows that we can write $\bar{x}^{(k)}$ as

$$\bar{x}^{(k)} = \mathcal{P}_k\left(\overline{A}^T\overline{A}\right)\overline{A}^T b,$$

where $\mathcal{P}_k$ is a polynomial of degree at most $k-1$. If we insert $\overline{A} = A\,L^{\#}$ and $x^{(k)} = L^{\#}\bar{x}^{(k)} + x_{\mathcal{N}}$ into this expression, then we obtain

$$\begin{aligned} x^{(k)} &= L^{\#}\,\mathcal{P}_k\left((L^{\#})^T A^T A\,L^{\#}\right)(L^{\#})^T A^T b + x_{\mathcal{N}} \\ &= \mathcal{P}_k\left(M\,A^T A\right) M\,A^T b + x_{\mathcal{N}}, \end{aligned}$$

where we have introduced the symmetric matrix $M = L^{\#}(L^{\#})^T$. We see that the iterates $x^{(k)}$ produced in this fashion lie in the affine space

$$\operatorname{span}\{MA^T b, (MA^T A)\,MA^T b, (MA^T A)^2 MA^T b, \ldots, (MA^T A)^{k-1} MA^T b\} + x_{\mathcal{N}}.$$

Of course, we should not implement CGLS this way, but the above expression shows that the matrix M acts as a "preconditioner" for CGLS. This preconditioner is not a traditional one that accelerates convergence; its purpose is to modify the smoothness properties of the solution via the modification of the underlying Krylov subspace. For this reason, we refer to this type of preconditioning for CGLS as *smoothing preconditioning*.

The standard-form transformation described above thus allows us to use CGLS for any general-form problem via the smoothing preconditioning, provided, of course, that we can implement the operations with L^{-1}, $L^{\dagger}$, or $L^{\#}$ in a computationally efficient manner. This is extremely useful, since we can consider CGLS as quite a general "workhorse" for iterative regularization. At the same time, we emphasize that we cannot use this standard-form transformation to provide preconditioning for the iterative methods MR-II and RRGMRES from Section 6.3.4. The reason is that the transformed matrix $\overline{A}$ is, in general, not square, which is required for these methods. An alternative smoothing preconditioner for these methods, with no restrictions on the dimensions or rank of L, is given in [36]. The iterative regularization methods in *Regularization Tools* come with smoothing preconditioning based on these techniques.

8.5 The Quest for the Standard-Form Transformation*

In this section we derive the standard-form transformation from the previous section, using oblique projections and a splitting technique. The derivation is new and provides insight, so it deserves to appear in a section and not in an appendix, but it is not necessary to read this section in order to use the standard-form transformation or the smoothing preconditioner from the previous section.

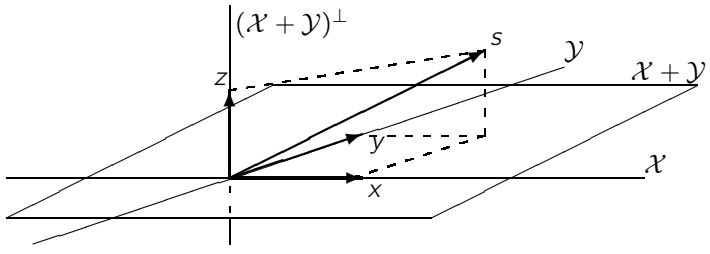

Figure 8.6. *Illustration of the oblique projection. The subspaces $\mathcal{X}$ and $\mathcal{Y}$ are represented by two solid lines, while the plane represents their sum $\mathcal{X}+\mathcal{Y}$, and the third (vertical) line represents the orthogonal complement of $\mathcal{X}+\mathcal{Y}$. The vector $x = E_{\mathcal{X},\mathcal{Y}}\, s$ is the oblique projection of s on $\mathcal{X}$ along $\mathcal{Y}$.*

8.5.1 Oblique Projections

We sidetrack for a bit into the world of oblique projections in order to establish the necessary mathematical framework. Recall that the orthogonal complement $\mathcal{X}^\perp$ of a subspace $\mathcal{X} \in \mathbb{R}^n$ is defined as $\mathcal{X}^\perp = \{y \in \mathbb{R}^n \mid x^T y = 0,\ x \in \mathcal{X}\}$.

Definition 1. Let $\mathcal{X}$ and $\mathcal{Y}$ be subspaces of $\mathbb{R}^n$ that intersect trivially, i.e., $\mathcal{X} \cap \mathcal{Y} = \{0\}$. Then the *projection on $\mathcal{X}$ along $\mathcal{Y}$* is the linear operator (the projector) $E_{\mathcal{X},\mathcal{Y}}$ which satisfies the following three requirements:

(a) $\forall\, x \in \mathcal{X}:\ E_{\mathcal{X},\mathcal{Y}}\, x = x$,

(b) $\forall\, y \in \mathcal{Y}:\ E_{\mathcal{X},\mathcal{Y}}\, y = 0$,

(c) $\forall\, z \in \mathbb{R}^n:\ E_{\mathcal{X},\mathcal{Y}}\, z \in \mathcal{X}$.

The *orthogonal* projector $P_{\mathcal{X}}$ on $\mathcal{X}$ corresponds to the special instance $\mathcal{Y} = \mathcal{X}^\perp$. Projectors that are not orthogonal are usually referred to as *oblique projectors*.

We can give a geometric interpretation of the above by noting that we can always decompose an arbitrary vector $s \in \mathbb{R}^n$ into the three components

$$x \in \mathcal{X}, \qquad y \in \mathcal{Y}, \qquad z \in (\mathcal{X}+\mathcal{Y})^\perp.$$

With this definition we have $x = E_{\mathcal{X},\mathcal{Y}}\, s$. Figure 8.6 illustrates this geometrical viewpoint. If $\mathcal{X}+\mathcal{Y} = \mathbb{R}^n$, then clearly $z = 0$. To perform computations with oblique projectors we need a matrix representation for $E_{\mathcal{X},\mathcal{Y}}$.

Theorem 1. Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ be subspaces with $\mathcal{X} \cap \mathcal{Y} = \{0\}$, and assume that we are given two matrices X and Y_0 such that

$$\mathcal{X} = \mathcal{R}(X), \qquad \mathcal{Y}^\perp = \mathcal{R}(Y_0). \tag{8.15}$$

Then the matrix representation of the projector $E_{\mathcal{X},\mathcal{Y}}$ is given by

$$E_{\mathcal{X},\mathcal{Y}} = X\,(Y_0^T X)^\dagger\, Y_0^T. \tag{8.16}$$

Note that $E_{\mathcal{X},\mathcal{Y}}$ is not symmetric. The well-known formula $P_{\mathcal{X}} = X\,X^{\dagger}$ for the orthogonal projector (which is symmetric) follows from the relation

$$P_{\mathcal{X}} = E_{\mathcal{X},\mathcal{X}^{\perp}} = X\,(X^T X)^{\dagger} X^T = X\,X^{\dagger}.$$

Theorem 2. If $P_{\mathcal{X}+\mathcal{Y}}$ denotes the orthogonal projector on $\mathcal{X}+\mathcal{Y}$, then

$$E_{\mathcal{X},\mathcal{Y}} + E_{\mathcal{Y},\mathcal{X}} = P_{\mathcal{X}+\mathcal{Y}} \qquad \text{and} \qquad E_{\mathcal{X},\mathcal{Y}} + E_{\mathcal{Y},\mathcal{X}} + P_{(\mathcal{X}+\mathcal{Y})^{\perp}} = I.$$

In particular, if $\mathcal{X}+\mathcal{Y} = \mathbb{R}^n$, then

$$E_{\mathcal{X},\mathcal{Y}}^T = E_{\mathcal{Y}^{\perp},\mathcal{X}^{\perp}} \qquad \text{and} \qquad E_{\mathcal{X},\mathcal{Y}} + E_{\mathcal{Y},\mathcal{X}} = I_n.$$

Orthogonal projectors are special cases with $P_{\mathcal{X}} + P_{\mathcal{X}^{\perp}} = I_n$.

Definition 2. If $x, y \in \mathbb{R}^n$, then we use the notation $x \perp_A y$ to denote

$$Ax \perp Ay \qquad \Leftrightarrow \qquad x^T A^T A\, y = 0.$$

Definition 3. If $\mathcal{X}$ is a subspace of $\mathbb{R}^n$, then we define the oblique complement:

$$\mathcal{X}^{\perp_A} = \{y \in \mathbb{R}^n :\ x \perp_A y \ \ \forall\, x \in \mathcal{X}\},$$

and we use the short notation $E_{\mathcal{X}} = E_{\mathcal{X},\mathcal{X}^{\perp_A}}$.

