Numerical Aspects of Deconvolution

Per Christian Hansen
Department of Mathematical Modelling
Technical University of Denmark
http://www.imm.dtu.dk/~pch

July 1999

Abstract

This lecture note was written for use in the course 04310 Scientific Computing at the Technical University of Denmark, and it also supplements the material in the author's book [11] about regularization methods. It continues to evolve as the author obtains more insight into world of structured matrices.

By deconvolution we mean the solution of a first-kind integral equation with a convolution-type kernel, i.e., a kernel that depends only on the difference between the two independent variables. The corresponding computational problem takes the form of structured matrix problem. The aim of the lecture note is to present numerical methods for the practical treatment of these discretized deconvolution problems, with emphasis on methods that take the special structure of the matrix into account. Wherever possible, analogies to classical DFT-based deconvolution problems are drawn.

We start with a brief introduction to deconvolution problems and their discretization in §1. Then we discuss general properties of first-kind Fredholm integral equations in §2 and give an introduction to numerical regularization methods in §§3–4, with no emphasis on matrix structure.

The core algorithmic material for structured matrices is presented in §5 and §6 that treat 1-D and 2-D problems, respectively. In §5 we introduce Toeplitz and circulant matrices, and show how Toeplitz matrix-vector computation is performed by means of FFT, being useful in iterative methods. We also present some direct methods for regularization with Toeplitz matrices. In §6 we introduce the Kronecker product and show how it is used in the discretization and solution of 2-D problems whose variables separate.

Contents

1. Introduction to Deconvolution Problems

The main purpose of this lecture note is to present modern *computational methods* for treating linear deconvolution problems, along with some of the underlying theory. The main focus is on large-scale problems, and we shall illustrate how mathematics helps to arrive at efficient numerical algorithms.

The term "deconvolution" seems to have slightly different meanings in different communities – to some, it is strongly connected to the operation of convolution in digital signal processing, while to others it denotes a broader class of problems. In the remainder of Chapter 1, we shall briefly describe both meaning of the term.

Before we present the algorithms, we make a detour in Chapter 2 into the world of inverse problems and, in particular, Fredholm integral equations of the first kind. The reason for this is that deconvolution problems are special cases of these integral equations. Extensive amounts of both theory and algorithms have been developed for these general problems, and our detour will provide us with important insight and therefore a stronger background for solving deconvolution problems numerically.

Having thus set the stage, we turn to the numerical methods in Chapters 3–6. We start in Chapter 3 with a general discussion of the discretization and numerical treatment of first-kind integral equations, and we introduce the singular value decomposition which is perhaps the most powerful "tool" for understanding the difficulties inherent in these integral equations. Next, in Chapter 4 we discuss general regularization algorithms for computing stabilized solutions to the discretized linear systems, with no emphasis on matrix structure.

In the remaining chapters we turn to numerical methods that exploit matrix structure. In Chapter 5 we discuss various direct and iterative methods for one-dimensional problems, and show how we can compute the numerical solutions accurately and efficiently. Finally, in Chapter 6 we turn to two-dimensional problems, and here we focus on problems in which the variables separate, allowing us to also develop very efficient numerical schemes for these problems.

Some acquaintance with numerical linear algebra is necessary; the most complete reference is the book [8] that discusses all state-of-the-art techniques, classical as well

as modern.

The presentation lends itself strongly to Matlab, and we shall occasionally use Matlab notation in our expressions, such as ".*" for element-wise multiplication of two vectors or matrices. We shall also make reference to the Matlab package REGULARIZATION TOOLS [10] from time to time. More details about the theory that is briefly presented in this note can be found in the "deconvolution primer" [19], while a recent survey of numerical methods for general inverse problems is given in the monograph [11].

1.1. Deconvolution in Digital Signal Processing

Given two discrete (digital) signals f and h, both of infinite length, the *convolution* of f and h is defined as a new infinite-length signal g as follows

$$g_i = \sum_{j=-\infty}^{\infty} f_j \, h_{i-j}, \qquad i \in \mathbb{Z}.$$

Note that one (or both) signals may have finite length, in which case the above relation still holds when the remaining elements are interpreted as zeros. The deconvolution is used to represent many useful operations in digital signal processing. For example, the output of a FIR filter with n nonzero filter coefficients $h_0, \ldots, h_{n-1}$ is given by

$$g_i = \sum_{j=0}^{n-1} f_j \, h_{i-j}, \qquad i \in \mathbb{Z}.$$

Throughout, let $\hat{\imath}$ denote the imaginary unit satisfying $\hat{\imath}^2 = -1$. If $\widehat{g}(\omega) = \sum_{j=-\infty}^{\infty} g_j \, e^{-2\pi \hat{\imath} j \omega}$ is the Fourier transform of g, then the following important relation holds for the Fourier transforms of f, g, and h:

$$\widehat{g}(\omega) = \widehat{f}(\omega) \, \widehat{h}(\omega), \qquad \omega \in \mathbb{R}.$$

If the two signals f and h are both *periodic* with period N, and represented by the sequences $f_0, f_1, \ldots, f_{N-1}$ and $h_0, h_1, \ldots, h_{N-1}$, then the convolution of f and h is another periodic signal g with period N, and the elements of g are defined as

$$g_i = \sum_{j=0}^{N-1} f_j \, h_{i-j}, \qquad i = 0, 1, \ldots, N-1,$$

where the subscript of h is to be taken modulo N. The discrete Fourier transform (DFT) of g is defined as the sequence

$$G_k \equiv \frac{1}{N} \sum_{j=0}^{N-1} g_j \, e^{-i 2\pi jk/N}, \qquad k = 0, 1, \ldots, N-1,$$

and it is convenient to denote the N-vector consisting of this sequence by $\mathrm{DFT}(g)$. Then, using Matlab notation, the following relation holds for the DFT of the convolution of f and g

$$\mathrm{DFT}(g) = \mathrm{DFT}(f) \,.*\, \mathrm{DFT}(h), \tag{1.1}$$

i.e., $\mathrm{DFT}(g)$ equals the element-wise product of the two vectors $\mathrm{DFT}(f)$ and $\mathrm{DFT}(h)$. Consequently, $g = \mathrm{IDFT}(\mathrm{DFT}(f) \,.*\, \mathrm{DFT}(h))$, where IDFT denotes the inverse DFT. Note that f and h can be interchanged without changing g. We remind that the DFT and the IDFT of a signal can always be computed efficiently by means of the fast Fourier transform (FFT) in $\mathcal{O}(N \log_2 N)$ operations. A very thorough discussion of the DFT (and the FFT) is presented in [4].

Deconvolution is then defined as the process of computing the signal f given the other two signals g and h. For periodic discrete signals, Eq. (1.1) leads to a simple expression for computing f, considered as an N-vector:

$$f = \mathrm{IDFT}\left(\mathrm{DFT}(g) \,./\, \mathrm{DFT}(h)\right). \tag{1.2}$$

Again, the FFT can be used to perform these computations efficiently. For general discrete signals, the Fourier transform of the deconvolved signal f is formally given by $f(\omega) = g(\omega)/h(\omega)$, but there is no similar simple computational scheme for computing f – although the FFT and Eq. (1.2) are often used (and occasionally misused) for computing approximations to f efficiently; cf. Section 5.5.

1.2. General Deconvolution Problems

Outside the field of signal processing, there are many computational problems which resemble the classical deconvolution problem in signal processing, and these problems are also often called deconvolution problems. Given two functions f and h, the *general convolution* operation takes the generic form

$$g(s) = \int_0^1 h(s-t)\, f(t)\, dt, \qquad 0 \le s \le 1, \tag{1.3}$$

where we assume that both integration intervals have been transformed into the interval $[0, 1]$. Certain convolution problems involve the interval $[0, \infty)$, but in order

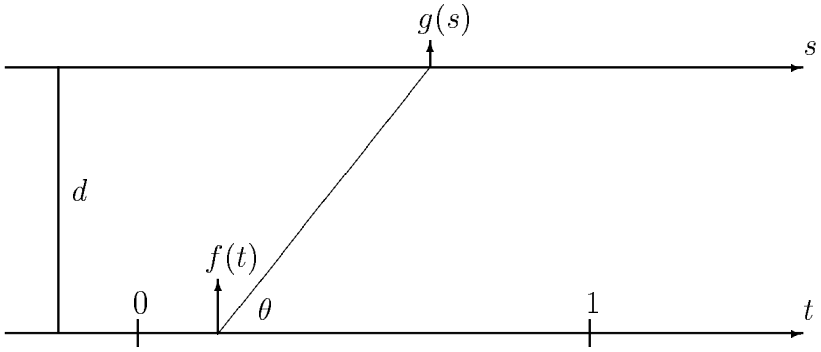

Figure 1.1: The geometry of the geomagnetic prospecting model problem.

to focus our presentation on the principles of deconvolution we omit these problems here. Notice that by means of a change of variables, we can also write $g(s) = \int_0^1 f(s-\tau)\,h(\tau)\,d\tau$, showing that the two forms are equivalent. Hence, from a mathematical point of view, the two functions f and h play similar roles – although in applications, they may have very different interpretations.

The general problem of deconvolution is now to determine either f or h, given the other two quantities. Due to the above-mentioned equivalence between f and h, we shall always assume that h is the known function while we want to compute f. Usually, the function h comes from a mathematical model of the underlying problem, while g is available as measured data for a discrete set of s-values, i.e., we are given "noisy samples" of g at certain discrete abscissas $s_1,\ s_2,\ \ldots,\ s_m$:

$$\widetilde{g}_i = g(s_i) + e_i, \qquad i = 1, \ldots, m.$$

Here, e_i denotes the measurement noise associated with the ith data point. The noise usually comes from some statistical distribution, which may be known or unknown.

One example of deconvolution is the following model problem in *geomagnetic prospecting*. Assume that a horizontal iron ore deposit lies at depth d below the surface – cf. Fig. 1.1 for the geometry and the location of the s and t axes. From measurements of the vertical component of the magnetic field, denoted $g(s)$, at the surface, we want to compute the vertical component of the field, denoted $f(t)$, right at the ore. The contribution to g from an infinitesimal part dt of the ore at t is given by

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

where the angle θ is shown in Fig. 1.1, and the distance is given by $r = \sqrt{d^2 + (s-t)^2}$.

Using that $\sin\theta = d/r$, we get

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

The total value of $g(s)$ for any s is therefore

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

Thus, we arrive at a deconvolution problem for computing the desired quantity f with kernel given by $h(s-t) = d\,(d^2+(s-t)^2)^{-3/2}$.

Deconvolution problems also arise, e.g., in statistics, and we shall briefly describe an example from astrometry. Assume that we are given a measured statistical distribution g of stellar parallaxes (the parallax is a small angle from which the distance to the star can be computed). This distribution is perturbed due to various measurement errors, and it can be shown that if f denotes the true distribution of parallaxes, then f and g are related by the equation

$$\int_0^1 \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{s-t}{\sigma}\right)^2\right) f(t)\,dt = g(s),$$

where σ is a parameter that characterizes the accuracy of the particular set of measurements. The above equation is obviously a deconvolution problem with kernel $h(s-t) = \left(\sigma\sqrt{2\pi}\right)^{-1} \exp\left(-\frac{1}{2}\left(\frac{s-t}{\sigma}\right)^2\right)$ for computing the true distribution f of parallaxes. This problem is implemented as the function parallax in REGULARIZATION TOOLS [10].

The inverse heat equation is yet another example of a deconvolution problem. Assume that we wish to determine the temperature f, as a function of time t, on one side of a wall (which is inaccessible) given measurements g of the temperature on the other (accessible) side of the wall. This leads to a deconvolution problem of the generic form

$$\int_0^s h(s-t)\,f(t)\,dt = g(s) \tag{1.4}$$

with kernel h given by

$$h(s-t) = \frac{(s-t)^{-3/2}}{2\,\kappa\,\sqrt{\pi}}\,\exp\left(-\frac{1}{4\,\kappa^2\,(s-t)}\right),$$

where the parameter κ describes the heat conduction properties of the wall. This model problem is implemented in REGULARIZATION TOOLS as the function heat.

2. First-Kind Fredholm Integral Equations

At this stage, we begin our detour into the world of inverse problems and Fredholm integral equations of the first kind. Inverse problems can often be characterized as problems in which we wish to compute certain properties of the interior of a domain, given measurements made from the outside along with a mathematical model of the relation between the interior and the measured data. Seismology is a classical example of an inverse problem, where the goal is to map the various layers of the earth, given measurements at the earth's surface of seismic waves penetrating the layers and reflected by the layers. Another classical example is computerized X-ray tomography, where we wish to compute images of, say, the human brain, given measurements of the damping of the X-rays through the brain.

It should be emphasized here that inverse problems do not always involve a strictly interior region. It is perhaps more precise to say that from measured data one wants to infer about certain hidden data, unaccessible to be measured directly. The two examples mentioned above are clearly in this class of problems. Another example is image deblurring: here, given a recorded (e.g., digitized) blurred image and a mathematical model for the blurring process, the goal is to reconstruct the original sharp image as accurately as possible. For several years, before it was finally repaired, the Hubble Space Telescope provided nothing but such blurred images.

All the above-mentioned inverse problems, as well as the deconvolution problems from the previous chapter, can be formulated as Fredholm integral equations of the first kind. The generic form of a first-kind *Fredholm integral equation* looks as follows

$$\int_0^1 K(s,t)\, f(t)\, dt = g(s), \qquad 0 \le s \le 1, \tag{2.1}$$

where the function K, called the *kernel*, is a known function of the two variables s and t, and the right-hand side g is also known – or at least measured for discrete values of s – while f is the unknown function that we wish to compute. Note that the deconvolution problems from the previous chapter are merely special cases of (2.1) in which the kernel $K(s,t) = h(s-t)$ depends solely on the difference $s-t$ between the two variables.

We point out that the integration interval in Eq. (1.4) is from 0 to s; such an integral equation is called a Volterra integral equation of the first kind. It can be considered as a special Fredholm integral equation whose kernel is zero for $t > s$ and it shares the same difficulties as the Fredholm equation.

2.1. Smoothing and Inversion

As already mentioned, the advantage of working with the general formulation (2.1) is that the underlying theory of first-kind Fredholm integral equations is so well developed. For example, it is well understood that the integration of f with K is a smoothing operation that tends to dampen high-frequency components in f, such that the function g is a smoother function than f. Consequently, the inverse process, namely, that of computing f from g, can be expected to amplify the high-frequency components and it is thus a "de-smoothing process," so to speak.

The above statement is quantified by the following example. If we choose the function f to be given by

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

such that the corresponding right-hand side g is given by

$$g(s) = \int_0^1 K(s,t)\, \sin(2\pi pt)\, dt, \qquad p = 1, 2, \ldots,$$

then the Riemann-Lebesgue lemma[1] states that

$$g \to 0 \qquad \text{as} \qquad p \to \infty.$$

In words, the higher the frequency of f, the more g is damped – and this is true independently of the kernel K; see Fig. 2.1 for a numerical example. Consequently, the reverse process, i.e., that of computing f from g, will amplify the high frequencies.

If all data were unperturbed, and all computations could be done in infinite precision, then the inversion process in getting from g to f would be perfectly possible. Unfortunately, this situation never arises in practice: measured data are always contaminated by errors, and numerical computations always involve small but non-negligible rounding errors. Due to the above result, small perturbations of the high-frequency components in g are transformed into large perturbations of f, and the higher the frequency the larger the perturbation of f. Hence, the

[1] The Riemann-Lebesgue lemma can be formulated as follows: if the function ψ has limited total fluctuation in the interval $(0,1)$, then, as $\lambda \to \infty$, $\int_0^1 \psi(\theta)\, \sin(\lambda\theta)\, d\theta$ is $\mathcal{O}(\lambda^{-1})$.

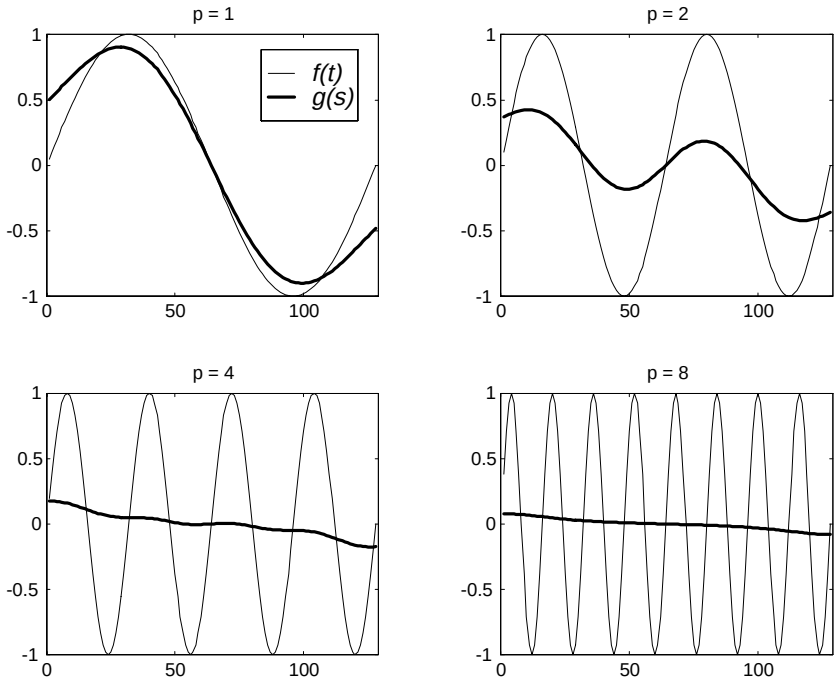

Figure 2.1: Illustration of the Riemann-Lebesgue lemma for the geomagnetic prospecting problem, with $f(t) = \sin(2\pi\, pt)$, $g(s) = \int_0^1 h(s-t)\, f(t)\, dt$, and $p = 1$, 2, 4, and 8.

unavoidable data and rounding errors make the practical inversion process a very unstable process indeed, and in practise it is impossible to compute or estimate f by means of a direct inversion. Some kind of stabilization technique is needed in order to recover a reasonably accurate approximation to the desired solution f.

2.2. An Example from Signal Processing

We can illustrate the above-mentioned difficulties by means of the deconvolution formula (1.2) for periodic digital signals. If h corresponds to a smoothing operation, then the spectral components of h, i.e., the components of the vector $\mathrm{DFT}(h)$, are such that the higher the frequency, the smaller the corresponding element. Hence, the same is true for the elements of the DFT of the exact right-hand side. Assume now that we are given the vector $\widetilde{g} = g + e$ of measured values, consisting of the exact data g contaminated by additive white noise e with elements from a normal distribution with zero mean and standard deviation σ_{noise}. Then the DFT of $\widetilde{g}$ is given by

$$\mathrm{DFT}(\widetilde{g}) = \mathrm{DFT}(g) + w,$$

where all the elements in the vector $w = \mathrm{DFT}(e)$ have the same probability. Thus, the expression for the DFT of the computed solution $\widetilde{f}$ becomes

$$\begin{aligned} \mathrm{DFT}(\widetilde{f}) &= \mathrm{DFT}(\widetilde{g})\,./\mathrm{DFT}(h) \\ &= \mathrm{DFT}(g)\,./\mathrm{DFT}(h) + w\,./\mathrm{DFT}(h) \\ &= \mathrm{DFT}(f) + w\,./\mathrm{DFT}(h), \end{aligned}$$

showing that the high-frequency components of $\widetilde{f}$ are perturbed the most, due to the division of the elements in w by the small elements in $\mathrm{DFT}(h)$.

We illustrate the above with a numerical example involving a low-pass filter with filter coefficients 0.5, 1, 1, 1, and 0.5, applied to a short sequence of 512 samples of a speech signal. Fig. 2.2 shows various power spectra of the signals involved in this example. The noise e is generated in Matlab as `e = 0.1*randn(512,1)`, and the power spectrum of e is flat, as we expect from white noise. Notice how the high-frequency components of the deconvolved signal are perturbed wildly, especially around the zeros of the low-pass filter! Clearly, "naive" inversion by means of Eq. (1.2) is useless.

2.3. The Singular Value Expansion

A deeper understanding of the behavior of first-kind Fredholm integral equations can be achieved by means of the *singular value expansion* (SVE) of the kernel K, as long

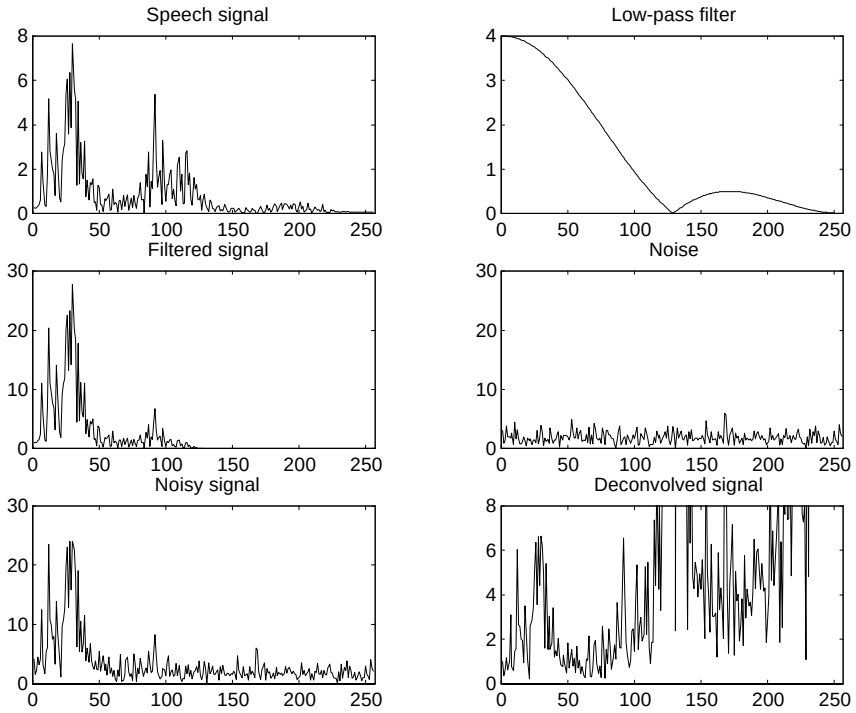

Figure 2.2: Power spectra of the various signals involved in the low-pass filtering example.

as K is square integrable, i.e., as long as the quantity $\|K\|_2^2 = \int_0^1 \int_0^1 K(s,t)^2 \, ds \, dt$ is bounded. Such kernels can always be written as the following infinite sum

$$K(s,t) = \sum_{i=1}^{\infty} \mu_i \, u_i(s) \, v_i(t), \tag{2.2}$$

where the functions u_i and v_i are termed the singular functions of K, and the numbers μ_i are called the singular values of K. The singular functions are orthonormal with respect to the usual inner product, i.e.,

$$(u_i, u_j) = (v_i, v_j) = \delta_{ij}, \qquad \text{where} \qquad (\phi, \psi) = \int_0^1 \phi(t) \, \psi(t) \, dt,$$

and the singular values are ordered in non-increasing order such that

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

The singular values satisfy the relation $\sum_{i=1}^{\infty} \mu_i^2 = \|K\|_2^2$, showing that the μ_i must decay faster than $i^{-1/2}$.

Perhaps the most important relation between the singular values and functions is the following fundamental relation

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

which shows that any singular function v_i is mapped onto the corresponding singular function u_i, and that the singular value μ_i is the amplification factor of this particular mapping. If this relation, together with Eq. (2.2), is inserted into the integral equation (2.1), then we obtain the equation

$$\sum_{i=1}^{\infty} \mu_i \, (v_i, f) \, u_i(s) = \sum_{i=1}^{\infty} (u_i, g) \, u_i(s), \qquad i = 1, 2, \ldots,$$

which, in turn, leads to the following expression for the solution

$$f(t) = \sum_{i=1}^{\infty} \frac{(u_i, g)}{\mu_i} \, v_i(t). \tag{2.3}$$

The overall behavior of the singular values and functions is by no means "arbitrary," and the difficulties with solving first-kind Fredholm integral equations are due to the following two facts.

- The singular values μ_i decay to zero, at least as fast as $i^{-1/2}$.
- The smaller the μ_i, the more oscillations (or zero-crossings) there will be in the corresponding singular functions u_i and v_i.

The practical implication of the above is that the SVE can be regarded as a kind of spectral expansion in which the coefficients (v_i, f) and $(u_i, g) = \mu_i (v_i, f)$ describe the spectral properties of the solution f and the right-hand side g. Again, we see that the integration with K has a smoothing effect: the higher the spectral components in f, the more they are damped in g due to the multiplication with μ_i. Moreover, Eq. (2.3) shows that the inverse problem, that of computing f from g, indeed has the opposite effect on the oscillations in g, namely, an amplification of g's spectral components (u_i, g) with the factor μ_i^{-1}. This, of course, amplifies the high-frequency components.

The trouble arises when the right-hand side g is contaminated by errors, for then we will most likely divide small but non-negligible coefficients (u_i, g) with very small singular values μ_i, which will lead to a dramatic high-frequent perturbation of the computed solution.

3. Numerical Treatment

Before we turn our attention to practical schemes for stabilizing the inversion process, it is convenient to discuss the numerical treatment of integral equations, as well as some important concepts from numerical linear algebra associated with ill-conditioned matrices.

3.1. Discretization by Quadrature Rules

There is a variety of schemes available for discretization of integral equations, i.e., for turning the integral equation into a system of linear equations that can be solved numerically in order to provide an approximate discrete solution to the integral equation. Here, for pedagogical reasons, we limit or discussion to *quadrature methods* based on well-know quadrature rules. Recall that a quadrature rule for computing an approximation to a definite integral takes the following general from

$$\int_0^1 \varphi(t)\,dt \simeq \sum_{j=1}^{n} w_j\,\varphi(t_j),$$

where $t_1, \ldots, t_n$ are the abscissas for the particular quadrature rule, and $w_1, \ldots, w_n$ are the corresponding weights. For example, for the *midpoint rule*, we have

$$t_j = (j - 0.5)/n, \qquad w_j = 1/n, \qquad j = 1, \ldots, n,$$

while for *Simpson's rule* (where n must be odd) the abscissas are given by

$$t_j = \frac{j-1}{n-1}, \qquad j = 1, \ldots, n$$

and the weights are

$$w_1, w_2, w_3, w_4, w_5, \ldots, w_{n-1}, w_n = c, 4c, 2c, 4c, 2c, \ldots, 4c, c$$

with $c = 1/(3(n-1))$. This formulation also covers more advanced rules such as the Newton-Cotes rules. Using a quadrature rule, we can approximate the integral

in our Fredholm integral equation as follows

$$\int_0^1 K(s,t)\, f(t)\, dt \simeq \sum_{j=1}^{n} w_j\, K(s,t_j)\, \widetilde{f}(t_j) = \psi(s),$$

and we emphasize that the right-hand side $\psi(s)$ in the above expression is still a function of the variable s. Notice that we have replaced f with $\widetilde{f}$ because we introduce approximation errors in the above expression and thus cannot expect to compute f exactly.

In order to arrive at a system of linear equations, we can now use *collocation*, i.e., we require that the function ψ defined above equals the right-hand side g at given points $s_1, \ldots, s_m$:

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

Here, the numbers $g(s_i)$ are usually the measured values of the function g. Note that m need not necessarily be equal to n, but to keep our exposition simple we shall assume so throughout the note, i.e., $m = n$. Inserting the expression for $\psi(s)$ into the collocation scheme, we arrive at the following equations

$$\sum_{j=1}^{n} w_j\, K(s_i,t_j)\, \widetilde{f}(t_j) = g(s_i), \qquad i,j = 1, \ldots, n.$$

When we rewrite these equation in matrix notation, we obtain the system

$$\begin{pmatrix} w_1K(s_1,t_1) & w_2K(s_1,t_2) & \cdots & w_nK(s_1,t_n) \\ w_1K(s_2,t_1) & w_2K(s_2,t_2) & \cdots & w_nK(s_2,t_n) \\ \vdots & \vdots & & \vdots \\ w_1K(s_n,t_1) & w_2K(s_n,t_2) & \cdots & w_nK(sn,t_n) \end{pmatrix} \begin{pmatrix} \widetilde{f}(t_1) \\ \widetilde{f}(t_2) \\ \vdots \\ \widetilde{f}(t_n) \end{pmatrix} = \begin{pmatrix} g(s_1) \\ g(s_2) \\ \vdots \\ g(s_n) \end{pmatrix}$$

or simply $A\,x = b$, where A is an $n \times n$ matrix. The elements of A, b, and x are given by

$$\left.\begin{array}{rcl} a_{ij} & = & w_j K(s_i,t_j) \\ b_i & = & g(s_{i)} \\ x_j & = & \widetilde{f}(t_j) \end{array}\right\} \qquad i,j = 1, \ldots, n.$$

To illustrate the above scheme, we use the midpoint rule to discretize the geomagnetic prospecting problem from Chapter 1, with quadrature and collocations points equidistantly distributed in the interval $[0,1]$ as $s_i = t_i = (i - 0.5)/n$, $i = 1, \ldots, n$. Thus, the matrix elements are given by

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

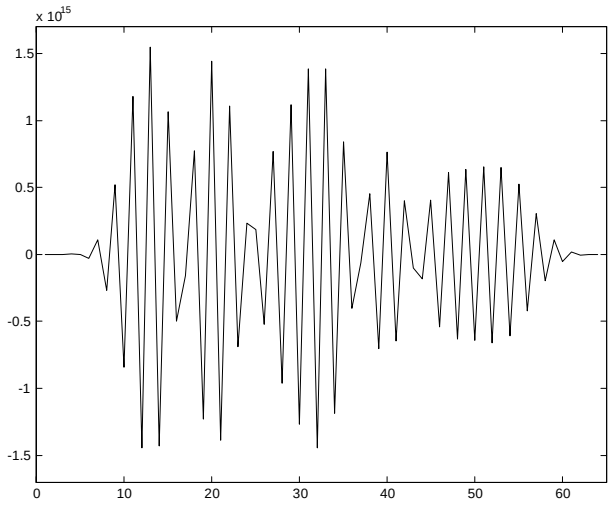

Figure 3.1: "Naive solution" to the geomagnetic prospecting model problem.

As the solution in our model problem, we choose $f(t) = \sin(\pi t) + 0.5\sin(2\pi t)$, and the elements of the exact solution x thus consists of the sampled values of f at the abscissas $t_j = (j - 0.5)/n$ for $j = 1, \ldots, n$. Finally, the right-hand side b is computed as $b = A\,x$. Due to our choice of quadrature and collocation points we obtain a symmetric matrix, and the depth is chosen to be $d = 0.25$.

At this stage, we emphasize that in practise the right-hand side is usually a perturbed version of this b. That is, we solve the system $A\,x = \widetilde{b}$, where $\widetilde{b} = b + e$, and the vector e represents the perturbation of the exact data.