Theorem 3. If $\mathcal{X} = \mathcal{R}(X)$, then the matrix representation of $E_{\mathcal{X},\mathcal{X}^{\perp_A}}$ is

$$E_{\mathcal{X}} = X\,(AX)^{\dagger} A, \tag{8.17}$$

and we have a splitting $x = E_{\mathcal{X}}\,x + (I - E_{\mathcal{X}})\,x$ such that, for all $x \in \mathbb{R}^n$,

$$E_{\mathcal{X}}\,x \perp_A (I - E_{\mathcal{X}})\,x.$$

Definition 4. Given the $p \times n$ matrix X with $p \le n$, and a subspace $\mathcal{Y} \subset \mathbb{R}^n$ such that $\mathcal{X} = \mathcal{R}(X)$ and $\mathcal{Y}$ intersect trivially, we define the *oblique pseudoinverse* $X_{\mathcal{Y}}^{\dagger}$ as the $n \times p$ matrix

$$X_{\mathcal{Y}}^{\dagger} = E_{\mathcal{Y},\mathcal{N}(X)}\,X^{\dagger}. \tag{8.18}$$

In the special case $\mathcal{Y} = \mathcal{N}(X)^{\perp} = \mathcal{R}(X^T)$ we have $E_{\mathcal{N}(X)^{\perp},\mathcal{N}(X)} = P_{\mathcal{N}(X)^{\perp}}$, and thus the oblique pseudoinverse satisfies $X_{\mathcal{N}(X)^{\perp}}^{\dagger} = P_{\mathcal{N}(X)^{\perp}}\,X^{\dagger} = X^{\dagger}$ (the ordinary pseudoinverse).

Theorem 4. The oblique pseudoinverse $X_{\mathcal{Y}}^{\dagger}$ satisfies the three relations

$$X\,X_{\mathcal{Y}}^{\dagger} = P_{\mathcal{X}}, \qquad X_{\mathcal{Y}}^{\dagger}\,X = E_{\mathcal{Y},\mathcal{N}(X)}, \qquad X^{\dagger} = P_{\mathcal{R}(X^T)} X_{\mathcal{Y}}^{\dagger}, \tag{8.19}$$

and if $\mathcal{Y} = \mathcal{R}(Y)$, then $X_{\mathcal{Y}}^{\dagger} = Y\,(XY)^{\dagger}$.

8.5.2 Splitting of the Beast

What makes the standard-form transformation a nontrivial business is primarily the occurrence of a nontrivial null space of L. Whenever this is the case, we want to introduce a splitting of the solution such that one component is bound to lie in L's null space:

$$x = x_{\mathcal{M}} + x_{\mathcal{N}} \qquad \text{with} \qquad x_{\mathcal{N}} \in \mathcal{N}(L).$$

A first guess might be that the other component $x_{\mathcal{M}}$ should be orthogonal to $x_{\mathcal{N}}$, but this is not so. We insert the quantities

$$A\,x = A\,x_{\mathcal{M}} + A\,x_{\mathcal{N}} \qquad \text{and} \qquad L\,x = L\,x_{\mathcal{M}} + L\,x_{\mathcal{N}} = L\,x_{\mathcal{M}}$$

into the Tikhonov problem to obtain

$$\min \left\{ \|A\,x_{\mathcal{M}} + A\,x_{\mathcal{N}} - b\|_2^2 + \lambda^2 \, \|L\,x_{\mathcal{M}}\|_2^2 \right\}.$$

If we require that $A\,x_{\mathcal{M}}$ is orthogonal to $A\,x_{\mathcal{N}}$, then we obtain two *independent* least squares problems for the two components $x_{\mathcal{M}}$ and $x_{\mathcal{N}}$:

$$\min \left\{ \|A\,x_{\mathcal{M}} - b\|_2^2 + \lambda^2 \, \|L\,x_{\mathcal{N}}\|_2^2 \right\} \qquad \text{and} \qquad \min \|A\,x_{\mathcal{N}} - b\|_2^2.$$

Hence the correct requirement to $x_{\mathcal{M}}$ is that $x_{\mathcal{M}} \perp_A x_{\mathcal{N}}$, which corresponds to the splitting $\mathbb{R}^n = \mathcal{N}(L) + \mathcal{N}(L)^{\perp_A}$. The associated oblique projectors are

$$E_{\mathcal{M}} = I - E_{\mathcal{N}(L)}, \qquad E_{\mathcal{N}} = E_{\mathcal{N}(L)}, \qquad E_{\mathcal{M}} + E_{\mathcal{N}} = I$$

such that

$$x = x_{\mathcal{M}} + x_{\mathcal{N}} = E_{\mathcal{M}} x + E_{\mathcal{N}} x.$$

Note that $x_{\mathcal{M}} \not\perp x_{\mathcal{N}}$. Thus we arrive at two separate minimization problems:

1. The Tikhonov problem for $x_{\mathcal{M}} = E_{\mathcal{M}} x$,

$$\min \left\{ \|A\,E_{\mathcal{M}} x - b\|_2^2 + \lambda^2 \|L\,E_{\mathcal{M}} x\|_2^2 \right\}.$$

2. The unregularized least squares problem for $x_{\mathcal{N}} = E_{\mathcal{N}} x \in \mathcal{N}(L)$,

$$\min \|A\,E_{\mathcal{N}}\,x - b\|_2.$$

Let us first consider the second minimization problem. If we use the matrix W whose columns span the null space of L, i.e., $\mathcal{R}(W) = \mathcal{N}(L)$, then we can write $x_{\mathcal{N}} = W z$, which leads to

$$z = \operatorname{argmin} \|A\,W\,z - b\|_2 = (A\,W)^{\dagger}\,b,$$

and thus

$$x_{\mathcal{N}} = W\,(A\,W)^{\dagger}\,b = A^{\dagger}_{\mathcal{N}(L)} b.$$

This is the expression from (8.14) for the null-space component.

To finally arrive at the standard-form problem, we first note that due to (8.19) we can write

$$E_{\mathcal{M}} = E_{\mathcal{N}(L)^{\perp_A},\mathcal{N}(L)} = L^{\dagger}_{\mathcal{N}(L)^{\perp_A}} L.$$

The oblique pseudoinverse $L^{\dagger}_{\mathcal{N}(L)^{\perp_A}}$ is also called the A-weighted pseudoinverse, and it was given the short-hand notation $L^{\#}$ in the previous section. It can be computed as

$$\begin{aligned} L^{\dagger}_{\mathcal{N}(L)^{\perp_A}} &= E_{\mathcal{N}(L)^{\perp_A},\mathcal{N}(L)}\, L^{\dagger} \\ &= \left(I - E_{\mathcal{N}(L),\mathcal{N}(L)^{\perp_A}}\right) L^{\dagger} \\ &= \left(I - W\,(AW)^{\dagger} A\right) L^{\dagger}. \end{aligned}$$

Hence, in the Tikhonov problem we can insert

$$A\,E_{\mathcal{M}} = A\,L^{\dagger}_{\mathcal{N}(L)^{\perp_A}} L = \bar{A}\,L \qquad \text{with} \qquad \overline{A} = A\,L^{\dagger}_{\mathcal{N}(L)^{\perp_A}}.$$

Using another relation in (8.19), we also get

$$L\,E_{\mathcal{M}} = L\,L^{\dagger}_{\mathcal{N}(L)^{\perp_A}} L = P_{\mathcal{R}(L)}\,L = L.$$

Thus we arrive at the standard-form problem

$$\min\left\{\|\bar{A}\,\bar{x} - b\|_2^2 + \lambda^2\,\|\bar{x}\|_2\right\}, \qquad \overline{A} = A\,L^{\dagger}_{\mathcal{N}(L)^{\perp_A}}, \qquad \bar{x} = L\,x.$$

The key to obtaining this formulation is to use the oblique pseudoinverse $L^{\dagger}_{\mathcal{N}(L)^{\perp_A}}$. When L is $p \times n$ with $p > n$ and L has full rank, then the oblique pseudoinverse reduces to the ordinary pseudoinverse $L^{\dagger}$, and it further reduces to the inverse L^{-1} when L is square and invertible.