The elements of the computed vector $x_{\text{naive}} = A^{-1}b$ are, in principle, approximations to the desired solution, i.e.,we compute "samples" of the function $\widetilde{f}$ at the abscissas $t_1, \ldots, t_n$ given by

$$\widetilde{f}(t_j) = (x_{\text{naive}})_j, \qquad j = 1, \ldots, n.$$

From a naive point of view one would think that these quantities are supposed to approximate the solution f at the same abscissas. But as we have already emphasized several times, we cannot expect the naively computed solution to a first-kind Fredholm integral equation to be stable with respect to perturbations of then right-hand side, and hence we should not expect the computed values $\widetilde{f}(t_j)$ to be good approximations to the quantities $f(t_j)$ if we solve the system $A\,x = b$ by standard tools in numerical analysis.

Figure 3.1, which shows the "naive" solution x_{naive} to the geomagnetic model problem, illustrates this fact – the solution has nothing in common with the exact

solution, the norm is not even of the correct order of magnitude. We postpone the discussion of how to stabilize the computed solutions to the next chapter, and turn our attention to tools-of-the-trade in numerical linear algebra.

3.2. Linear Equations and Condition Numbers

One of many useful results in linear algebra is concerned with the perturbation of the solution x to a system of linear equations $A\,x = b$ due to a perturbation of the right-hand side b. The following notation is quite standard: let the perturbed right-hand side be $\widetilde{b} = b + e$, where e is the perturbation, and let the corresponding solution be $\widetilde{x}$, which is formally given by $\widetilde{x} = A^{-1}\widetilde{b}$. We now wish to bound the norm of the perturbation $\widetilde{x} - x$.

To arrive at a useful result, we fist note that due to the linearity of the problem, the perturbation can be written as $\widetilde{x} - x = A^{-1}e$, and hence $\|\widetilde{x} - x\| \leq \|A^{-1}\|\,\|e\|$, where $\|\cdot\|$ denotes a vector norm such as the 2-norm. This is an upper bound on the absolute perturbation, but it is always more useful to bound the relative perturbation of the solution. To arrive at such a bound, we use the relation $A\,x = b$ to derive the inequality $\|b\| \leq \|A\|\,\|x\|$, which we rewrite as $1/\|x\| \leq \|A\|/\|b\|$. Combining the two bounds, we thus arrive at the following important *perturbation bound*

$$\frac{\|\widetilde{x} - x\|}{\|x\|} \leq \|A\|\,\|A^{-1}\|\,\frac{\|e\|}{\|b\|}.$$

The factor $\|A\|\,\|A^{-1}\|$ is a magnification factor which controls how much the relative perturbation $\|e\|/\|b\|$ of the right-hand side can be amplified into a relative perturbation $\|\widetilde{x} - x\|/\|x\|$ of the solution. The larger this factor, the more sensitive the solution is to perturbations of the right-hand side.

Notice that this sensitivity is merely controlled by properties of the matrix A, namely, the product of the norm of the matrix times the norm of the inverse of the matrix. This quantity is so important that it has its own name – it is called the *condition number* of A, and we shall denote it by

$$\mathrm{cond}(A) \equiv \|A\|\,\|A^{-1}\|.$$

We shall also make the definition that if A is singular, then the condition number is infinite. As long as $\mathrm{cond}(A)$ is small, we say that the matrix A is well conditioned, because small perturbations are bound to lead to small perturbations of the solution. However, if $\mathrm{cond}(A)$ is large, then even small perturbations in b can lead to large perturbations in x, and we say that A is ill conditioned. The larger the condition number, the more ill conditioned the matrix (and a singular matrix is infinitely ill conditioned).

3.3. The Singular Value Decomposition

One of the most versatile tools in linear algebra is probably the *singular value decomposition* (SVD) of a matrix, and we shall make heavy use of this decomposition throughout the note. The SVD is defined for any rectangular matrix, but we limit our discussion to square $n \times n$ matrices, for which the SVD takes the form

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

The two matrices U and V consist of the left and right singular vectors,

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

and both matrices are orthogonal, i.e., $U^TU = V^TV = I$. This implies that the left and right singular vectors are orthonormal, $u_i^Tu_j = v_i^Tv_j = \delta_{ij}$. The middle matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ is a diagonal matrix whose diagonal elements σ_i are the singular values of A. They are nonnegative and ordered in non-increasing order, i.e.,

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0,$$

and it can be shown that the number of nonzero singular values is equal to the rank of A. The condition number of A as defined above has a simple expression in terms of the SVD if we use the 2-norm, for then

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

If A is singular, i.e., $\mathrm{rank}(A) < n$, then $\sigma_n = 0$ and $\mathrm{cond}(A) = \infty$ in agreement with the above definition. Software for computing the SVD is available in all modern software packages.

If A is a symmetric matrix, $A = A^T$, then the SVD of A is related to the eigenvalue decomposition $A = W\,\Lambda\,W^T$ with $W = (w_1, \ldots, w_n)$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ as follows

$$(u_i, \sigma_i, v_i) = \begin{cases} (w_i, \lambda_i, w_i) & \text{for} \quad \lambda_i \geq 0 \\ (w_i, -\lambda_i, -w_i) & \text{for} \quad \lambda_i < 0 \end{cases} \tag{3.1}$$

and this relation can be used to simplify the computation of the SVD for symmetric matrices.

The singular values and vectors satisfy a number of important relations, and the following is the most important:

$$A\,v_i = \sigma_i\,u_i, \qquad \|A\,v_i\|_2 = \sigma_i\,, \qquad i = 1, \ldots, n. \tag{3.2}$$

This relation leads to a simple expression for the solution in terms of the SVD. First note that we can expand b and x in terms of the left and the right singular vectors u_i and v_i, respectively, as

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

The latter relation, combined with (3.2), leads to the expression

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

and equating the expressions for b and $A\,x$, we arrive at the relations $(u_i^T b) = \sigma_i\, (v_i^T x)$ for $i = 1, \ldots, n$. Hence, the "naive" solution to $A\,x = b$ can be written as

$$x_{\text{naive}} = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i}\, v_i. \tag{3.3}$$

In the next section, we shall use these relations to analyze and explain the difficulties associated with the solution of discretizations of first-kind Fredholm integral equations.

3.4. SVD Analysis and Insight

For general matrices, one cannot say much about the singular values and vectors, except that the singular values decay, by definition. However, for matrices that arise from the discretization of first-kind Fredholm integral equations, a lot can actually be said. We omit the underlying theory here (see, e.g., [11] for details) and state the most important results here.

1. The singular values of A decay gradually, until they level off at a plateau approximately at the machine precision times σ_1 (in infinite precision they would decay to zero).

2. Consequently, the condition number $\text{cond}(A) = \sigma_1/\sigma_n$ is approximately the reciprocal of the machine precision which, for practical purposes, can be considered as infinite.

3. There is no particular gap in the singular value spectrum – typically, the singular values follow a harmonic progression $\sigma_i \simeq i^{-\alpha}$ or a geometric progression $\sigma_i \simeq e^{-\alpha i}$, where α is a positive real constant.

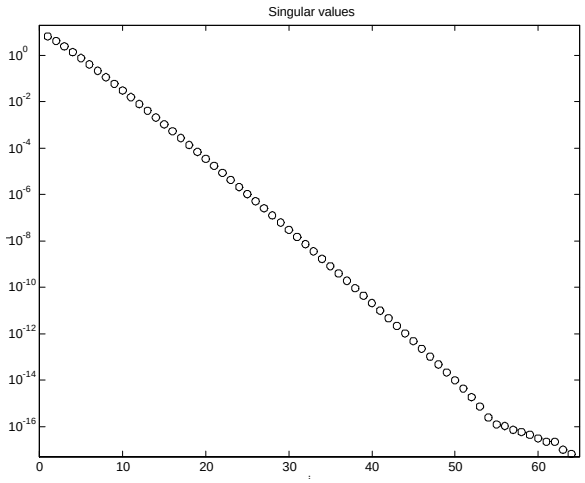

Figure 3.2: The singular values of the geomagnetic prospecting model problem, discretized with the midpoint quadrature rule and $n = 64$.

4. The singular vectors u_i and v_i have an increasing number of sign changes in their elements as i increases, i.e., as the corresponding singular values σ_i decrease. Often, the number of sign changes is precisely $i - 1$.

To illustrate the above results, we computed the SVD of the matrix A from the geomagnetic prospecting problem from Chapter 1, discretized as explained above with $n = 64$. The singular values are shown in Fig. 3.2, and we see that all the singular values indeed decay gradually (almost geometrically) until they level off at approximately 10^{-16}. Not surprisingly, the decay rate depends on the depth d – the larger the d, the faster the decay.

The coefficient matrix A in this example is symmetric, and for symmetric matrices the singular vectors satisfy $u_i = \pm v_i$. Thus, we only need to consider the left singular vectors u_i which are shown in Fig. 3.3. We see that the number of sign changes in the elements of u_i is precisely $i - 1$, supporting our claim that the higher the index i and the smaller the corresponding singular value σ_i, the more high-frequent the singular vectors u_i and v_i.

From the analysis in the previous section, it is clear that in connection with the analysis and solution of the problem $A\,x = b$ we should monitor the behavior of the singular values σ_i, the SVD-components $u_i^T b$ of the right-hand side, and the SVD components $u_i^T b/\sigma_i$ of the computed "naive" solution. Figure 3.4 shows these quantities for the geomagnetic model problem, and we see that the quantities $|u_i^T b|$ decay until they, too, hit the level set by the machine precision. Note that in the

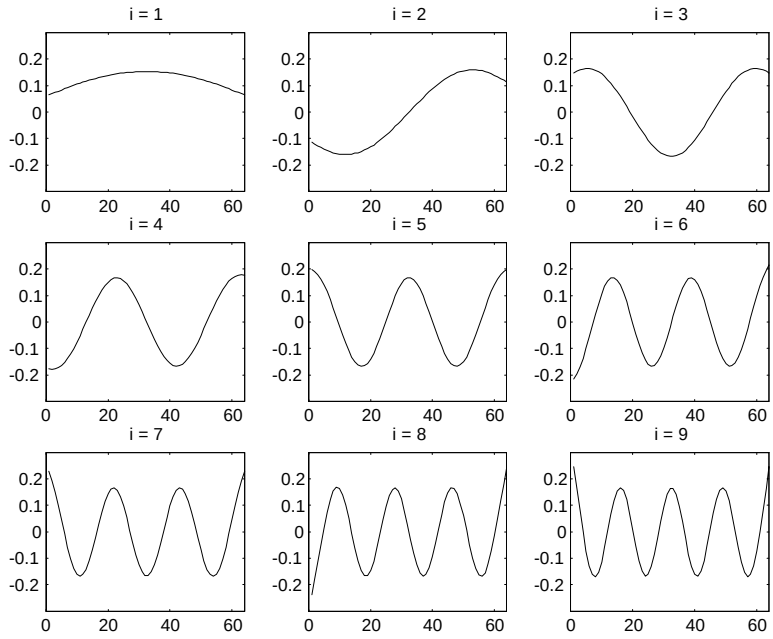

Figure 3.3: The first nine left singular vectors u_i for the geomagnetic prospecting problem.

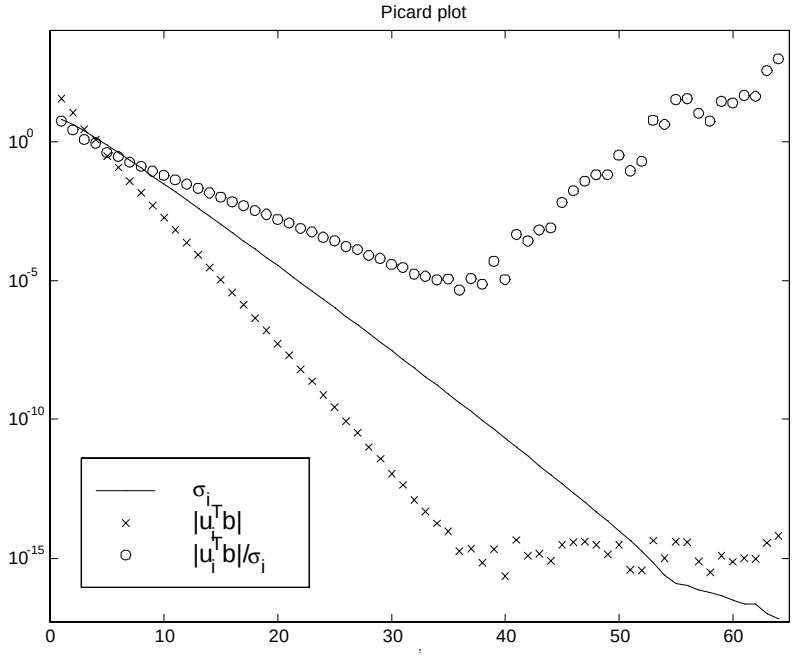

Figure 3.4: Combined plot of the singular values σ_i and the coefficients $|u_i^T b|$ and $|u_i^T b|/\sigma_i$ for the geomagnetic prospecting model problem. Apart from rounding errors, there is no noise in the right-hand side b.

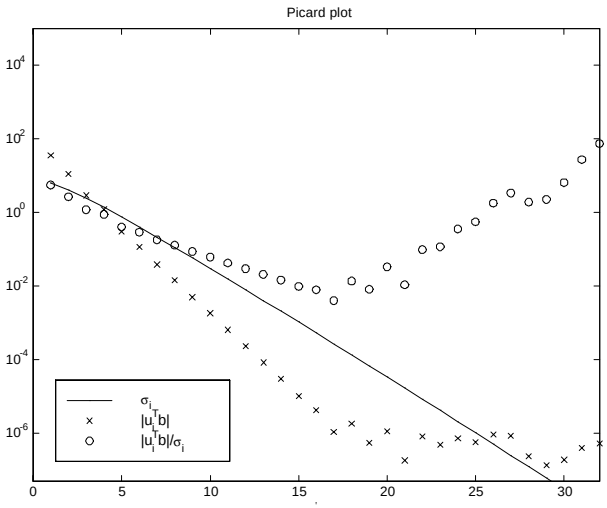

Figure 3.5: Same plot as before; noise with $\sigma_{\text{noise}} = 10^{-6}$ added to the right-hand side.

decay region, the quantities $|u_i^T b|$ decay strictly faster than the singular value, such that in the same region the quantities $|u_i^T b/\sigma_i|$ also decay, only slower. In the region where $|u_i^T b|$ levels off, the quantities $|u_i^T b/\sigma_i|$ increase. The "naive" solution x_{naive} given by Eq. (3.3) is completely dominated by the SVD components corresponding to the smallest singular values, and therefore x_{naive} appears as a highly oscillatory solution with a large norm $\|x_{\text{naive}}\|_2 = 6.1 \cdot 10^{15}$. Figure 3.1 from the previous chapter confirms this.

It is not a coincidence that the absolute values of the SVD components $u_i^T b$ in this example decay faster than the singular values. It can be shown that as long as there exists a square integrable solution to the underlying integral equation (i.e., a solution f such that $\int_0^1 f(t)^2\,dt < \infty$), then independently of the particular discretization scheme the quantities $|u_i^T b|$ will decay faster than the singular values σ_i (until they eventually hit the machine precision level). It is beyond the scope of these notes to go further into the underlying theory; see [10] and [11] for details. Before attempting to solve any discretization of a first-kind Fredholm equation, one should always try to check whether the quantities $|u_i^T b|$ indeed decay faster than the singular values – otherwise there is no point in trying to solve the problem.