To summarize our results in this dense section, it is crucial to use the splitting $\mathbb{R}^n = \mathcal{N}(L) + \mathcal{N}(L)^{\perp_A}$. Only with this splitting of $\mathbb{R}^n$ are we able to split the residual norms of the Tikhonov problem into two separate least squares problems, as shown in Figure 8.7. The rest is manipulations with oblique projectors, etc.

8.6 Prelude to Total Variation Regularization

We finish this book with an important example of a smoothing norm that does not involve the 2-norm (which is explicitly or implicitly present in most of the other regularization methods that we have encountered). Once again, we start with the continuous formulation (2.2), and we define a smoothing norm which is the 1-norm of the gradient:

$$S_{\mathrm{TV}}(f) = \|f'\|_1 = \int_0^1 |f'(t)|\,dt. \tag{8.20}$$

This is known as the *total variation* (TV) of the function $f(t)$. The basic idea is that if we replace $S(f)^2$ in (8.1) with $S_{\mathrm{TV}}(f)$, then we still obtain smooth solutions, but the 1-norm in $S_{\mathrm{TV}}(f)$ penalizes nonsmoothness in a quite different manner than the 2-norm.

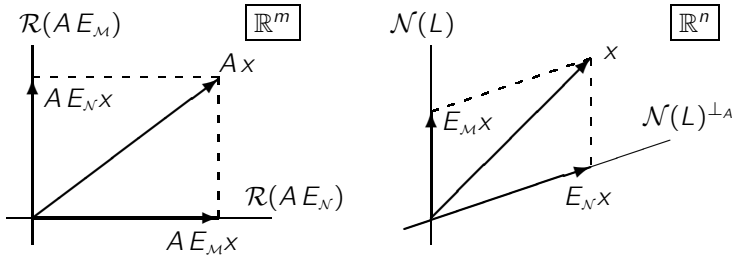

Figure 8.7. *Illustration of the splitting underlying the standard-form transformation. Only with the oblique splitting $\mathbb{R}^n = \mathcal{N}(L) + \mathcal{N}(L)^{\perp_A}$ are we able to split the residual norms of the Tikhonov problem into two separate least squares problems via the orthogonal splitting $\mathbb{R}^m = \mathcal{R}(A\,E_{\mathcal{M}}) + \mathcal{R}(A\,E_{\mathcal{N}})$.*

We illustrate the difference between the two smoothing norms with a simple example that involves the piecewise linear function

$$f(t) = \begin{cases} 0, & 0 \le t < \frac{1}{2}(1-h), \\ \dfrac{t}{h} - \dfrac{1-h}{2h}, & \frac{1}{2}(1-h) \le t \le \frac{1}{2}(1+h), \\ 1, & \frac{1}{2}(1+h) < t \le 1, \end{cases}$$

which increases linearly from 0 to 1 in the interval $[0, 1]$; see Figure 8.8. It is easy to show that the smoothing norms associated with the 1- and 2-norms of $f'(t)$ satisfy

$$\|f'\|_1 = \int_0^1 |f'(t)|\,dt = \int_0^h \frac{1}{h}\,dt = 1,$$

$$\|f'\|_2^2 = \int_0^1 f'(t)^2\,dt = \int_0^h \frac{1}{h^2}\,dt = \frac{1}{h}.$$

We see that the total variation smoothing norm $S_{\mathrm{TV}}(f) = \|f'\|_1$ is independent of the slope of the middle part of $f(t)$, while the smoothing norm based on the 2-norm penalizes steep gradients (when h is small). The 2-norm of $f'(t)$ will not allow any steep gradients, and therefore it produces a very smooth solution (in Exercise 4.7 we saw the same behavior for the 2-norm of $f(t)$). The 1-norm, on the other hand, allows some steep gradients—but not too many—and it is therefore able to produce a less smooth solution, such as a piecewise smooth solution.

Assuming that we use a quadrature discretization, the discrete version of TV regularization takes the form of the following minimization problem:

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L_1\,x\|_1 \right\}.$$

Like Tikhonov regularization, this is a convex optimization problem with a unique solution, but the second term involving the 1-norm is not differentiable everywhere with respect to x.

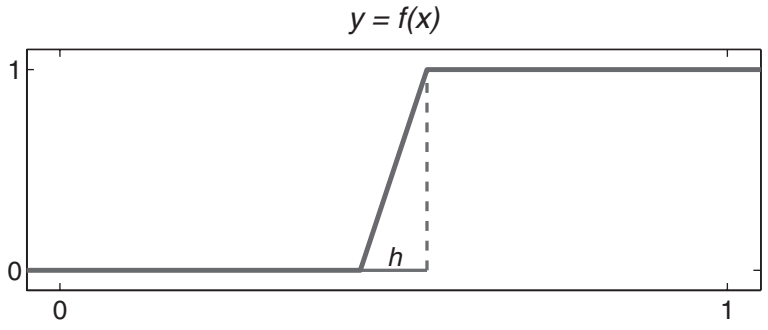

Figure 8.8. *A piecewise linear function $f(t)$ with a slope $f'(t) = 1/h$ at the nonhorizontal part. For this function $\|f'\|_1 = 1$ and $|f'\|_2^2 = 1/h$.*

The barcode deconvolution problem from Section 7.1 provides a good example of the use of discrete TV regularization, since we want to reconstruct a piecewise constant function (the exact barcode signal x^{exact}). We use the same problem as in Figure 7.1 except that we use a higher noise level $\eta = 10^{-4}$ to make the problem more difficult to solve. Figure 8.9 shows the exact and blurred signals, together with the best TSVD and total variation reconstructions, for $k = 120$ and $\lambda = 0.037$, respectively. Even the best TSVD solution does not give a reliable reconstruction, while the total variation solution is quite close to the exact solution. For such problems it is a clear advantage to use total variation.

In two dimensions, given a function $f(\mathbf{t})$ with $\mathbf{t} = (t_1, t_2)$, we replace the absolute value of the gradient in (8.20) with the gradient magnitude, defined as

$$|\nabla f| = \left(\left(\frac{\partial f}{\partial t_1} \right)^2 + \left(\frac{\partial f}{\partial t_2} \right)^2 \right)^{1/2},$$

to obtain the 2D version of the total variation smoothing norm $S_{\text{TV}}(f) = \big\| |\nabla f| \big\|_1$. The relevant smoothing norms of $f(\mathbf{t})$ are now

$$\big\| |\nabla f| \big\|_1 = \int_0^1 \int_0^1 |\nabla f| \, dt_1 \, dt_2 = \int_0^1 \int_0^1 \left(\left(\frac{\partial f}{\partial t_1} \right)^2 + \left(\frac{\partial f}{\partial t_2} \right)^2 \right)^{1/2} dt_1 \, dt_2,$$

$$\big\| |\nabla f| \big\|_2^2 = \int_0^1 \int_0^1 |\nabla f|^2 \, dt_1 \, dt_2 = \int_0^1 \int_0^1 \left(\frac{\partial f}{\partial t_1} \right)^2 + \left(\frac{\partial f}{\partial t_2} \right)^2 dt_1 \, dt_2.$$

To illustrate the difference between these two smoothing norms, consider a function $f(\mathbf{t})$ with the polar representation

$$f(r, \theta) = \begin{cases} 1, & 0 \le r < R, \\ 1 + \frac{R}{h} - \frac{r}{h}, & R \le r \le R + h, \\ 0, & R + h < r. \end{cases}$$

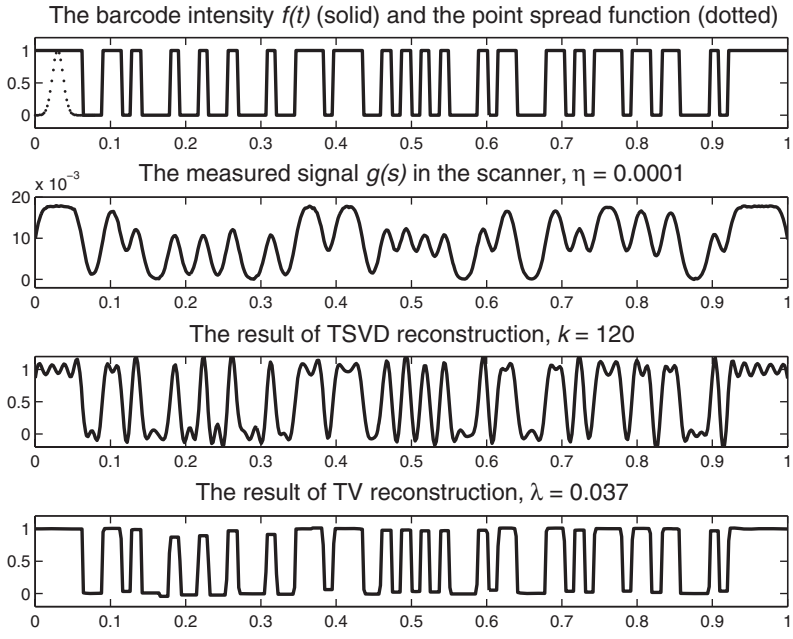

Figure 8.9. *Same barcode example as in Figure* 7.1 *except the noise level is larger. Top: The exact solution (the printed barcode) together with the point spread function. Second from top: The recorded noisy signal. Third from top: Best TSVD reconstruction for* $k = 120$. *Bottom: Best total variation reconstruction for* $\lambda = 0.037$.