We now repeat the SVD analysis of the same problem as before, but with noise added to the right-hand side $\widetilde{b} = b + e$, where the noise vector e has normally distributed elements with zero mean and standard deviation σ_{noise}. Figures 3.5 and 3.6 show plots of the SVD quantities σ_i, $u_i^T \widetilde{b}$, and $u_i^T \widetilde{b}/\sigma_i$ for two noise levels,

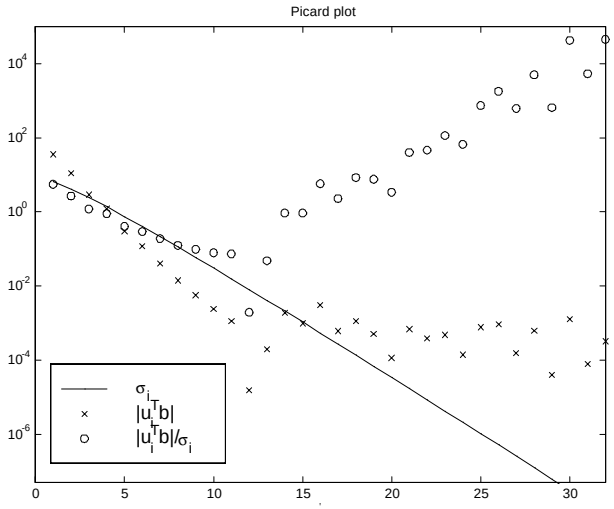

Figure 3.6: Same plot as before; noise with $\sigma_{\text{noise}} = 10^{-3}$ added to the right-hand side.

$\sigma_{\text{noise}} = 10^{-6}$ and $\sigma_{\text{noise}} = 10^{-3}$. We see that as the noise level increases, the decay region of the quantities $|u_i^T \widetilde{b}|$ gets narrower, and the norm of the "naive" solution increases.

What happens in these plots is obviously that for small indices i, the SVD components $u_i^T \widetilde{b}$ are dominated by the exact parts $u_i^T b$, while for larger indices in the region where $|u_i^T \widetilde{b}|$ has levelled off, the SVD components $u_i^T \widetilde{b}$ are dominated by the noise part $u_i^T e$. In this last region, the information in the exact right-hand side b is lost, due to the noise. Moreover, the number of lost components increases as the noise level increases. Consequently, only the first SVD components $u_i^T \widetilde{b}/\sigma_i$ of the "naive" solution carry information about the desired solution. Hence, all numerical algorithms – direct or iterative – that attempt to compute x_{naive} fail to compute a reasonable approximation to the desired solution; and increasing the problem size n or the machine precision do not provide any improvements.

4. Regularization

It is now the time to present various algorithms for stabilizing the computed solution, such that it becomes less sensitive to the perturbations. This process is called regularization, and the corresponding algorithms are called regularization algorithms.

4.1. A Simple Approach: Truncated SVD

It should be clear from the discussion in the previous chapter that although the "naive" solution $x_{\text{naive}} = A^{-1}b$ is useless, because it is dominated by the contributions from the errors in the right-hand side, some of the SVD components still carry useful information about the desired solution. This information is associated with the first SVD components, corresponding to the largest singular values. The question is how to extract this information, while discarding the remaining, erroneous SVD components.

Clearly, a simple "brute force" approach to achieve this is to actually compute the SVD and neglect all the undesired SVD components. This type of regularization is called *truncated SVD* (TSVD), and the TSVD solution is computed as

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i, \tag{4.1}$$

where the truncation parameter k must be chosen appropriately, e.g., from a plot of the SVD quantities, such that only the desired SVD components are retained in the TSVD solution x_k. The TSVD method is implemented in Regularization Tools as the function tsvd.

In spite of its simplicity, this regularization method has been used successively in a variety of applications. Figure 4.1 shows some TSVD solutions to the geomagnetic model problem with $\sigma_{\text{noise}} = 10^{-3}$, along with the exact solution.

The choice of the truncation parameter k is quite straightforward. Monitor the quantities $|u_i^T b|$ and choose k at the transition between the decaying region and the flat region. This ensures that x_k consists mainly of those SVD components that can be trusted; cf. the discussion in the previous chapter. There are also other

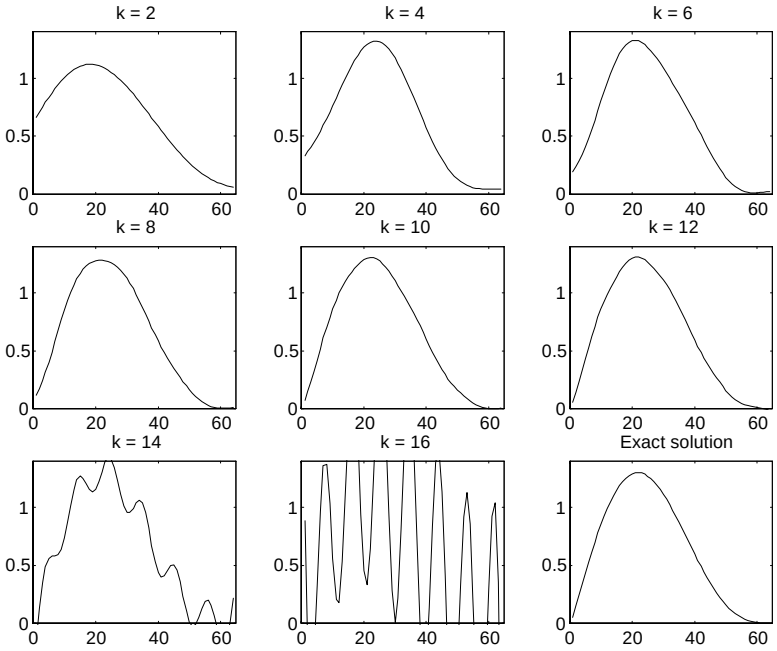

Figure 4.1: TSVD solutions x_k to the geomagnetic prospecting model problem with $\sigma_{\text{noise}} = 10^{-3}$ for $k = 2, 4, \ldots, 16$, and the exact solution.

techniques available that do not require the monitoring of the $|u_i^T b|$-quantities; but this topic lies outside the scope of this note (see [11]).

There is an alternative formulation of the TSVD methods which gives more insight. We first introduce the rank-k matrix A_k, defined in terms of the SVD:

$$A_k = \sum_{i=1}^{k} u_i \, \sigma_i \, v_i^T$$

where k is the TSVD truncation parameter. Then the TSVD solution solves the following minimization problem

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

i.e., there is an infinity of solutions to the rightmost minimization problem involving the rank deficient matrix A_k, and we single out the unique solution with minimum 2-norm. We mention in passing that other regularized solutions with different properties can be obtained by replacing the 2-norm in the left minimization problem with other norms – e.g., the use of the 1-norm leads to solution vectors that represent piecewise polynomial solutions, cf. [12].

We emphasize that the TSVD method is only useful when it is reasonable to compute the SVD. Indeed, this is possible for small problems – but as the problem size increases, it becomes prohibitive to compute the SVD.

4.2. Tikhonov Regularization

This algorithm was developed independently by Phillips [16] and Tikhonov [17]. It is most commonly referred to as *Tikhonov regularization*, and occasionally as damped least squares. The key idea is to accept a nonzero residual $A \, x - b$ and in return obtain a smaller solution norm. This problem can be formulated as follows

$$\min \left\{ \|A \, x - b\|_2^2 + \lambda^2 \|x\|_2^2 \right\} , \tag{4.2}$$

where the regularization parameter λ controls the weight given to minimization of the solution norm $\|x\|_2$ relative to minimization of the residual norm $\|A \, x - b\|_2$. It can be shown that there is always a unique solution x_λ to the above problem, which we denote the Tikhonov solution. Note that as λ approaches zero, the Tikhonov solution x_λ approaches the "naive solution" x_{naive} (or the least squares solution, if A is rectangular), while $x_\lambda \to 0$ as $\lambda \to \infty$. In between, there is (hopefully) a range of λ-values for which there is a reasonable balance between the residual and solution norms, and for which the Tikhonov solution x_λ is a reasonable approximation

to the desired solution. In REGULARIZATION TOOLS, Tikhonov regularization is implemented in the function tikhonov.

To obtain a deeper insight into the properties of the Tikhonov solution, we can express x_λ in terms of the SVD of A as follows:

$$x_\lambda = \sum_{i=1}^{n} \frac{\sigma_i^2}{\sigma_i^2 + \lambda^2} \frac{u_i^T b}{\sigma_i} v_i.$$

The quantities $f_i = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ are called the Tikhonov *filter factors*. They all satisfy $0 \leq f_i \leq 1$, and they control the damping of the individual SVD components of the solution x_λ. Specifically, if λ is fixed somewhere between σ_1 and σ_n, then for $\sigma_i \gg \lambda$ we have $f_i = 1 + \mathcal{O}(\lambda^2/\sigma_i^2) \simeq 1$, while for $\sigma_i \ll \lambda$ we have $f_i = \sigma_i^2/\lambda^2 + \mathcal{O}(\sigma_i^4/\lambda^4) \simeq \sigma_i^2/\lambda^2$. For singular values σ_i near λ, the filter factors are in a transition region between the two above extremes. Thus, we see that the first SVD components, corresponding to singular values greater than λ, contribute with almost full strength to the Tikhonov solution x_λ. Similarly, the last SVD components corresponding to singular values smaller than λ are damped considerably and therefore contribute very little to x_λ. Hence, we would expect that the Tikhonov solutions resemble the SVD solutions when k and λ are chosen such that $\sigma_k \simeq \lambda$; more details can be found in [9].

Figure 4.2 shows various Tikhonov solutions to the geomagnetic prospecting problem with $\sigma_{\text{noise}} = 10^{-3}$ for a large range of λ-values – notice the resemblance of these solutions with the TSVD solutions in Fig. 4.1.

Equation (4.2) is not suited for numerical computations, but there are two other formulations that lend themselves more to numerical computations. The first of them is the following least squares formulation

$$\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \tag{4.3}$$

where I is the identity matrix of order n, and the other is the normal equations for this least squares problem

$$(A^T A + \lambda^2 I)\, x = A^T b.$$

While the latter may be suited in certain special situations, it is the least squares formulation (4.3) that is best suited for numerical computations.

We shall now describe a two-stage algorithm developed by Eldén [5] for computing the Tikhonov solution. The first step consists of computing a *bidiagonalization* of the coefficient matrix A:

$$A = \overline{U}\, B\, \overline{V}^T,$$

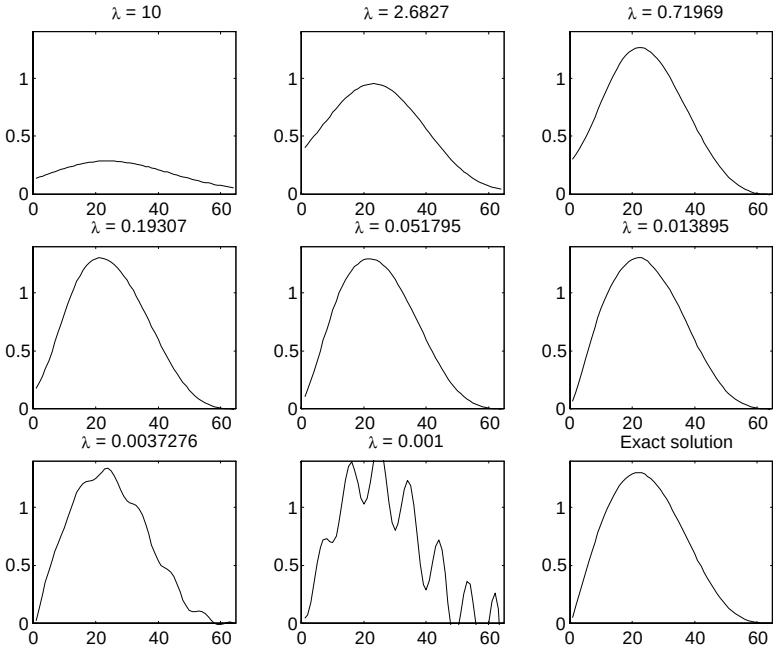

Figure 4.2: Tikhonov solutions x_λ to the geomagnetic prospecting model problem with $\sigma_{\text{noise}} = 10^{-3}$ for λ in the range 10^{-3} to 10, plus the exact solution.

where $\overline{U}$ and $\overline{V}$ are orthogonal matrices, and B is an upper bidiagonal matrix. This decomposition resembles the SVD, and in fact most algorithms for computing the SVD include bidiagonalization as the first step. The bidiagonalization is most conveniently implemented by means of alternating Householder transformations, and requires $4\,(m - n/3)\,n^2$ flops, cf. §5.4.3 in [8]. Software for computing the bidiagonalization is available in many software packages, either as stand-alone routines or as part of the SVD routines; and it is available in REGULARIZATION TOOLS as the function bidiag. Once the bidiagonalization of A has been computed, we can rewrite (4.3) as

$$\min \left\| \begin{pmatrix} B \\ \lambda I \end{pmatrix} \xi - \begin{pmatrix} \overline{U}^T b \\ 0 \end{pmatrix} \right\|_2 ,$$

where $\xi = \overline{V}^T x \Leftrightarrow x = \overline{V}\xi$. The second step of Eldén's algorithm is an efficient way to solve the above equation for ξ, and this is done by a sequence of strategic Givens rotations that annihilate the diagonal elements of λI one at a time, starting from the top. First we apply a Givens rotation to rows 1 and $n+1$ that annihilates the first element of λI, followed by a Givens rotation applied to rows $n+1$ and $n+2$ that annihilates the newly created fill in position $(n+1, 2)$:

$$\begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix} \rightarrow \begin{pmatrix} * & * & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ 0 & * & & \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & 0 & & \\ & * & & \\ & & \times & \\ & & & \times \end{pmatrix} .$$

Next, we apply a Givens rotation to rows 2 and $n+2$ that annihilates the second element of λI, followed by a Givens rotation applied to rows $n+2$ and $n+3$ that annihilates the fill in position$(n+2, 3)$:

$$\begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & & \\ & * & * & \\ & & \times & \times \\ & & & \times \\ & 0 & * & \\ & & \times & \\ & & & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & 0 & \\ & & * & \\ & & & \times \end{pmatrix} .$$

This process is repeated until all elements of λI have been removed, and the complete step requires approximately $7\,n$ flops.

The power of Eldén's algorithm lies in the fact that the regularization problem in bidiagonal form can be solved in only $\mathcal{O}(n)$ operations for each new value of λ, which is important because λ usually has to be determined via computing and inspecting a sequence of Tikhonov solutions for varying λ-values. Most methods for computing a suited value of λ do not require x_λ explicitly, but only the solution norm and the residual norm – and these norms can also be computed in $\mathcal{O}(n)$ operations due to the following relations

$$\|x\|_2 = \|\xi\|_2 \qquad \text{and} \qquad \|A\,x - b\|_2 = \|B\,\xi - \overline{U}^T b\|_2.$$

Hence, the transformation $x = \overline{V}\xi$ back to the original setting, which requires $2\,n^2$ operations, needs usually only be done once.

4.3. Variations of Tikhonov Regularization

When A is symmetric and positive definite, there exists an alternative form of Tikhonov's method – usually attributed to Franklin – that avoids the normal equations by instead working with the system

$$(A + \lambda\,I)\,x = b, \tag{4.4}$$

which can be solved by means of Cholesky factorization. If $A = \sum_{i=1}^{n} w_i\,\alpha_i\,w_i^T$ is the eigenvalue decomposition of A, then the regularized solution in Franklin's method is given by $x = \sum_{i=1}^{n} (w_i^T b)/(\alpha_i + \lambda)\,w_i$. This method does not seem to be used much in practise.

At this stage, it should be remarked that the 2-norm of the solution is not always the optimal quantity to include in the Tikhonov formulation. Experience, supported by theory, suggest that for some problems, it is better to minimize the 2-norm of a quantity that approximates a derivative of the underlying solution. For example, if we use the midpoint quadrature rule for discretization, then approximations to the first and second derivatives are given by $L_1\,x$ and $L_2\,x$, respectively, where the two matrices L_1 and L_2 are given by

$$L_1 = n^{-1} \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \tag{4.5}$$

and

$$L_2 = n^{-2} \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix}. \tag{4.6}$$

Notice that these matrices are not square: L_1 is $(n-1)\times n$ and L_2 is $(n-2)\times n$. This leads to the formulation of Tikhonov regularization in general form – as opposed to the standard form in (4.2):

$$\min \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2 \right\}. \tag{4.7}$$

This general-form version of Tikhonov regularization is also available via the function tikhonov in REGULARIZATION TOOLS. Eldén's algorithm can be extended to treat the general case $L \neq I$, but we omit the details here; see instead [5] or §5.1.1 in [11].

Tikhonov regularization can be extended in other ways. For example, one can add certain constraints to the Tikhonov solution, such as nonnegativity, monotonicity, or convexity. All three constraints can be formulated as inequality constraints of the form $G\,x \geq 0$, taking the following special forms

$$\begin{aligned} x &\geq 0 \quad \text{(nonnegativity)} \\ L_1\,x &\geq 0 \quad \text{(monotonicity)} \\ L_2\,x &\geq 0 \quad \text{(convexity)} \end{aligned}$$

where L_1 and L_2 are the two matrices from above. The constraints can, of course, be combined by stacking the constraint matrices in G. Thus, the inequality-constrained Tikhonov solution solves the problem

$$\min \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|L\,x\|_2^2 \right\} \qquad \text{subject to} \qquad G\,x \geq 0.$$

A Matlab function tikhcstr for solving this inequality constrainted Tikhonov problem is available from the author's home page.

Other variants of Tikhonov regularization arise when the two norms in (4.7) are changed. Regarding the residual norm, it is often appropriate to include a weighting matrix arising from statistical knowledge about the errors e in the right-hand side. Specifically, if the covariance matrix for these errors has the form $C\,C^T$, where C may be a Cholesky factor, then the residual norm $\|A\,x - b\|_2$ should be replaced with $\|C^{-1}(A\,x - b)\|_2$. Regarding the solution norm, the 1-norm $\|L\,x\|_1$ is an interesting alternative to the 2-norm, because the 1-norm can provide regularized solutions with steep gradients and small curvatures – which is not possible with the 2-norm. In certain problems one can also replace the 2-norm with the negative of the entropy function, i.e., $\sum_{i=1}^{n} x_i \log(w_i\,x_i)$, where x_i are the elements of the solution

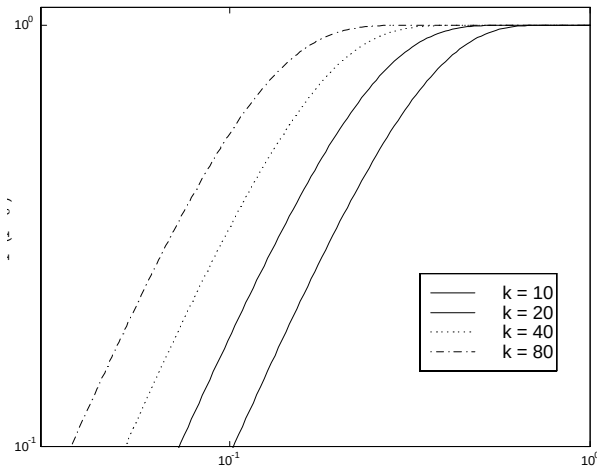

Figure 4.3: The function $1-(1-\sigma^2)^k$ which defines the Landweber filter factors, for $\omega = 1$ and $k = 10$, 20, 40, and 80.

x, and w_i are associated weights that depend on the particular application. Entropy regularization is possible when the solution has strictly positive elements, and it is often used in image processing.

4.4. Iterative Methods

Iterative regularization methods are important for treating large-scale problems, for which the direct computations involved in, e.g., the SVD algorithm and in Eldén's algorithm, become prohibitive. Iterative methods only "touch" the coefficient matrix A via matrix-vector multiplication, and these methods are therefore well suited for sparse and structured matrices.

We start with a classical stationary method called *Landweber iteration* (developed independently also by Cimino, Fridman, Picard, Richardson, and many others). Let $x^{(0)}$ denote the starting vector, and often $x^{(0)} = 0$; then Landweber iteration takes the form

$$x^{(k)} = x^{(k-1)} + \omega\, A^T(b - A\, x^{(k-1)}), \qquad k = 1, 2, 3, \ldots,$$

where ω is a real positive parameter satisfying $0 < \omega < 2/\|A^T A\|_2$, whose value controls the behavior of the iterations. If we insert the SVD of A into the above

formula, then we can show that the kth iteration vector is given by

$$x^{(k)} = \sum_{i=1}^{n} \left(1 - (1 - \omega\,\sigma_i^2)^k\right) \frac{u_i^T b}{\sigma_i}\, v_i,$$

showing that the filter factors for this method are given by $f_i^{(k)} = 1 - (1 - \omega\,\sigma_i^2)^k$, for $i = 1, \ldots, n$. Plots of these filter factors are shown in Fig. 4.3 for $k = 10$, 20, 40, and 80. We see that the filter factors resemble those for Tikhonov regularization, with $f_i^{(k)} \simeq 1$ for $\sigma_i^2 \gg 1/\omega$ and $f_i^{(k)} \simeq k\,\omega\,\sigma_i^2$ for $\sigma_i^2 \ll 1/\omega$. This analysis shows that the iteration number k plays the role of the regularization parameter: initially, only the largest SVD components are included in the iteration vector $x^{(k)}$, and as we increase the number of iterations we include smaller and smaller SVD components into the iteration vector. The analysis also reveals that Landweber's method converges slowly: doubling the number of iterations merely halves the value of the small filter factors.

The literature is full of similar stationary iterative methods, some of them extensions of the classical Landweber method, and all of them simple to implement and analyze (because the iteration matrix is independent of the right-hand side). Unfortunately, all them share essentially the same slow convergence.

Instead, we turn our attention to the use of the conjugate gradient (CG) algorithm. In connection with regularization problems, we need a variant of CG that solves the normal equations $A^T A\,x = A^T b$ associated with a least squares problem $\min \|A\,x - b\|_2$. This variant is called *CGLS*, and it was described in the original paper by Hestenes and Stiefel [13] where the CG method was originally published. Again, let $x^{(0)}$ denote the initial guess, and define the two auxiliary vectors $r^{(0)} = b - A\,x^{(0)}$ and $d^{(0)} = A^T r^{(0)}$. Then the CGLS iterations take the following form for $k = 1, 2, \ldots$

$$\begin{aligned}
\alpha_k &= \|A^T r^{(k-1)}\|_2^2 / \|A\,d^{(k-1)}\|_2^2 \\
x^{(k)} &= x^{(k-1)} + \alpha_k\,d^{(k-1)} \\
r^{(k)} &= r^{(k-1)} - \alpha_k\,A\,d^{(k-1)} \\
\beta_k &= \|A^T r^{(k)}\|_2^2 / \|A^T r^{(k-1)}\|_2^2 \\
d^{(k)} &= A^T r^{(k)} + \beta_k\,d^{(k-1)}.
\end{aligned}$$

The vector $r^{(k)}$ is the residual vector for the least squares problem, i.e., $r^{(k)} = b - A\,x^{(k)}$, while the vector $d^{(k)}$ is the residual vector for the normal equations, i.e., $d^{(k)} = A^T b - A^T A\,x^{(k)}$.

The CGLS algorithm can be used for regularization in two fashions. One is to apply it to the Tikhonov problem in the least squares formulation (4.3) with A and

b replaced by $\binom{A}{\lambda I}$ and $\binom{b}{\mathbf{0}}$, and in this fashion compute regularized solutions. The other is to apply CGLS directly to $Ax = b$, and use the fact that the iteration number k often plays the same role as a regularization parameter as in Landweber's method.

The first approach needs a good preconditioner to be of practical value, but none of the preconditioners for general problems are suited for Tikhonov regularization problems, due to the spectral properties of the coefficient matrix $A^T A + \lambda^2 I$. A promising specialized preconditioner for regularization problems has been developed recently by Hanke and Vogel, but more experience with the preconditioner is needed before it can be evaluated (it will hopefully be included in future versions of this note).

The second approach is much simpler to use, because the preconditioner is avoided – we merely apply the above CGLS scheme to A and b. Unfortunately, this method is hard to analyze theoretically, because it is a non-stationary method (the iteration matrix depends on the iteration number k) and because the iteration matrix depends on the right-hand side. Once again, we express the iteration vector in terms of the SVD of A, and we obtain

$$x^{(k)} = \sum_{i=1}^{n} f_i^{(k)} \, \frac{u_i^T b}{\sigma_i} \, v_i,$$

where the CGLS filter factors $f_i^{(k)}$ depend in a nonlinear way on both b and all the singular values $\sigma_1, \ldots, \sigma_n$. It can be show that $f_i^{(k)} \simeq 1$ for the large singular values and $f_i^{(k)} = \mathcal{O}(\sigma_i^2)$ for the small singular values. In the transition range, certain filter factors can become slightly larger than one.

Plots of typical filter factors are shown in Fig. 4.4 – notice the resemblance with the Tikhonov filter factors. Also note that for certain values of k, there is a filter factor slightly larger than one. We see once again that the filter factors dampen the SVD components corresponding to small singular values, and the transition value of σ_i at which the damping sets in decreases as k increases. But the converge of the CGLS method is faster than that of Landweber and other classical stationary methods. One can think of CGLS as an iterative scheme for computing approximate TSVD or Tikhonov solutions, in which a number $\ell(k) \leq k$ of SVD components are captured in the kth iteration, cf. Table 4.1.

One of the difficulties with this use of CGLS as an iterative regularization method is to determine under which conditions the algorithm has an intrinsic regularization property. I.e., for which class of problems do the filter factors look as in Fig. 4.4, with a flat region where $f_i^{(k)} \simeq 1$ and a decaying region where $f_i^{(k)} = \mathcal{O}(\sigma_i^2)$. A full analysis of this problem has not been developed yet – partial results can be found

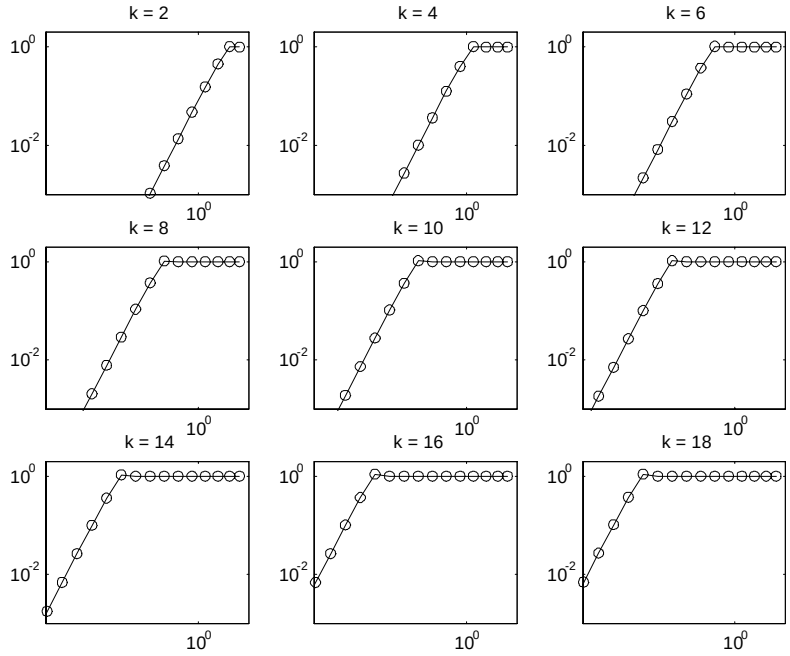

Figure 4.4: CGLS filter factors $f_i^{(k)}$ as functions of the singular values σ_i for the geomagnetic prospecting test problem with $\sigma_{\text{noise}} = 10^{-3}$.

k	2	4	6	8	10	12	14	16	18
$\ell(k)$	2	4	6	7	8	9	10	11	11

Table 4.1: Corresponding values of k and $\ell(k)$ in the CGLS algorithm for the problem in Fig. 4.4.

in [11], where it is concluded that the desired behavior occurs under the following conditions.

- The singular values should decay gradually to zero.
- The decay should not be too gentle.
- The quantities $|u_i^T b|$, i.e., the SVD coefficients of the right-hand side, should decay faster than the singular values.

All three conditions are usually satisfied in connection with discretizations of ill-posed problems – but on the other hand there is no perfect guarantee to obtain the desired features of the filter factors. Things are complicated even further in the presence of finite-precision arithmetic.

An analysis of the MINRES algorithm (an iterative algorithm related to CGLS) is presented by Kilmer and Stewart in [14], where it is demonstrated that for discretizations of ill-posed problems, this algorithm will capture all the SVD components associated with the large singular values before it starts to include the remaining SVD components. It is also shown that the number of captured SVD components in the kth step of this method is often much larger than k.

5. Deconvolution in One Dimension

We now return to the main theme of this note, namely, deconvolution problems, and we show how the above general algorithms specialize – an various ways – for this particular class of regularization problems. The key observation here is that for convolution problems, where the kernel satisfies

$$K(s,t) = h(s-t),$$

the corresponding matrix A derived from a quadrature rule discretization, with elements $a_{ij} = w_j\, h(s_i - t_j)$, can be written in the form

$$A = H\,W, \tag{5.1}$$

where $W = \mathrm{diag}(w_1, \ldots, w_n)$ is a diagonal matrix consisting of the quadrature weights, and the elements of the matrix H are "samples" of h, i.e.,

$$h_{ij} = h(s_i - t_j), \qquad i,j = 1, \ldots, n. \tag{5.2}$$

At this stage we will assume that the quadrature points s_i and the collocation points t_j are identical and chosen equidistantly spaced (recall out assumption that both integration intervals are from 0 to 1), i.e., $s_i = t_i = \alpha + \beta\, i$, $i = 1, \ldots, n$. In this case the elements of the matrix H satisfy

$$h_{ij} = h(s_i - t_j) = h(s_{i+\ell} - t_{j+\ell}) = h_{i+\ell,j+\ell}$$

for all relevant i, j, and ℓ. This special structure of the coefficient matrix can be used to derive very efficient algorithms. We shall first explore the structure of H, and then turn to the regularization algorithms.

5.1. Toeplitz Matrices and their SVD

A *Toeplitz matrix* T is a matrix whose elements depend only on the difference $i-j$ between the indices, i.e., T can be written as

$$T = \begin{pmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{1-n} \\ t_1 & t_0 & t_{-1} & \cdots & t_{2-n} \\ t_2 & t_1 & t_0 & \cdots & t_{3-n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & t_{n-3} & \cdots & t_0 \end{pmatrix}. \tag{5.3}$$

Notice that an $n \times n$ Toeplitz matrix is characterized by only $2n-1$ different elements (or merely n elements if T is symmetric), and it is this feature that makes it possible to derive efficient algorithms for Toeplitz matrices. Obviously, the matrix H in Eq. (5.2) is a Toeplitz matrix when the quadrature and collocation points are identical and equidistantly spaced.