This function is 1 inside the disk with radius $r = R$, zero outside the disk with radius $r = R + h$, and has a linear radial slope between 0 and 1; see Figure 8.10. In the area between these two disks the gradient magnitude is $|\nabla f| = 1/h$, and elsewhere it is zero. Thus we obtain

$$\left\| |\nabla f| \right\|_1 = \int_0^{2\pi} \int_R^{R+h} \frac{1}{h}\, r\, dr\, d\theta = 2\pi R + \pi h,$$

$$\left\| |\nabla f| \right\|_2^2 = \int_0^{2\pi} \int_R^{R+h} \frac{1}{h^2}\, r\, dr\, d\theta = \frac{2\pi R}{h} + \pi.$$

Similar to the one-dimensional example, we see that the total variation smoothing norm is almost independent of the size of the gradient, while the 2-norm penalizes steep gradients. In fact, as $h \to 0$, we see that $S_{\mathrm{TV}}(f)$ converges to the circumference $2\pi R$.

For this reason, if we want to reconstruct a 2D solution (an image) which is allowed to have some, but not too many, steep gradients, then we should use the total variation smoothing norm $S_{\mathrm{TV}}(f) = \left\| |\nabla f| \right\|_1$. In practice, this smoothing norm tends to produce piecewise constant reconstructions in one dimension, and "blocky" recon-

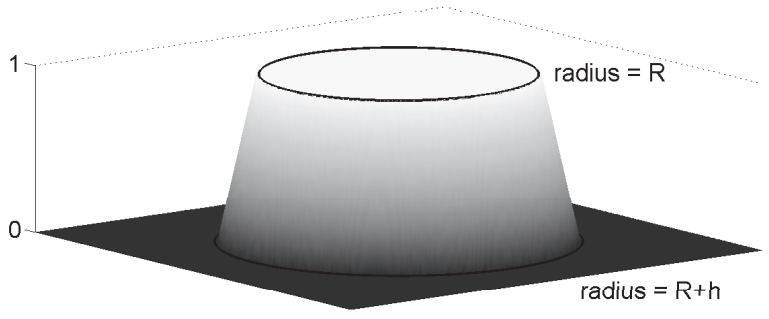

Figure 8.10. *A* 2D *function* $f(\mathbf{t})$ *whose gradient magnitude is* $|\nabla f| = 1/h$ *in the ring-formed area. For this function* $\||\nabla f|\|_1 = 2\pi R + \pi h$ *and* $\||\nabla f|\|_2^2 = 2\pi R/h + \pi$.

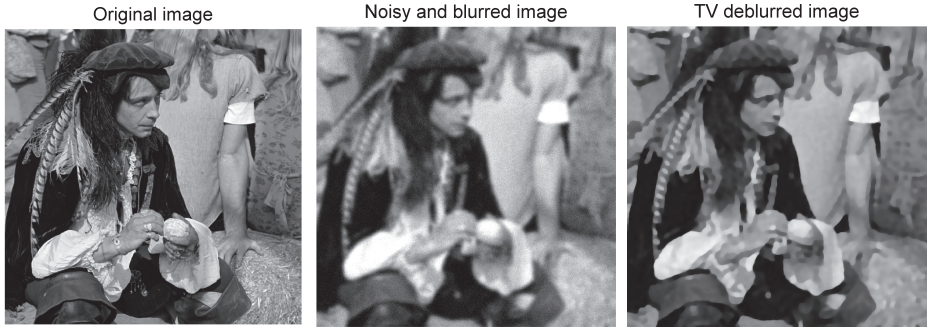

Figure 8.11. *Illustration of total variation image deblurring.*

structions (images) in 2D. This has made total variation regularization very popular in some image reconstruction problems.

We finish with an example that illustrates the use of total variation regularization for image deblurring. Figure 8.11 shows a sharp image, a noisy and blurred version, and a total variation reconstruction computed by means of the function `TVdeblur` in the MATLAB package *mxTV* [13]. The edges in the original image are well reconstructed in the deblurred image, but a close inspection shows some unavoidable blocking artifacts due to the total variation smoothing norm.

8.7 And the Story Continues . . .

The last chapter is a brief introduction to a much larger world of regularization algorithms than we have considered in the previous chapters—a world where the reader must leave behind the safe grounds of the SVD. We focused on Tikhonov regularization in the general form and showed how to incorporate requirements about additional smoothness via discrete smoothing norms. We also showed how to use smoothing preconditioners in iterative regularization methods through the introduc-

tion of standard-form transformations and oblique pseudoinverses. We also hinted at ways to implement such methods efficiently. Finally, we gave a brief introduction to total variation regularization, which uses a quite different kind of smoothing norm.

This book does not intend to tell the whole story about the regularization of discrete inverse problems. Rather, the goal is to set the stage and provide the reader with a solid understanding of important practical and computational aspects. The book therefore stops here, because the practical use and implementation of regularization methods with general smoothing norms, especially for large-scale problems, tend to be very problem dependent. But the story continues.

Exercises

8.1. **Alternative First-Derivative Smoothing Norms**
The matrices L_1, L_1^z, and L_1^r introduced in Sections 8.1 and 8.2 are based on differences between neighboring elements of the solution vector x, representing samples of the solution at the grid points $t_j = (j - \frac{1}{2})/n$, $j = 1, \ldots, n$. Alternatively, we can use central differences that involve elements with a distance of two, i.e., x_{j-1} and x_{j+1}. Show that the corresponding matrices for the first derivative are given by

$$\widehat{L}_1 = \begin{pmatrix} -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \end{pmatrix}, \quad \widehat{L}_1^z = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{pmatrix},$$

and

$$\widehat{L}_1^r = \begin{pmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{pmatrix}.$$

Also, show that the null spaces of these matrices are as follows:

- $\mathcal{N}(\widehat{L}_1) = \text{span}\{ (1, 1, \ldots, 1)^T, (1, -1, 1, -1, \ldots)^T \}$,
- $\mathcal{N}(\widehat{L}_1^z) = \begin{cases} \text{trivial null space}, & n \text{ even}, \\ \text{span}\{(1, 0, 1, 0, \ldots)^T\}, & n \text{ odd}, \end{cases}$
- $\mathcal{N}(\widehat{L}_1^r) = \text{span}\{ (1, 1, \ldots, 1)^T \}$.

Why is it dangerous to use $\widehat{L}_1$ and $\widehat{L}_1^z$ in connection with smoothing norms?

8.2. **Periodic Derivative Matrices**
Write the derivative matrices for first and second orders, for periodic boundary conditions. What are the null spaces?

8.3. **General-Form Tikhonov Regularization**
This exercise illustrates that standard-form regularization, corresponding to $L = I$, is not always a good choice in regularization problems. The test problem here is the inverse Laplace transform, which is implemented in *Regularization Tools* as the function `i_laplace`. The integral equation takes the form

$$\int_0^\infty \exp(-s\,t)\,f(t)\,dt = g(s)\ , \qquad 0 \le s \le \infty,$$

with right-hand side g and solution f given by

$$g(s) = \frac{1}{s} - \frac{1}{s+1/2}\ , \qquad f(t) = 1 - \exp(-t/2).$$

Use the call `[A,b,x] = i_laplace(n,2)` to produce this test problem. Also generate the first derivative smoothing matrix L_1 from (8.4) by means of `L = get_l(n,1)`.