Toeplitz matrices T are persymmetric, i.e., they are symmetric across the antidiagonal. Hence, in addition to the relation $t_{ij} = t_{i-\ell,j-\ell} = t_{i-j}$ for all relevant i, j, and ℓ, their elements satisfy the relation $t_{ij} = t_{n-j+1,n-i+1}$. This can also be expressed as the fact that the Toeplitz matrix with its columns in reverse order is symmetric. Let J denote the exchange matrix

$$J = \begin{pmatrix} & & 1 \\ & \cdot & \\ 1 & & \end{pmatrix}$$

which, when multiplied from the right to T, reverses the order of the columns of T. Thus we have $T\,J = (T\,J)^T = J\,T^T$ and hence (because $J^2 = I \Leftrightarrow J^{-1} = J$) the persymmetry of T can be expressed as $T = J\,T^T J$. One of the implications of this property is that the inverse of T is also persymmetric, since $T^{-1} = (J\,T^T J)^{-1} = J^{-1}(T^{-1})^T J^{-1} = J\,(T^{-1})^T J$.

The persymmetry of T can be used to derive certain symmetry relations between the singular vectors of T. From the symmetry of the matrix $T\,J$, it follows from (3.1) that

$$T\,J = \sum_{j=1}^{n} w_j\,\gamma_j \lambda_j\,(\gamma_j w_j)^T \qquad \Leftrightarrow \qquad T = \sum_{j=1}^{n} w_j\,\gamma_j \lambda_j\,(\gamma_j J w_j)^T,$$

where the quantities $\gamma_j = \pm 1$ are chosen to make $\gamma_j \lambda_j$ positive. The rightmost equation above is identical to the SVD of T, and we recognize $u_j = w_j$ and $v_j =$

$\gamma_j J w_j = \gamma_j J u_j$. Hence, the elements of the left and right singular vectors are related by

$$v_{ij} = \gamma_j \, u_{n-i+1,j}, \qquad i,j = 1,\ldots,n.$$

I.e., except perhaps for a sign change, the vector v_j is identical to u_j with its elements in reverse order.

If T is symmetric such that $u_j = \widehat{\gamma}_j v_j$, where the quantities $\widehat{\gamma}_j = \pm 1$ are generally different from the γ_j, then additional symmetries occur because $u_{ij} = \widehat{\gamma}_j v_{ij} = \widehat{\gamma}_j(\gamma_j u_{n-i+1,j}) = \widehat{\gamma}_j \gamma_j u_{n-i+1,j}$, and similarly for the right singular vectors. We can summarize these symmetry relations as follows:

$$|u_{ij}| = |u_{n-i+1,j}| = |v_{ij}| = |v_{n-i+1,j}|, \qquad i,j = 1,\ldots,n.$$

In other words, the left and right singular vectors 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. We illustrate this by means of a small example:

$$T = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix},$$

$$U = \begin{pmatrix} .628 & .707 & -.325 \\ .460 & 0 & .888 \\ .628 & -.707 & -.325 \end{pmatrix}, \qquad V = \begin{pmatrix} .628 & -.707 & .325 \\ .460 & 0 & -.888 \\ .628 & .707 & .325 \end{pmatrix}.$$

Here, $u_1 = J\,u_1 = v_1 = J\,v_1$, $u_2 = -J\,u_2 = -v_2 = J\,v_2$, and $u_3 = J\,u_3 = -v_3 = -J\,v_3$. The singular values of T are $\sigma_1 = 2.73$, $\sigma_2 = 2$, and $\sigma_3 = .732$.

5.2. Circulant Matrices and Convolution

Circulant matrices form a special class of Toeplitz matrices in which the bottom element of a column "wraps around" as the column is repeated to the right in a down-shifted version. Hence, an $n \times n$ *circulant matrix* C has the form

$$C = \begin{pmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{pmatrix},$$

i.e., C has n different elements satisfying the relation $c_{ij} = c_{(i-j) \bmod n}$. Among other things, circulant matrices can be used to express the convolution of two periodic

signals in matrix form. recall that if f and h are two n-periodic signals, then the convolution of f and h is a new n-periodic signal g with elements given by $g_i = \sum_{j=1}^{n} f_j\, h_{i-j}$. Thus, if h is the first column of C_h, then g is obviously given by

$$g = C_h\, f. \tag{5.4}$$

Note that we can interchange the roles of f and h without changing g, hence there is nothing special about the circulant matrix being made up of h.

An amazing fact about circulant matrices is that all circulant matrices of order n have the same eigenvectors, and the eigenvalues are given by the DFT of the first column of C. In order to derive this result, we use the fact that the DFT of an n-vector f can be written as $\widehat{f} = \mathrm{DFT}(f) = F_n\, f$, where the elements of the *complex symmetric* matrix F_n are given by

$$(F_n)_{ij} = (\exp(-2\pi\hat{\imath}/n))^{(i-1)(j-1)}\,,$$

where $\hat{\imath}$ is the imaginary unit. The inverse of F_n is given by $F_n^{-1} = n^{-1}\mathrm{conj}(F_n)$ – notice the complex conjugation – showing that the IDFT can be written as $f = \mathrm{IDFT}(\widehat{f}) = F_n^{-1}\widehat{f} = n^{-1}\mathrm{conj}(F_n)\,\widehat{f}$. We now insert these two relations into the expression (1.1) for g to obtain

$$\begin{aligned} g &= \mathrm{IDFT}(\mathrm{DFT}(h).*\mathrm{DFT}(f)) \\ &= F_n^{-1}(F_n\, h).*(F_n\, f)) \\ &= F_n^{-1}(\mathrm{diag}(F_n\, h)\, F_n\, f) \\ &= F_n^{-1}\mathrm{diag}(F_n\, h)\, F_n\, f. \end{aligned}$$

Via the definition of F_n and the expression

$$F_n^{-1}\mathrm{diag}(F_n h)\, F_n = n^{-1}\mathrm{conj}(F_n)\,\mathrm{diag}(F_n h)\, F_n$$

it is easy to show that this matrix is circulant. Hence, the expression for g is identical to the expression in (5.4) if we choose $C_h = F_n^{-1}\mathrm{diag}(F_n\, h)\, F_n$, and since there are only n free elements in C_h as well as in $F_n^{-1}\mathrm{diag}(F_n\, h)\, F_n$ this relation is unique. This shows that $\mathrm{diag}(F_n h)$ is a similarity transform of C_h, which preserves eigenvalues, and hence the elements of the vector $F_n h$ must be the eigenvalues of C_h. Moreover, the columns of the matrix $n^{-1/2}F_n$ are orthonormal eigenvectors of C_h.

At this stage we emphasize that all DFTs and IDFTs should be computed by means of the FFT algorithm. If n is a power of two, then a complex FFT requires $5\, n\, \log_2 n$ flops, and if data are real then the FFT can be implemented in such a way that only half as many flops are required. These numbers actually ignore the

overhead in computing the complex exponentials; in Matlab the total amount of flops is $2.5\,n\,\log_2 n + 5\,n + \mathcal{O}(1)$ for real data. Notice that the interchanges of the elements that is usually part of an FFT algorithm is not necessary in the above application and can therefore be switched off, thus saving computational overhead (although this is not possible in Matlab).

5.3. Matrix-Vector Multiplication by FFT

In connection with iterative algorithms, it is important to know that matrix-vector multiplication with a Toeplitz matrix can be performed in $\mathcal{O}(n\,\log_2 n)$ flops – as opposed to $2\,n^2$ flops for a general matrix-vector multiplication. We derive the algorithm here, and more details can be found in [18].

The key idea is to embed the $n \times n$ Toeplitz matrix T in a larger $p \times p$ circulant matrix C, and use the FFT algorithm to implement the fast matrix-vector multiplication with C. Specifically, using the notation of Eq. (5.3), the first column of C is constructed such that

$$C(:,1) = (t_0, t_1, \ldots, t_{n-1}, 0, \ldots, 0, t_{1-n}, \ldots, t_{-1})^T.$$

The middle part of this vector consists of $p - 2n + 1$ zeros, and p should be chosen to be the smallest possible power of 2 satisfying $p \geq 2n$ in order to speed up the FFT computations. For example, if T is the nonsymmetric 3×3 matrix

$$T = \begin{pmatrix} 3 & -2 & -1 \\ 2 & 3 & -2 \\ 1 & 2 & 3 \end{pmatrix}$$

then $p = 8$ and C takes the form

$$C = \begin{pmatrix} 3 & -2 & -1 & & & & 1 & 2 \\ 2 & 3 & -2 & -1 & & & & 1 \\ 1 & 2 & 3 & -2 & -1 & & & \\ & 1 & 2 & 3 & -2 & -1 & & \\ & & 1 & 2 & 3 & -2 & -1 & \\ & & & 1 & 2 & 3 & -2 & -1 \\ -1 & & & & 1 & 2 & 3 & -2 \\ -2 & -1 & & & & 1 & 2 & 3 \end{pmatrix},$$

where we recognize T as the leading 3×3 principal submatrix of C.

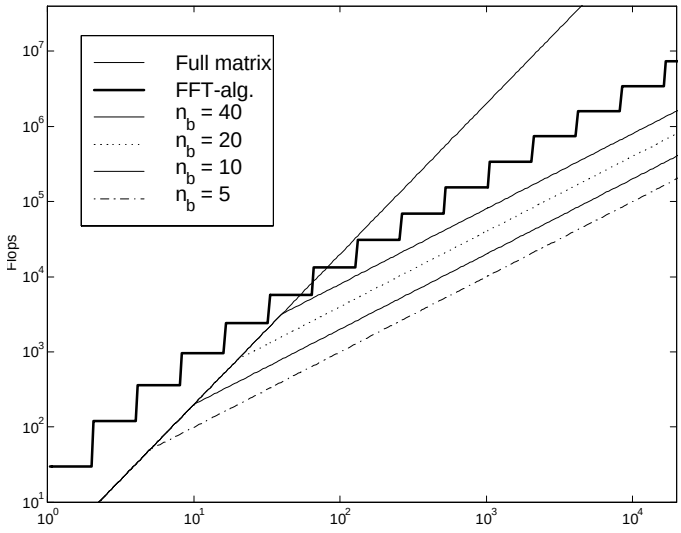

Figure 5.1: Flop counts for the FFT-based Toeplitz matrix-vector multiplication algorithm, compared to flop counts for ordinary matrix-vector multiplication with full and banded matrices.

In order to multiply a vector x with T we first pad this vector with $p - n$ zeros to obtain

$$\widetilde{x} = \begin{pmatrix} x \\ 0 \end{pmatrix},$$

and when we multiply $\widetilde{x}$ with C, we obtain the new vector

$$\widetilde{z} = C\,\widetilde{x} = \begin{pmatrix} T\,x \\ \widetilde{z}_2 \end{pmatrix}.$$

Due to the zeros in $\widetilde{x}$, the first n components of $\widetilde{z}$ are identical to the desired matrix-vector product $T\,x$. If data are real and the DFT of $C(:,1)$ has been precomputed, then the matrix-vector multiplication $C\,\widetilde{x}$ can be computed very efficiently by means of essentially two FFTs (one real and one complex) in only about $7.5\,p\,\log_2 p$ flops. Since p is bounded above by $4n$ we obtain the approximate upper bound $30\,n\,\log_2 n$ for the flop count involved in one matrix-vector multiplication, and if n is a power of 2 (such that $p = 2n$) then the approximate bound reduces to $15\,n\,\log_2 n$ flops.

Note that if T is real and symmetric, then the vector $C(:,1)$ is "real even" and $\mathrm{DFT}(C(:,1))$ is a real vector – but the complexity of the Toeplitz-based matrix-vector multiplication algorithms stays the same.

Whether the FFT-based matrix-vector multiplication actually pays off depends on the elements of the Toeplitz matrix T: if T is a banded matrix with bandwidth n_b (or if the elements outside this band are so small that they can be considered as zeros), then the matrix-vector multiplication requires $2\,n_b\,n$ flops. If n is a power of two, then the break-even value of n, for which the two approaches require about the same work, is $n = 2^{n_b/7.5}$ – i.e., for a fixed bandwidth n_b the ordinary banded matrix-vector multiply is faster than the FFT-based approach for $n > 2^{n_b/7.5}$. Figure 5.1 illustrates this point.

5.4. Direct Algorithms for Toeplitz Matrices

We have shown in the previous section that iterative regularization algorithms can be implemented very efficiently for convolution problems with Toeplitz matrices. As we shall see in this section, the same is true for direct methods implementing Tikhonov's method. Since the diagonal weight matrix W in (5.1) can always be "absorbed" into the solution vector x, we can assume without loss of generality that the coefficient matrix A is a Toeplitz matrix.

We consider first the special case where the coefficient matrix A is triangular and Toeplitz, and L is a $p \times n$ Toeplitz matrix with dimensions $p \leq n$ and with zeros below the main diagonal. Triangular Toeplitz matrices arise, e.g., in connection with Volterra-type deconvolution problems (1.4), and L is almost always a banded Toeplitz matrix when the integral equation is discretized with equidistant abscissas. We shall here consider the case where A is upper triangular; problems with a lower triangular coefficient matrix can easily be brought into this form. The coefficient matrix in the least squares formulation of Tikhonov regularization then takes the following from (shown here for $n = 6$ and $p = 4$, and ignoring the possible band structure of L):

$$\begin{pmatrix} A \\ \lambda\, L \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ & a_1 & a_2 & a_3 & a_4 & a_5 \\ & & a_1 & a_2 & a_3 & a_4 \\ & & & a_1 & a_2 & a_3 \\ & & & & a_1 & a_2 \\ & & & & & a_1 \\ l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \\ & l_1 & l_2 & l_3 & l_4 & l_5 \\ & & l_1 & l_2 & l_3 & l_4 \\ & & & l_1 & l_2 & l_3 \end{pmatrix}$$

We shall now demonstrate how to transform this matrix into upper triangular form by means of n Givens transformations. First, we apply a Givens rotation to

rows 1 and $n+1$ in order to annihilate the element l_1; this rotation changes all the elements in the two rows, and we denote the new elements with a prime, e.g., the (1,1)-element becomes a'_1. The same rotation is applied (implicitly) to the row pairs $(j, n+j)$ for $j = 2, \ldots, p$, and due to the Toeplitz structure of A and L the new rows are just shifted versions of rows 1 and $n+1$. Thus, we obtain the first intermediate matrix

$$M' = \begin{pmatrix} a'_1 & a'_2 & a'_3 & a'_4 & a'_5 & a'_6 \\ & a'_1 & a'_2 & a'_3 & a'_4 & a'_5 \\ & & a'_1 & a'_2 & a'_3 & a'_4 \\ & & & a'_1 & a'_2 & a'_3 \\ & & & & a'_1 & a'_2 \\ & & & & & a'_1 \\ & l'_2 & l'_3 & l'_4 & l'_5 & l'_6 \\ & & l'_2 & l'_3 & l'_4 & l'_5 \\ & & & l'_2 & l'_3 & l'_4 \\ & & & & l'_2 & l'_3 \end{pmatrix}.$$

We now apply the same procedure to the submatrix $M'(2\colon n+p, 2\colon n)$ to obtain the second intermediate matrix M'', where a double prime denotes an element that is changed in the second step:

$$M'' = \begin{pmatrix} a'_1 & a'_2 & a'_3 & a'_4 & a'_5 & a'_6 \\ & a''_1 & a''_2 & a''_3 & a''_4 & a''_5 \\ & & a''_1 & a''_2 & a''_3 & a''_4 \\ & & & a''_1 & a''_2 & a''_3 \\ & & & & a''_1 & a''_2 \\ & & & & & a''_1 \\ & & l''_3 & l''_4 & l''_5 & l''_6 \\ & & & l''_3 & l''_4 & l''_5 \\ & & & & l''_3 & l''_4 \\ & & & & & l''_3 \end{pmatrix}.$$

Obviously, we can repeat this process on the submatrix $M''(3\colon n+p, 3\colon n)$, and so on, until we end up with an upper triangular matrix after n steps. The total amount of work is approximately $8\,n^2$, and the storage requirement is dominated by the need to store $n^2/2$ elements of the final triangular matrix. This algorithm is due to Eldén [6].