Add Gaussian white noise with standard deviation $\eta = 10^{-4}$ to the right-hand side. Then compute ordinary Tikhonov solutions and vary the regularization parameter λ until the noise starts to dominate x_λ, and plot the solutions. Notice that none of these solutions are able to reconstruct the exact solution very well in the rightmost part of the plot.

Finally, compute the general-form Tikhonov solutions by means of the function `tikhonov`. You should use the calls

```
[U,sm,X] = cgsvd(A,L);
Xreg = tikhonov(U,sm,X,b,lambda);
```

Again, vary λ until the noise starts to dominate $x_{L,\lambda}$. This approach should be able to yield better approximations to the exact solution. Study the right generalized singular vectors x_i' and explain why.

8.4. **Preconditioned CGLS**
This is another example which illustrates that $L = I_n$ is not always a good choice in regularization problems. We use the second derivative test problem `deriv2(n)` from *Regularization Tools* (see Exercise 2.3), which requires an L matrix that approximates the first derivative operator.

Generate the test problem `deriv2(n,2)` with $n = 100$, add noise to the right-hand side with relative noise level `rnl` $= 10^{-3}$, and study the Picard plot. According to this plot, the problem looks innocent; nothing here reveals that the choice $L = I_n$ might be bad.

Then compute the TSVD solutions and plot them. Notice that none of them are able to reconstruct the exact solution very well in the rightmost part of the plot. This is due to the particular structure in the right singular vectors v_i; plot some of these vectors and explain which structure hurts the reconstruction.

Now generate the matrix L_1 (8.4) and a basis W for its null space by means of the call `[L,W] = get_l(n,1)`. Then use the preconditioned CGLS method, implemented in the function `pcgls`, to compute the regularized solution with the smoothing norm $\|L_1 \cdot\|_2$. This approach should produce better reconstructions. Study the generalized right singular vectors x_i' and explain why.

Appendix A

Linear Algebra Stuff

Span, Column Space, and Null Space

The *span* of a set of k vectors $a_i \in \mathbb{R}^m$ is the linear subspace consisting of all linear combinations of the vectors:

$$\text{span}\{a_1, \ldots, a_k\} = \left\{ y \in \mathbb{R}^m \,|\, y = x_1\, a_1 + \cdots + x_k\, a_k,\ x_i \in \mathbb{R} \right\}.$$

Given a real square matrix $A \in \mathbb{R}^{m \times n}$, we define the *column space* (or *range*) of A as the subspace spanned by the columns of A:

$$\mathcal{R}(A) = \text{span}\{a_1, a_2, \ldots, a_n\} = \left\{ y \in \mathbb{R}^m \,|\, y = A\,x,\ x \in \mathbb{R}^n \right\}.$$

We say that a linear system $A\,x = b$ is compatible if the right-hand side b lies in the column space of A, i.e., $b \in \mathcal{R}(A)$. If $m \leq n$ and A has full rank, then $\mathcal{R}(A) = \mathbb{R}^n$ and all systems are compatible. We define the *null space* of A as the subspace spanned by the null vectors, i.e., those vectors x for which $A\,x = 0$:

$$\mathcal{N}(A) = \left\{ x \in \mathbb{R}^n | A\,x = 0 \right\}.$$

If $m \geq n$ and A has full rank, then it has the trivial null space $\mathcal{N}(A) = \{0\}$.

Orthogonal Matrices and Projections

A real, square matrix $U \in \mathbb{R}^{n \times n}$ is *orthogonal* if its inverse equals its transpose, $U^{-1} = U^T$. Consequently we have the two relations

$$U^T U = I \qquad \text{and} \qquad U\,U^T = I.$$

The columns of U are *orthonormal*, i.e., they are orthogonal and the 2-norm of each column is one. To see this, let u_i denote the ith column of U so that $U = (\, u_1, \cdots, u_n\,)$. Then the relation $U^T U = I$ implies that

$$u_i^T\, u_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

An orthogonal matrix is perfectly well conditioned; its condition number is one. Moreover, any operation with its inverse merely involves a matrix product with its transpose.

An *orthogonal transformation* is accomplished by multiplication with an orthogonal matrix. Such a transformation leaves the 2-norm unchanged, because

$$\|U\,x\|_2 = \left((U\,x)^T(U\,x)\right)^{1/2} = (x^T x)^{1/2} = \|x\|_2.$$

An orthogonal transformation can be considered as a change of basis between the "canonical" basis $e_1, e_2, \ldots, e_n$ in $\mathbb{R}^n$ (where e_i is the ith column of the identity matrix) and the basis $u_1, u_2, \ldots, u_n$ given by the orthonormal columns of U. Specifically, for an arbitrary vector $x \in \mathbb{R}^n$ we can find scalars $z_1, \ldots, z_n$ so that

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{i=1}^{n} x_i\, e_i = \sum_{i=1}^{n} z_i\, u_i = U\,z,$$

and it follows immediately that the coordinates z_i in the new basis are the elements of the vector

$$z = U^T x.$$

Because they do not distort the size of vectors, orthogonal transformations are valuable tools in numerical computations.

For any k less than n, define the matrix $U_k = (\,u_1, \cdots, u_k\,) \in \mathbb{R}^{n\times k}$; then

$$\|U_k\,x\|_2 = \left\| U \begin{pmatrix} x \\ 0 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} x \\ 0 \end{pmatrix} \right\|_2 = \|x\|_2$$

but

$$\|U_k^T y\|_2 = \left(\textstyle\sum_{i=1}^{k} (u_i^T y)^2\right)^{1/2} \le \left(\textstyle\sum_{i=1}^{n} (u_i^T y)^2\right)^{1/2} = \|y\|_2.$$

Hence, only multiplication with U_k leaves the 2-norm unchanged. Now define the subspace $\mathcal{S}_k = \text{span}\{u_1, \ldots, u_k\}$. The *orthogonal projection* $x_{\mathcal{S}_k} \in \mathbb{R}^n$ of an arbitrary vector $x \in \mathbb{R}^n$ onto this subspace is the unique vector in $\mathcal{S}_k$ which is closest to x in the 2-norm, and it is computed as

$$x_{\mathcal{S}_k} = U_k\,U_k^T x, \qquad \text{with} \qquad U_k = (\,u_1, \cdots, u_k\,).$$

The matrix $U_k\,U_k^T$, which is $n \times n$ and has rank k, is called an *orthogonal projector*.

Orthogonal matrices and projectors arise naturally in connection with the least squares problem:

$$\min_x \|A\,x - b\|_2, \qquad \text{where} \qquad A \in \mathbb{R}^{m\times n} \quad \text{and} \quad m > n.$$

If A has full rank, then the least squares solution x_{LS} is unique, and it is conveniently computed via the *QR factorization* of A:

$$A = U \begin{pmatrix} R \\ 0 \end{pmatrix} = U_n\,R, \qquad R \in \mathbb{R}^{n\times n}, \qquad U_n = (\,u_1, \cdots, u_n\,) \in \mathbb{R}^{m\times n},$$

where $U \in \mathbb{R}^{m \times m}$ is orthogonal, R is upper triangular, and the columns of U_n form an orthonormal basis for the column space of A. Then x_{LS} is the solution to the compatible system $A\,x = b_n$, where $b_n = U_n\,U_n^T\,b$ is the orthogonal projection of b onto the column space of A. By inserting A's QR factorization, it follows immediately that

$$x_{\mathrm{LS}} = R^{-1} U_n^T\, b.$$

The corresponding least squares residual is $b - A\,x_{\mathrm{LS}} = b - b_n = (I - U_n\,U_n^T)\,b$. We note that the pseudoinverse of A can be written as $A^\dagger = (\,R^{-1}\,,\,0\,)\,U^T = R^{-1}\,U_n^T$ and that the least squares solution is formally given by $x_{\mathrm{LS}} = A^\dagger b$.

The Spectral Decomposition and the SVD

A real, symmetric matrix $A = A^T$ always has an *eigenvalue decomposition* (or *spectral decomposition*) of the form

$$A = U\,\Lambda\,U^T,$$

where U is orthogonal, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix whose diagonal elements λ_i are the *eigenvalues* of A. A real symmetric matrix always has real eigenvalues. The columns u_i of U are the *eigenvectors* of A, and the eigenpairs (λ_i, u_i) satisfy

$$A\,u_i = \lambda_i\,u_i, \qquad i = 1, \ldots, n.$$

The matrix A represents a linear mapping from $\mathbb{R}^n$ onto itself, and the geometric interpretation of the eigenvalue decomposition is that U represents a new, orthonormal basis in which this mapping is the diagonal matrix Λ. In particular, each basis vector u_i is mapped to a vector in the same direction, namely, the vector $A\,u_i = \lambda_i\,u_i$.

A real square matrix which is not symmetric can not be diagonalized by an orthogonal matrix. It takes two orthogonal matrices U and V to diagonalize such a matrix, by means of the *singular value decomposition*,

$$A = U\,\Sigma\,V^T = \sum_{i=1}^{n} u_i\,\sigma_i\,v_i^T,$$

where $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a real diagonal matrix whose diagonal elements σ_i are the *singular values* of A, while the *singular vectors* u_i and v_i are the columns of the orthogonal matrices U and V. The singular values are nonnegative and are typically written in nonincreasing order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. We note that if A is symmetric, then its singular values are equal to the absolute values of its eigenvalues.

The geometric interpretation of the SVD is that it provides two sets of orthogonal basis vectors—the columns of U and V—such that the mapping represented by A becomes a diagonal matrix when expressed in these bases. Specifically, we have

$$A\,v_i = \sigma_i\,u_i, \qquad i = 1, \ldots, n.$$

That is, σ_i is the "magnification" when mapping v_i onto u_i. Any vector $x \in \mathbb{R}^n$ can be written as $x = \sum_{i=1}^{n} (v_i^T x)\,v_i$, and it follows that its image is given by

$$A\,x = \sum_{i=1}^{n} (v_i^T x)\,A\,v_i = \sum_{i=1}^{n} \sigma_i\,(v_i^T x)\,u_i.$$

If A has an inverse, then the mapping of the inverse also defines a diagonal matrix:

$$A^{-1}\, u_i = \sigma_i^{-1}\, v_i,$$

so that σ_i^{-1} is the "magnification" when mapping u_i back onto v_i. Similarly, any vector $b \in \mathbb{R}^n$ can be written as $b = \sum_{i=1}^{n} (u_i^T b)\, u_i$, and it follows that the vector $A^{-1}\, b$ is given by

$$A^{-1}\, b = \sum_{i=1}^{n} (u_i^T b)\, A^{-1}\, u_i = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i}\, v_i.$$

Rank, Conditioning, and Truncated SVD

The *rank* of a matrix is equal to the number of nonzero singular values: $r = \text{rank}(A)$ means that

$$\sigma_r > 0, \qquad \sigma_{r+1} = 0.$$

The matrix A has *full rank* (and, therefore, an inverse) only if all of its singular values are nonzero. If A is rank deficient, then the system $A\,x = b$ may not be compatible; in other words, there may be no vector x that solves the problem. The columns of $U_r = (\, u_1\,,\, u_2\,,\, \ldots\,,\, u_r\,)$ form an orthonormal basis for the column space of A, and the system $A\,x = b_r$ with $b_r = U_r\, U_r^T b$ is the closest compatible system. This compatible system has infinitely many solutions, and the solution of minimum 2-norm is

$$x_r = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i}\, v_i.$$

Consider now a perturbed version $A\,\tilde{x} = \tilde{b}$ of the original system $A\,x = b$, in which the perturbed right-hand side is given by $\tilde{b} = b + e$. If A has full rank, then the perturbed solution is given by $\tilde{x} = A^{-1}\tilde{b} = x + A^{-1}e$, and we need an upper bound for the relative perturbation $\|x - \tilde{x}\|_2 / \|x\|_2$. The worst-case situation arises when b is in the direction of the left singular vector u_1 while the perturbation e is solely in the direction of u_n, and it follows that the perturbation bound is given by

$$\frac{\|x - \tilde{x}\|_2}{\|x\|_2} \le \text{cond}(A)\, \frac{\|e\|_2}{\|b\|_2}, \qquad \text{where} \qquad \text{cond}(A) = \frac{\sigma_1}{\sigma_n}.$$

The quantity $\text{cond}(A)$ is the *condition number* of A. The larger the condition number, the more sensitive the system is to perturbations of the right-hand side.

The smallest singular value σ_n measures how "close" A is to a singular matrix (and $\sigma_n = 0$ when A is singular). A perturbation of A with a matrix E, whose elements are of the order σ_n, can make A rank-deficient. The existence of one or more small singular values (small compared to the largest singular value σ_1) therefore indicates that A is "almost" singular.

Appendix B

Symmetric Toeplitz-Plus-Hankel Matrices and the DCT

A Toeplitz matrix T has identical elements along all its diagonals, meaning that $t_{ij} = t_{i+\ell,j+\ell}$ for all relevant integers i, j, and ℓ. If T is symmetric, then in addition we have $t_{ij} = t_{ji}$ for all i and j. Two examples:

$$\text{general } T = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 5 & 4 & 3 & 2 \\ 6 & 5 & 4 & 3 \\ 7 & 6 & 5 & 4 \end{pmatrix}, \qquad \text{symmetric } T = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

A Hankel matrix H has identical elements along all its antidiagonals, meaning that $h_{ij} = h_{i+\ell,j-\ell}$ for all relevant integers i, j, and ℓ. A Hankel matrix is symmetric by definition. Example:

$$H = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}.$$

A matrix A is *symmetric Toeplitz-plus-Hankel* (STH) if it is a sum of a symmetric Toeplitz matrix and a Hankel matrix, where the latter is obtained from the former by extending the diagonals beyond the matrix borders and flipping them inside, as shown in this example:

$$T = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix} \rightarrow \left(\begin{array}{ccc|cccc|ccc} 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{array}\right) \rightarrow H = \begin{pmatrix} 3 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

$$\rightarrow A = T + H = \begin{pmatrix} 7 & 5 & 3 & 1 \\ 5 & 5 & 3 & 3 \\ 3 & 3 & 5 & 5 \\ 1 & 3 & 5 & 7 \end{pmatrix}.$$

An $n \times n$ STH matrix can be written as the sum of n basis matrices $A_0, \ldots, A_{n-1}$:

$$A = \sum_{k=0}^{n-1} \alpha_k A_k,$$

in which the coefficients α_k are the elements in the first column of the Toeplitz matrix, $\alpha_k = t_{k+1,1}$. The elements of the basis matrix A_k are 1 if $|i-j| = k$, $i+j = k+1$, or $i+j = 2n-k+1$; all other elements are 0. For example, for $n = 4$ the matrices A_0, A_1, A_2, and A_3 take the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

The discrete cosine transform (DCT) is a linear transformation between a vector (e.g., representing a sampled signal) and its coefficients in a discrete cosine basis. It arises in connection with interpolation by means of cosine functions and many other applications; see [7]. DCTs come in several variants; the one used in this book is based on sampling the cosine functions in the n points

$$\tau_\ell = (\ell - \tfrac{1}{2})\ \pi/n, \qquad \ell = 1, \ldots, n,$$

leading to the discrete cosine vectors $w_1, w_2, \ldots, w_n$ given by

$$w_{j+1} = \varpi_j \left(\cos(j\tau_1), \cos(j\tau_2), \ldots, \cos(j\tau_n)\right)^T, \quad j = 0, 1, \ldots, n-1,$$

in which

$$\varpi_0 = \sqrt{1/n} \qquad \text{and} \qquad \varpi_1 = \cdots = \varpi_{n-1} = \sqrt{2/n}.$$

Figure 6.4 shows the first 10 DCT basis vectors.

The $n \times n$ DCT matrix is then defined as $W_n = (w_1, \ldots, w_n)$; it is orthogonal ($W_n^T W_n = I$), and it follows immediately that the relations between a vector x and its coefficients $\hat{x}$ in the discrete cosine basis are

$$\hat{x} = \mathrm{dct}(x) = W_n^T x \qquad \Leftrightarrow \qquad x = \mathrm{idtc}(\hat{x}) = W_n \hat{x}.$$

We emphasize that the DCT and its inverse are always computed via $O(n \log n)$ algorithms!