Next we consider the more general case where A is a general $m \times n$ Toeplitz matrix, and we present an algorithm due to Bojanczyk, Brent, and de Hoog [3] (with improvements by Park and Eldén [15]) for computing the triangular matrix R

in the QR factorization

$$\begin{pmatrix} A \\ \lambda I \end{pmatrix} = Q\,R.$$

The algorithm can be augmented to treat general L matrices. To get started, we partition the coefficient matrix in the following ways

$$\begin{pmatrix} A \\ \lambda I \end{pmatrix} = \begin{pmatrix} a_0 & u^T \\ v & \widehat{A} \\ \lambda & 0^T \\ 0 & \lambda\,\widehat{I} \end{pmatrix} = \begin{pmatrix} \widehat{A} & \widetilde{u} \\ \widetilde{v}^T & a_{m-n} \\ \lambda\,\widehat{I} & 0 \\ 0^T & \lambda \end{pmatrix},$$

where $\widehat{A}$ is the leading/trailing $(m-1)\times(n-1)$ submatrix of A, $\widehat{I}$ is the identity matrix of order $n-1$, and the four vectors u, v, $\widetilde{u}$, and $\widetilde{v}$ are chosen to fill out A correctly. We also partition the $n\times n$ upper triangular matrix R accordingly in the following two ways

$$R = \begin{pmatrix} r_{11} & z^T \\ 0 & R_b \end{pmatrix} = \begin{pmatrix} R_t & \widetilde{z} \\ 0^T & r_{nn} \end{pmatrix}.$$

By comparing the submatrices in the equation

$$\begin{pmatrix} A \\ \lambda I \end{pmatrix}^T \begin{pmatrix} A \\ \lambda I \end{pmatrix} = R^T R$$

we are lead to the following relations for the three nonzero blocks of R:

$$\begin{aligned} r_{11}^2 &= a_0^2 + v^T v + \lambda^2 \\ z^T &= (a_0\,u^T + v^T\widehat{A})/r_{11} \\ R_b^T R_b &= R_t^T R_t + u\,u^T - \widetilde{v}\,\widetilde{v}^T - z\,z^T. \end{aligned}$$

Notice that λ enters only explicitly in the formula for r_{11}.

We can use the first two relations to immediately compute the first row of R, namely, $(r_{11}\,,\,z^T)$. The third relation shows that the submatrix R_b is related to R_t via a sequence of three rank-one modifications. Hence, if we knew R_t then we could compute R_b by means of well-established numerical techniques for up- and downdating, see, e.g., §12.5 in [8].

The heart of the algorithm is to recognize that we already know the first row of R_t, which is identical to $R(1,1{:}\,n-1)$, and with this information we can compute the first row of R_b. First we determined a Givens rotation G_1 which, when applied to the first row of R_t and u^T, annihilates the first element of u, i.e.,

$$G_1 \begin{pmatrix} R_t \\ u^T \end{pmatrix} = \begin{pmatrix} \widehat{R}_t \\ (u')^T \end{pmatrix},$$

where $u'(1) = 0$. Note that $\widehat{R}_t$ is identical to R_t except for its first row. Next we determine a second Givens rotation $\widehat{G}_1$ which, when applied to $\widetilde{v}^T$ and z^T, annihilates the first element of z, i.e.,

$$\widehat{G}_1 \begin{pmatrix} \widetilde{v}^T \\ z^T \end{pmatrix} = \begin{pmatrix} \widehat{v}^T \\ (z')^T \end{pmatrix},$$

where $z'(1) = 0$. Finally we determine a stabilized hyperbolic rotation H_1 which, when applied to the first row of $\widehat{R}_t$ and $\widehat{v}^T$, annihilates the first element of $\widehat{v}$, i.e.,

$$H_1 \begin{pmatrix} \widehat{R}_t \\ \widehat{v}^T \end{pmatrix} = \begin{pmatrix} R'_t \\ (\widetilde{v}')^T \end{pmatrix},$$

where $\widetilde{v}'(1) = 0$. We have now finished the computation of the first row of R_b and thus the second row of the desired triangular matrix R.

To proceed, we continue the up- and downdating process on the second row of R'_t. First we use two Givens rotations G_2 and $\widehat{G}_2$ to annihilate the second element of u' by means of the second row of R'_t, and to annihilate the second element of z' by means of $\widetilde{v}'$, respectively:

$$G_2 \begin{pmatrix} R'_t \\ (u')^T \end{pmatrix} = \begin{pmatrix} \widehat{R}'_t \\ (u'')^T \end{pmatrix}, \qquad \widehat{G}_2 \begin{pmatrix} (\widetilde{v}')^T \\ (z')^T \end{pmatrix} = \begin{pmatrix} (\widehat{v}')^T \\ (z'')^T \end{pmatrix}.$$

Then we use a stabilized hyperbolic rotation H_2 to annihilate the second element of $\widehat{v}'$ by means of the second row of R''_t:

$$H_2 \begin{pmatrix} \widehat{R}'_t \\ (\widehat{v}')^T \end{pmatrix} = \begin{pmatrix} R''_t \\ (\widetilde{v}'')^T \end{pmatrix}.$$

At this stage, we have computed the first three rows of the desired matrix R, and it is clear that this process can be repeated until all of R has been computed. The algorithm requires $mn + 6n^2$ flops. Information about Q is not explicitly available, so we must computed the Tikhonov solution via the semi-normal equations

$$R^T R\, x_\lambda = A^T b.$$

This requires additional $(m + 2n)n$ flops, which can be reduced if the FFT is used to compute $A^T b$.

We emphasize that both algorithms must be started over if a regularized solution with a new value of λ is desired. The same is true if we use Levinson's algorithm (cf. §4.7.3 in [8]) to solve the system (4.4) in Franklin's method.

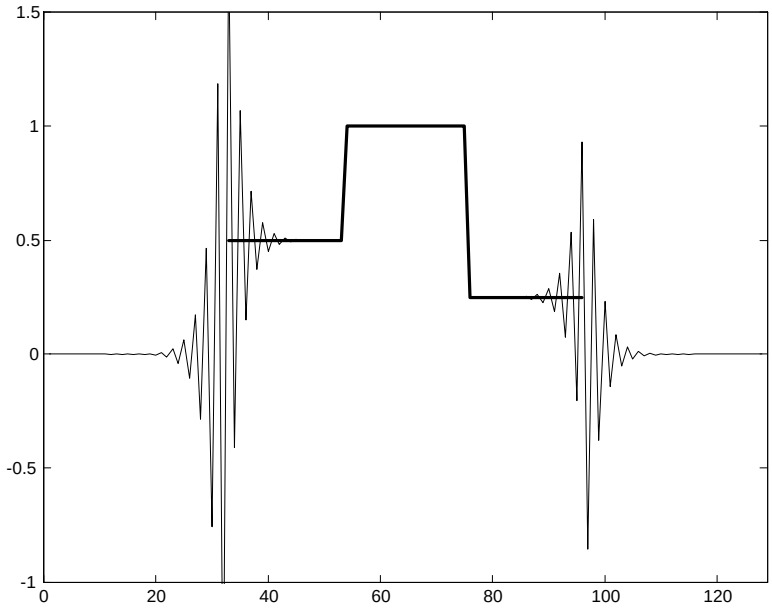

Figure 5.2: The approximate solution $\widetilde{x}$ is embedded in the vector $\widehat{x}$, and the exact solution x is superimposed. Notice the large oscillations ("ringing").

5.5. Periodic vs. Nonperiodic Deconvolution Problems

In the periodic deconvolution problem (1.1), the matrix A is already a circulant matrix, and if regularization is not needed then the solution is given by Eq. (1.2). If regularization is needed, then we can easily modify this formula to include Franklin's version of regularization (4.4). Since $A = W \Lambda W^T$ with $W = n^{-1/2} F_n$ and $\Lambda = \text{diag}(\text{DFT}(h))$, it is easy to see that the regularized solution can be computed as

$$f = \text{IDFT}\left(\text{DFT}(g)\,./(\text{DFT}(h) + \lambda)\right). \tag{5.5}$$

In deconvolution problems where A is not a circulant matrix, i.e., where the signal h is not periodic with period n, Eq. (5.5) is occasionally still used to compute a solution in order to achieve the computational speed of the FFT algorithm, at the cost of a degradation of the computed solution.

Inspired by the FFT-based technique to compute the Toeplitz matrix-vector product fast, a typical approach is to imbed A into a larger circulant matrix C as described above, pad the right-hand side b with additional $p - n$ zeros to make it conform with C, and then compute the vector $\widehat{x} = C^{-1}\widehat{b}$, where $\widehat{b}$ is the augmented

version of b. Then as the approximate solution $\widetilde{x}$ we use those elements of $\widehat{x}$ that correspond to the locations of the b-element in $\widehat{b}$. We note that if a cyclic shift is applied to $\widehat{b}$ then the same cyclic shift applies to $\widehat{x}$, and thus the approximate solution $\widetilde{x}$ is independent on the locations of the zeros in $\widehat{b}$.

Unfortunately, this approach is not guaranteed to yield a good approximation to the desired solution. To see this. we introduce the notation

$$C = \begin{pmatrix} T & C_{12} \\ C_{21} & C_{22} \end{pmatrix}, \qquad \widehat{x} = \begin{pmatrix} \widetilde{x} \\ z \end{pmatrix}, \qquad \widehat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix},$$

and it is easy to show that the inverse of C is given by

$$C^{-1} = \begin{pmatrix} T^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} T^{-1}C_{12}S^{-1}C_{21}T^{-1} & -T^{-1}C_{12}S^{-1} \\ -S^{-1}C_{21}T^{-1} & S^{-1} \end{pmatrix},$$

where the Schur complement S is given by $S = C_{22} - C_{21}T^{-1}C_{12}$. Hence, it follows that the approximation $\widetilde{x}$ can be expressed as

$$\widetilde{x} = x + T^{-1}C_{12}S^{-1}C_{21}x$$

showing that there is no guarantee that $\widetilde{x}$ resembles the exact solution x.

The error component $T^{-1}C_{12}S^{-1}C_{21}x$ often appears as artificial oscillations, called "ringing," at both ends of the approximate solution $\widetilde{x}$. We illustrate this with a numerical example with $n = 64$, a Gaussian kernel chosen such that $a_{ij} = \exp(-(i-j)^2/2)$, and a piecewise constant solution x. The right-hand side $b = A\,x$ is padded with 32 zeros on top and 32 zeros below, and the computed vector $\widehat{x}$ of length $2n = 128$ is shown in Fig. 5.2 with the exact solution x superimposed at the correct position. Notice the large, undesired ringing.

Alternatively, if the underlying signal h is peaked at index 0 and decaying to zero away from this index, thus making A as effectively banded matrix, then one could replace h with a periodic signal of the same length, i.e., with the sequence $h_1, h_2, \ldots, h_{n/2}, h_{n/2+1}, h_{n/2}, \ldots, h_3, h_2$. This correspond to replacing the Toeplitz matrix A with a circulant matrix of the same order whose first column – in Matlab notation – is given by $[A(1\colon n/2, 1); A(n/2+1\colon -1\colon 2)]$. The solution to this modified problem has exactly the same difficulties with ringing as the solution in the previously described approach.

6. Deconvolution in Two Dimensions

In this chapter we discuss certain numerical aspects of 2-D convolution problems. Such problems arise, e.g., in image processing, and the dimensions of these problems quickly get large – for example, 1024×1024 images are now common in astronomy as well as other areas.

Although the underlying techniques and difficulties are essentially the same as in the 1-D case, there are certain new techniques that come in very handy for 2-D problems. We start with a brief introduction to the world of 2-D convolution problems, and then we introduce a very useful tool from linear algebra, the Kronecker product. After that, we turn to the discretization and numerical treatment and solution of 2-D problems.

6.1. 2-D Deconvolution Problems

The general 2-D version of a first-kind Fredholm integral equation takes the form

$$\int_0^1 \int_0^1 K(x, y, x', y')\, f(x', y')\, dx'\, dy' = g(x, y). \tag{6.1}$$

We shall limit our discussion here to the important case where the kernel K is a convolution operator $K(x, y, x', y') = h(x - x', y - y')$ whose variables $x - x'$ and $y - y'$ *separate*, i.e., the kernel has the special form

$$K(x, y, x', y') = \kappa(x - x')\, \omega(y - y'),$$

where κ and ω are functions. When we insert this product into the general form of the integral equation, we notice that due to the separation of the variables in K, the integration can be split into the two variables x' and y':

$$\int_0^1 \kappa(x - x') \left(\int_0^1 \omega(y - y')\, f(x', y')\, dy' \right) dx' = g(x, y). \tag{6.2}$$

This Fredholm integral equation shares exactly the same analytical and numerical difficulties as the 1-D version.

We give an example of such a problem arising in confocal microscopy [2], a technique that provides improved resolution compared to conventional light microscopy. If we assume that the object is uniformly illuminated, and that the collector lens is simply an aperture of width d, then the 2-D object f is related to its image g via a 1-D convolution equation whose kernel separates as in (6.2), with functions κ and ω given by

$$\kappa(z) = \omega(z) = \frac{\sin(\pi dz)}{\pi dz}.$$

Precisely the same 2-D integral equation arises in connection with extrapolation of band-limited signals. 2-D convolution problems with separable kernels also arise in connection with image restoration problems, and we return to this subject shortly.

6.2. Kronecker Products

The *Kronecker product* $\overline{A} \otimes A$ of two matrices A and $\overline{A}$, of dimensions $m \times n$ and $\overline{m} \times \overline{n}$, respectively, is defined as a new matrix of dimensions $m\,\overline{m} \times n\,\overline{n}$ given by

$$\overline{A} \otimes A = \begin{pmatrix} \overline{a}_{11}A & \overline{a}_{12}A & \cdots & \overline{a}_{1\overline{n}}A \\ \overline{a}_{21}A & \overline{a}_{22}A & \cdots & \overline{a}_{2\overline{n}}A \\ \vdots & \vdots & & \vdots \\ \overline{a}_{\overline{m}1}A & \overline{a}_{\overline{m}2}A & \cdots & \overline{a}_{\overline{m}\,\overline{n}}A \end{pmatrix}. \tag{6.3}$$

The Kronecker product can also be expressed in terms of the columns of $A = (a_1, \ldots, a_n)$ and $\overline{A} = (\overline{a}_1, \ldots, \overline{a}_{\overline{n}})$ as

$$\overline{A} \otimes A = (\overline{a}_1 \otimes a_1, \ldots, \overline{a}_1 \otimes a_n, \overline{a}_2 \otimes a_1, \ldots, \overline{a}_{\overline{n}} \otimes a_1, \ldots, \overline{a}_{\overline{n}} \otimes a_n). \tag{6.4}$$

The Kronecker product is very useful when dealing with discretizations of 2-D problems, and it enjoys a number of properties that come in handy in this connection, such as the relations

$$(A \otimes B)^T = A^T \otimes B^T \tag{6.5}$$

and

$$(A \otimes B)\,(C \otimes D) = (A\,C) \otimes (B\,D). \tag{6.6}$$

Some norm relations for Kronecker products are

$$\|A \otimes B\|_p = \|A\|_p\,\|B\|_p, \qquad p = 1, 2, \infty, \mathrm{F}.$$

More details about Kronecker products can be found in [18].