We will now show that the discrete cosine vectors w_{j+1} are eigenvectors for each of the STH basis matrices A_k. This is trivially the case for $A_0 = I$ (the identity matrix) because $A_0\, w_{j+1} = w_{j+1}$ for all j. For $k = 1, \ldots, n-1$ consider the inner product between the ith row of A_k and w_{j+1}; clearly this inner product has only two contributions, from the two 1's in the row. Let us allow the elements of the Hankel part of A_k to be wrapped outside, leading to indices beyond the range $1, \ldots, n$. For example, for $k = 2$ the extended matrix becomes

$$A_2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \rightarrow \left(\begin{array}{ccc|cccc|ccc} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

It follows that the two nonzero contributions come from the two elements $(i, i-k)$ and $(i, i+k)$ in this extended notation. We extend w_{j+1} in a similar fashion, to be compatible with the extended basis matrix,

$$w_{j+1}^T \to (w_{j+1}(n{-}1{:}\,{-}1{:}\,1)^T,\ w_{j+1}^T,\ w_{j+1}(n{:}\,{-}1{:}\,2)^T),$$

and from the definition of τ_ℓ it follows that we can rewrite this extended vector as

$$\varpi_j\big(\cos(j\tau_{2-n}), \ldots, \cos(\tau_0),\ \cos(\tau_1), \ldots, \cos(\tau_n), \cos(j\tau_{n+1}), \ldots, \cos(j\tau_{2n-1})\big)^T.$$

With this notation it should be clear that the inner product between the ith row of A_k and w_{j+1} is given by

$$\begin{aligned}
[A_k]_{i,:}\, w_{j+1} &= \varpi_j(\cos(j\tau_{i-k}) + \cos(j\tau_{i+k})) \\
&= \varpi_j(\cos(j(i-k-\tfrac{1}{2})h) + \cos(j(i+k-\tfrac{1}{2}))) \\
&= \varpi_j(\cos(j(i-\tfrac{1}{2})h - jkh) + \cos(j(i-\tfrac{1}{2})h + jkh)) \\
&= \varpi_j 2\,\cos(jkh)\,\cos(j(i-\tfrac{1}{2})h) = 2\,\cos(jkh)\,[w_{j+1}]_i.
\end{aligned}$$

Since this holds for all i, j, and k, we have thus proved that all the vectors w_{j+1} are eigenvectors of all the basis matrices. Moreover, since any STH matrix A is a linear combination of the basis matrices, it follows that all the discrete cosine vectors w_{j+1} are also eigenvectors of A.

The eigenvalues d_i of A can be computed via this result in $O(n^2)$ flops, but they can be computed in $O(n \log n)$ flops as follows. Let $e_1 = (1, 0, \ldots, 0)^T$ and $D = \mathrm{diag}(d_1, \ldots, d_n)$. Then

$$A(:,1) = A\,e_1 = W_n\,D\,W_n^T e_1 \quad \Leftrightarrow \quad W_n^T A(:,1) = D\,W_n^T e_1,$$

and thus the ith eigenvalue is

$$d_i = [\mathrm{dct}(A(:,1))]_i \,/\, [\mathrm{dct}(e_1)]_i, \qquad i = 1, \ldots, n.$$

Appendix C

Early Work on "Tikhonov Regularization"

This short appendix summarizes some of the early work on the method which is commonly referred to as "Tikhonov regularization" today. As we shall see, this name is not quite fair to several authors of these early papers, especially Phillips and Twomey (who are occasionally mentioned along with Tikhonov).

Perhaps the first author to describe a scheme that is equivalent to Tikhonov regularization was James Riley, who, in his paper [60] from 1955, considered ill-conditioned systems $A\,x = b$ with a symmetric positive (semi)definite coefficient matrix. He proposed to solve instead the system $(A+\alpha\,I)\,x = b$, where α is a small positive constant. In the same paper, Riley also suggested an iterative scheme which is now known as iterated Tikhonov regularization; cf. Section 5.1.5 in [32].

The first paper devoted specifically to inverse problems was published by D. L. Phillips [58] in 1962. In this paper A is a square matrix obtained from discretization of a first-kind Fredholm integral equation by means of a quadrature rule, and L is the tridiagonal matrix L_2^z. Phillips arrived at the formulation in (8.2), but without matrix notation, and then proposed to compute the regularized solution as $x_\lambda = (A + \lambda^2 A^{-T} L^T L)^{-1} b$, using our notation. It is not clear whether Phillips computed A^{-1} explicitly, and the paper does not recognize (8.2) as a least squares problem.

In his 1963 paper [69], S. Twomey reformulated Phillips's expression for x_λ via the least squares formulation and obtained the well-known "regularized normal equations" expression for the solution:

$$x_{L,\lambda} = \left(A^T A + \lambda^2 L^T L\right)^{-1} A^T b,$$

still with $L = L_2^z$. He also proposed to include an a priori estimate x_0, but only in connection with the choice $L = I$, leading to the formula $x_\lambda = \left(A^T A + \lambda^2 I\right)^{-1} \left(A^T b + \lambda^2 x_0\right)$.

A. N. Tikhonov's paper [65] from 1963 is formulated in a much more general setting: he considered the problem $\mathcal{K} f = g$, where f and g are functions and $\mathcal{K}$ is an integral operator. Tikhonov proposed the formulation (8.1) with the particular smoothing norm

$$S(f) = \int_a^b \left(v(s)\, f(s)^2 + w(s)\, f'(s)^2\right) ds,$$

where v and w are positive weight functions. Turning to computations, Tikhonov used the midpoint quadrature rule to arrive at the problem

$$\min_x \left\{ \|A\,x - b\|_2^2 + \lambda^2 \left(\|D_v^{1/2}\,x\|_2^2 + \|D_w^{1/2}\,L\,x\|_2^2 \right) \right\},$$

in which D_v and D_w are diagonal weight matrices corresponding to v and w, and $L = \text{bidiag}(-1, 1)$. Via the "regularized normal equations" he then derived the expression $x_{L,\lambda} = \left(A^T A + \lambda^2 (D_v + L^T D_w\,L)\right)^{-1} A^T b$.

In 1965 Gene H. Golub [20] was the first to propose a modern approach to solving (8.2) via the least squares formulation (8.3) and QR factorization of the associated coefficient matrix. Golub proposed this approach in connection with Riley's iterative scheme, which includes the computation of x_λ as the first step. G. Ribiere [59] also proposed the QR-based approach to computing x_λ in 1967.

Stochastic perspectives on Tikhonov regularization go back to the paper [16] by M. Foster from 1961, which predates the Phillips and Tikhonov papers. This paper introduces the "regularized normal equations" formulation of general-form Tikhonov regularization, including the prewhitening from Section 4.8.2. In this setting, the matrix $\lambda^2\,L^T L$ (in our notation) represents the inverse of the covariance matrix for the solution.

In 1966, V. F. Turchin [68] introduced a statistical ensemble of smooth functions that satisfy the Fredholm integral equation (2.2) within data errors. Joel Franklin's paper [17] from 1970 revisits the formulation in general form in the setting of random processes over Hilbert spaces.

In much of the statistics literature, Tikhonov regularization is known as ridge regression, which seems to date back to the papers [44], [45] from 1970 by Hoerl and Kennard. The same year, Marquardt [52] used this setting as the basis for an analysis of his iterative algorithm from 1963 for solving nonlinear least squares problems [53], and which incorporates standard-form Tikhonov regularization in each step.

Finally, it should be mentioned that Franklin [18] in 1978, in connection with symmetric positive (semi)definite matrices A and B, proposed the regularized solution $\bar{x}_\lambda = (A + \alpha\,B)^{-1} b$, where α is a positive scalar, which nicely connects back to Riley's 1955 paper.

Bibliography

[1] J. M. Bardsley and J. G. Nagy, *Covariance-preconditioned iterative methods for nonnegatively constrained astronomical imaging*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1184–1197.

[2] R. S. Anderssen, R. P. Brent, D. J. Daley, and P. A. P. Moran, *Concerning* $\int_0^1 \cdots \int_0^t (x_1^2 + \cdots + x_k^2)^{1/2} dx_1 \cdots dx_k$ *and a Taylor series method*, SIAM J. Appl. Math., 30 (1976), pp. 22–30.

[3] R. C. Aster, B. Borchers, and C. H. Thurber, *Parameter Estimation and Inverse Problems*, Elsevier Academic Press, Amsterdam, 2005.

[4] C. T. H. Baker, *The Numerical Treatment of Integral Equations*, Clarendon Press, Oxford, UK, 1977.

[5] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[6] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM Classics in Applied Mathematics 14, Philadephia, 1995.

[7] W. L. Briggs and V. E. Henson, *The DFT: An Owners' Manual for the Discrete Fourier Transform*, SIAM, Philadelphia, 1995.

[8] D. Calvetti and E. Somersalo, *Introduction to Bayesian Scientific Computing: Ten Lectures on Subjective Computing*, Springer, New York, 2007.

[9] J. L. Castellanos, S. Gómez, and V. Guerra, *The triangle method for finding the corner of the L-curve*, Appl. Numer. Math., 43 (2002), pp. 359–373.

[10] J. Chung, J. Nagy, and D. P. O'Leary, *A weighted GCV Method for Lanczos hybrid regularization*, Electron. Trans. Numer. Anal., 28 (2008), pp. 149–167.

[11] D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory*, 2nd ed., Springer, Berlin, 1998.

[12] J. Cullum, *The effective choice of the smoothing norm in regularization*, Math. Comp., 33 (1979), pp. 149–170.

[13] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen, *Algorithms and software for total variation image reconstruction via first-order methods*, Numer. Algorithms, 53 (2010), pp. 67–92. The software is available from Netlib in the directory `numeralgo/na28`.

[14] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer, Dordrecht, The Netherlands, 1996.

[15] M. Fedi, P. C. Hansen, and V. Paoletti, *Tutorial: Analysis of depth resolution in potential-field inversion*, Geophysics, 70 (2005), pp. A1–A11.

[16] M. Foster, *An application of the Wiener-Kolmogorov smoothing theory to matrix inversion*, J. Soc. Indust. Appl. Math., 9 (1961), pp. 387–392.

[17] J. N. Franklin, *Well-posed stochastic extensions of ill-posed linear problems*, J. Math. Anal. Appl., 31 (1970), pp. 682–716.

[18] J. N. Franklin, *Minimum principles for ill-posed problems*, SIAM J. Math. Anal., 9 (1978), pp. 638–650.

[19] R. R. Goldberg, *Methods of Real Analysis*, 2nd ed., Wiley, New York, 1976.

[20] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[21] R. M. Gray, *Toeplitz and circulant matrices: A review*, Foundations and Trends in Communications and Information Theory, 2 (2006), pp. 155–239.

[22] R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms*, Wiley, New York, 1969.

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

[24] C. W. Groetsch, *Inverse Problems: Activities for Undergraduates*, Mathematical Associated of America, Washington, DC, 1999.

[25] P. C. Hansen, *Computation of the singular value expansion*, Computing, 40 (1988), pp. 185–199.

[26] P. C. Hansen, *Perturbation bounds for discrete Tikhonov regularization*, Inverse Problems, 5 (1989), pp. L41–L44.

[27] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, 30 (1990), pp. 658–672.

[28] P. C. Hansen, *Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 503–518.

[29] P. C. Hansen, *Numerical tools for analysis and solution of Fredholm integral equations of the first kind*, Inverse Problems, 8 (1992), pp. 849–872.

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

[31] P. C. Hansen, *Regularization Tools: A MATLAB package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35.

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

[33] P. C. Hansen, *Regularization Tools Version 4.0 for MATLAB 7.3*, Numer. Algorithms, 46 (2007), pp. 189–194.

[34] P. C. Hansen, *The L-curve and its use in the numerical treatment of inverse problems*, invited chapter in Computational Inverse Problems in Electrocardiology, P. Johnston, ed., WIT Press, Southampton, 2001, pp. 119–142.

[35] P. C. Hansen, *Deconvolution and regularization with Toeplitz matrices*, Numer. Algorithms, 29 (2002), pp. 323–378.

[36] P. C. Hansen and T. K. Jensen, *Smoothing-norm preconditioning for regularizing minimum-residual methods*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 1–14.

[37] P. C. Hansen, T. K. Jensen, and G. Rodriguez, *An adaptive pruning algorithm for the discrete L-curve criterion*, J. Comput. Appl. Math., 198 (2007), pp. 483–492.

[38] P. C. Hansen, M. Kilmer, and R. H. Kjeldsen, *Exploiting residual information in the parameter choice for discrete ill-posed problems*, BIT, 46 (2006), pp. 41–59.

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

[40] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[41] P. C. Hansen, T. Sekii, and H. Shibahashi, *The modified truncated SVD method for regularization in general form*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1142–1150.

[42] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[43] I. Hnětynková, M. Plešinger, and Z. Strakoš, *The regularizing effect of the Golub–Kahan iterative bidiagonalization and revealing the noise level in the data*, BIT, 49 (2009), pp. 669–696.

[44] A. E. Hoerl and R. W. Kennard, *Ridge regression. Biased estimation for nonorthogonal problems*, Technometrics, 12 (1970), pp. 55–67.

[45] A. E. Hoerl and R. W. Kennard, *Ridge regression. Applications to nonorthogonal problems*, Technometrics, 12 (1970), pp. 69–82.

[46] T. K. Jensen and P. C. Hansen, *Iterative regularization with minimum-residual methods*, BIT, 47 (2007), pp. 103–120.

[47] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions, Vol. 1*, 2nd ed., Wiley, New York, 1994.

[48] A. C. Kak and Malcolm Slaney, *Principles of Computerized Tomographic Imaging*, SIAM, Philadelphia, 2001.

[49] M. E. Kilmer and D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.

[50] R. Kress, *Linear Integral Equations*, 2nd ed., Springer, Heidelberg, 1999.

[51] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974; reprinted by SIAM, Philadelphia, 1995.

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

[53] D. W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 431–441.

[54] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

[55] P. Midy and I. Brissaud, *Application of a new algorithm to depth profiling by PIXE*, Nucl. Instr. and Meth. B, 103 (1995), pp. 489–493.

[56] F. Natterer and F. Wübbeling, *Mathematical Methods in Image Reconstruction*, SIAM, Philadelphia, 2001.

[57] A. Neumaier, *Solving ill-conditioned and singular linear systems: A tutorial on regularization*, SIAM Rev., 40 (1998), pp. 636–666.

[58] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, J. Assoc. Comput. Mach., 9 (1962), pp. 84–97.

[59] G. Ribiere, *Regularisation d'operateurs*, R.I.R.O., 1 (1967), pp. 57–79.

[60] J. D. Riley, *Solving systems of linear equations with a positive definite, symmetric, but possibly ill-conditioned matrix*, Math. Tables Aids Comput., 9 (1955), pp. 96–101.

[61] B. W. Rust, *Truncating the Singular Value Decomposition for Ill-Posed Problems*, Report NISTIR 6131, Mathematical and Computational Sciences Division, NIST, 1998.

[62] B. W. Rust and D. P. O'Leary, *Residual periodograms for choosing regularization parameters for ill-posed problems*, Inverse Problems, 24 (2008), 034005.

[63] G. W. Stewart, *On the early history of the singular value decomposition*, SIAM Rev., 35 (1993), pp. 551–566.

[64] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, Philadelphia, 2004.

[65] A. N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Soviet Math. Dokl., 4 (1963), pp. 1035–1038; English translation of Dokl. Akad. Nauk. SSSR, 151 (1963), pp. 501–504.

[66] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Winston, Washington, DC, 1977.

[67] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[68] V. F. Turchin, *Solution of the Fredholm equation of the first kind in a statistical ensemble of smooth functions*, USSR Comput. Math. and Math. Phys., 7 (1966), pp. 79–96.

[69] S. Twomey, *On the numerical solution of Fredholm integral equations of the first kind by inversion of the linear system produced by quadrature*, J. Assoc. Comput. Mach., 10 (1963), pp. 97–101.

[70] F. Ursell, *Introduction to the theory of linear integral equations*, in Numerical Solution of Integral Equations, L. M. Delves and J. Walsh, eds., Clarendon Press, Oxford, 1974, pp. 2–11.

[71] A. van der Sluis and H. A. van der Vorst, *SIRT- and CG-type methods for the iterative solution of sparse linear least-squares problems*, Linear Algebra Appl., 130 (1990), pp. 257–302.

[72] J. M. Varah, *Pitfalls in the numerical solution of linear ill-posed problems*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 164–176.

[73] J. M. Varah, *The prolate matrix*, Linear Algebra Appl., 187 (1993), pp. 269–278.

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

[75] G. Wahba, *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics 59, SIAM, Philadelphia, 1990.

[76] G. M. Wing and J. D. Zahrt, *A Primer on Integral Equations of the First Kind*, SIAM, Philadelphia, 1991.

Index