Along with the Kronecker product, we need the "vec" notation. If X is an $m \times n$ matrix with column partitioning $X = (x_1, \ldots, x_n)$, then we define the vector $\mathrm{vec}(X)$ of length mn as

$$\mathrm{vec}(X) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{6.7}$$

The Kronecker product and the "vec" notation are connected via the following important relation, which we shall make extensive use of

$$(\overline{A} \otimes A)\,\mathrm{vec}(X) = \mathrm{vec}(A\,X\,\overline{A}^T). \tag{6.8}$$

One of the many amazing properties of the Kronecker product $\overline{A} \otimes A$ is that its singular value decomposition is entirely made up of the SVDs of the two matrices A and $\overline{A}$ (and the same is true for the eigenvalue decomposition). Specifically, if both A and $\overline{A}$ are square, and if their SVDs are given by

$$A = U\,\Sigma\,V^T \qquad \text{and} \qquad \overline{A} = \overline{U}\,\overline{\Sigma}\,\overline{V}^T,$$

then it follows from (6.4) and (6.6) that

$$\begin{aligned} \overline{A} \otimes A &= (\overline{U} \otimes U)\,(\overline{\Sigma} \otimes \Sigma)\,(\overline{V} \otimes V)^T \\ &= \sum_{i=1}^{\overline{n}} \sum_{j=1}^{n} (\overline{u}_i \otimes u_j)\,(\overline{\sigma}_i\,\sigma_j)\,(\overline{v}_i \otimes v_j)^T \end{aligned}$$

which, except for the ordering of the singular values and vectors, constitutes the SVD of $\overline{A} \otimes A$. We see that the $n\overline{n}$ singular values of $\overline{A} \otimes A$ consist of all the products of the singular values of A and $\overline{A}$, while the left and right singular vectors of $\overline{A} \otimes A$ consist of all the Kronecker products of the left and right singular vectors of A and $\overline{A}$. Hence, to work with the SVD of $\overline{A} \otimes A$, one merely needs to compute the two SVDs of A and $\overline{A}$. As we shall see below, similar computations savings always can be made when working with Kronecker products.

6.3. Discretization when Variables Separate

After the above venture into the world of Kronecker products, we now turn our attention to the 2-D convolution problems in (6.2). To discretize this 2-D deconvolution problem, we use (for simplicity) the midpoint quadrature rule. First we

apply this rule to the "inner" integral at the quadrature points y'_ℓ, $\ell = 1, \ldots, n$, thus arriving at the function ϕ:

$$\int_0^1 \omega(y - y')\, f(x', y')\, dy' \simeq n^{-1} \sum_{\ell=1}^{n} \omega(y - y'_\ell)\, \widetilde{f}(x', y'_\ell) = \phi(x', y).$$

Next we apply the midpoint quadrature rule to the "outer" integral at the quadrature points x'_k, $k = 1, \ldots, n$, thus arriving at the function ψ:

$$\int_0^1 \kappa(x - x')\, \phi(x', y)\, dx' \simeq n^{-1} \sum_{k=1}^{n} \kappa(x - x'_k)\, \phi(x'_k, y) = \psi(x, y).$$

Then we use collocation to obtain a system of linear equations, and we use as many collocation points as quadrature points to make the system square,

$$\psi(x_i, y_j) = g(x_i, y_j), \qquad i, j = 1, \ldots, n.$$

Notice that we have used the same number of quadrature and collocation points in each variable – this is not required, but makes our exposition simpler.

We shall now derive the equations for the system of linear equations corresponding to the above discretization scheme. First we need to introduce the four $n \times n$ matrices A, $\overline{A}$, F, and G with elements given by

$$A_{ik} = n^{-1} \kappa(x_i - x'_k), \qquad \overline{A}_{j\ell} = n^{-1} \omega(y_j - y'_\ell)$$

$$F_{k\ell} = \widetilde{f}(x'_k, y'_\ell), \qquad G_{ij} = g(x_i, y_j),$$

where all indices are in the range $1, \ldots, n$. note that A and $\overline{A}$ consist of samples of the functions κ and ω, respectively, while F and G consist of samples of the approximate function $\widetilde{f}$ and the right-hand side g, respectively. We can now define an $n \times n$ matrix Φ that corresponds to the "inner" integration, with elements given by

$$\Phi_{kj} = \phi(x'_k, y_j) = n^{-1} \sum_{\ell=1}^{n} \omega(y_j - y'_\ell)\, \widetilde{f}(x'_k, y'_\ell), \qquad j, k = 1, \ldots, n,$$

and by studying the indices of the above expression it follows that Φ can be written as $\Phi = F\, \overline{A}^T$, where F and $\overline{A}$ are defined above. Similarly, we can define an $n \times n$ matrix Ψ that corresponds to the "outer" integration, with elements given by

$$\Psi_{ij} = \psi(x_i, y_j) = n^{-1} \sum_{k=1}^{n} \kappa(x_i - x'_k)\, \phi(x'_k, y_j), \qquad i, j = 1, \ldots, n,$$

and this matrix can be written as $\Psi = A\,\Phi$, where A is also defined above. Collocation then corresponds to the requirement that $\Psi = G$, and in this way we have arrived at the following linear relations between the solution and the right-hand side:

$$A\,F\,\overline{A}^T = G \qquad \Longleftrightarrow \qquad \left(\overline{A} \otimes A\right)\,\mathrm{vec}(F) = \mathrm{vec}(G). \tag{6.9}$$

The rightmost equation is a system of n^2 linear equations in n^2 unknowns, and corresponds to the linear system $A\,x = b$ in the 1-D case. This system is useful for theoretical considerations, while the leftmost expression in (6.9) is more suited for numerical computations.

6.4. Digital Image Reconstruction

Digital *image restoration* – in which a noisy, blurred image is restored on the basis of a mathematical model of the blurring process[1] – is probably the most well-known example of a 2-D deconvolution problem. A recent survey of this topic, including a discussion of many practical aspects, can be found in [1]; here we limit out discussion to those aspects of image restoration that are directly connected with numerical inversion algorithms.

A digital gray-scale image is really an $m \times n$ matrix F whose nonnegative entries represent light intensity, and today's digital images are usually recorded by means of a CCD camera that records the number of photons hitting a 2-D array of sensors. Hence, we can consider the image F as a sampled version of a continuous 2-D function $f(x', y')$ representing the exact image, and this models fits with the above-mentioned discretization scheme based on the midpoint quadrature rule. Digital images are usually stored as integers using 8 or 10 bits, but in connection with our deconvolution algorithms we can consider the matrix elements as real numbers.

The noise in a digital image typically consists of photon noise (usually with a Poisson distribution) arising from the discrete nature of the light, plus amplifier noise (usually colored Gaussian noise) from the electronic components that convert the recorded light intensity into a digital signal.

There are many sources of blur, and some of the most important are discussed here along with their point spread functions, i.e., the kernels K that model the blurring of the underlying exact image f. We focus on spatially invariant point spread functions whose kernels are separable and of convolution type, i.e., $K(x, y, x', y') = \kappa(x - x')\,\omega(y - y')$. This means that the blurring is identical in all parts of the image, and separates into pure horizontal and pure vertical components.

[1] Opposed to *image enhancement* where the image is manipulated on a heuristic basis.

Motion blur arises when the camera moves while the image is recorded, or when the object itself moves while the camera is steady. As long as the exposure time is small, we can consider the motion of the camera or the object as linear, and if the motion is horizontal then the corresponding point spread function takes the form

$$K(x,y,x',y') = h_L(x-x') = \begin{cases} \frac{1}{2L}, & |x-x'| \le L \\ 0, & \text{else} \end{cases} \tag{6.10}$$

where the parameter L characterizes the smearing of the image. Similarly, if the motion is vertical then the point spread function is given by

$$K(x,y,x',y') = h_L(y-y') = \begin{cases} \frac{1}{2L}, & |y-y'| \le L \\ 0, & \text{else} \end{cases}$$

where L plays the same role as before. The corresponding discretizations $\overline{A} \otimes A$ of these two point spread functions are given by $B_\ell \otimes I$ and $I \otimes B_\ell$, respectively, where I is the identity matrix and B_ℓ is a banded Toeplitz matrix of appropriate dimensions with elements given by

$$(B_\ell)_{ij} = \begin{cases} \frac{1}{2\ell}, & |i-j| \le \ell \\ 0, & \text{else} \end{cases} \tag{6.11}$$

where the parameter ℓ depends on both L and the discretization.

Out-of-focus blur arises, of course, when the lens is out of focus, i.e. when the focal point of the lens does not match with the light-sensitive CCD. The point spread function for out-of-focus blur is

$$K(x,y,x',y') = \begin{cases} \frac{1}{\pi R^2}, & \sqrt{(x-x')^2+(y-y')^2} \le R \\ 0, & \text{else} \end{cases} \tag{6.12}$$

where the parameter R characterizes the defocus. This function, however, does not separate in the variables $x-x'$ and $y-y'$, and it is therefore customary to use an alternative point spread function whose variables do separate, namely,

$$K(x,y,x',y') = h_R(x-x')\, h_R(y-y'),$$

where the function h_R is identical to the horizontal/vertical motion blur function (6.10) with L replaced by R. The corresponding discretization of the latter point spread function is given by $\overline{A} \otimes A = B_\ell \otimes B_\ell$.

Atmospheric turbulence blur arises, e.g., in remote sensing and astronomical imaging, and is due to long-term exposure through the atmosphere where turbulence

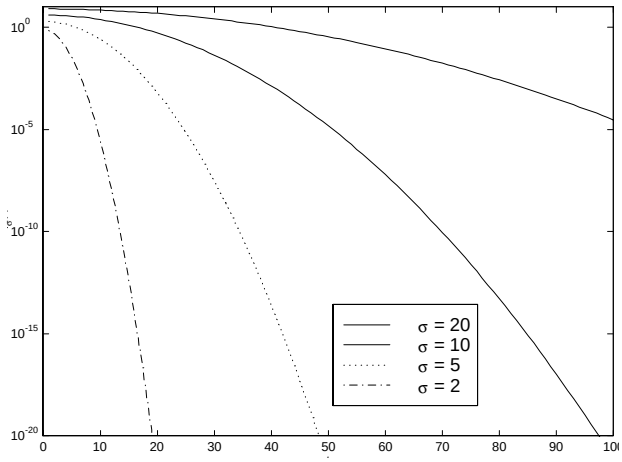

Figure 6.1: The decay of the function $\eta_\sigma(t)$ in the atmospheric turbulence blur model.

in the atmosphere gives rise to random variations in the refractive index. For many practical purposes, the blurring can be modelled by a Gaussian point spread function

$$\begin{aligned} K(x,y,x',y') &= \frac{1}{2\pi\sigma\overline{\sigma}} \exp\left(-\frac{1}{2}\left(\frac{x-x'}{\overline{\sigma}}\right)^2 - \frac{1}{2}\exp\left(\frac{y-y'}{\sigma}\right)^2\right) \\ &= \eta_{\overline{\sigma}}(x-x')\,\eta_\sigma(y-y'), \end{aligned} \tag{6.13}$$

where the function η_σ is given by

$$\eta_\sigma(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{t}{\sigma}\right)^2\right),$$

and where $\overline{\sigma}$ and σ are two constants that characterize the blurring in the x and y directions, respectively. If we introduce a Toeplitz matrix T_σ with elements given by

$$(T_\sigma)_{ij} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{i-j}{\sigma}\right)^2\right),$$

then the discretization of the atmospheric turbulence point spread function takes the form $\overline{A} \otimes A = T_{\overline{\sigma}} \otimes T_\sigma$, where the two Toeplitz matrices should have the appropriate dimensions. Depending on the dimensions and the value of σ, the matrix T_σ may be a full matrix or an effectively banded matrix. Figure 6.1 illustrates this point.

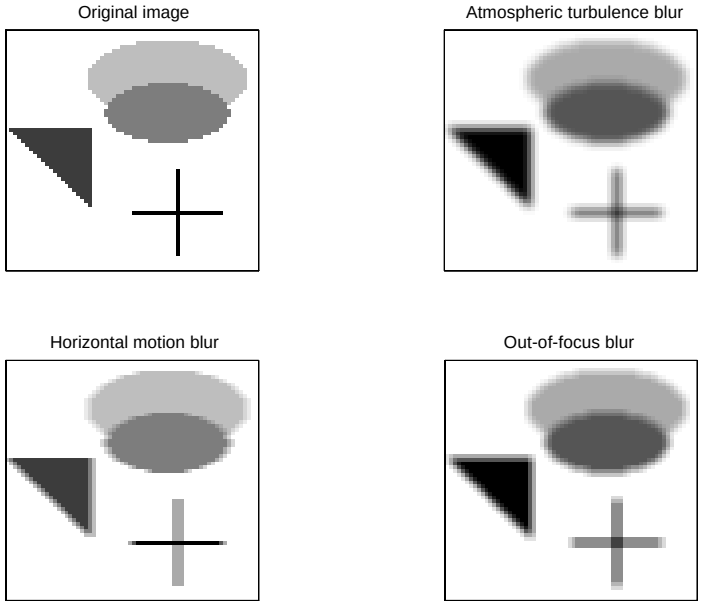

Figure 6.2: An exact image, and three blurred versions of the same image.

The above three blurring models are illustrated in Fig. 6.2 with an example using a simple test image (which is generated by means of the function blur in REGULARIZATION TOOLS). Notice the distinctly different nature of atmospheric turbulence blur and out-of-focus blur.

We remark that in image processing, where the "signals" are always nonnegative, it is common [1] to measure the power of an image by first subtracting the image's mean value from all pixels; i.e., if μ_F is the mean value of all the elements of F,

$$\mu_F = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij},$$

then the power of F is given by

$$P(F) = \sum_{i=1}^{m} \sum_{j=1}^{n} (f_{ij} - \mu_F)^2 = \|F - \mu_F \Xi\|_{\mathbf{F}}^2,$$

in which Ξ is an $m \times n$ matrix of all ones. Along this line, if $G_{\mathbf{pure}} = A\,F\,\overline{A}^T$ is the "pure" blurred image and E is additive Gaussian noise with zero mean and standard

deviation σ_{noise}, then the *blurred signal-to-noise ratio* (BSNR) of the blurred noisy image $G = G_{\text{pure}} + E$ is given by

$$\text{BSNR} = 10 \log_{10} \frac{P(G_{\text{pure}})}{P(E)} = 10 \log_{10} \frac{\|G - \mu_{G_{\text{pure}}} \Xi\|_{\text{F}}^2}{m\, n\, \sigma_{\text{noise}}^2}. \tag{6.14}$$

Similarly, the improvement of the restored image F_{res} over the noisy blurred image G is measured by the *improvement in SNR* (ISNR), defined by

$$\text{ISNR} = 10 \log_{10} \frac{\|F - G\|_{\text{F}}^2}{\|F - F_{\text{res}}\|_{\text{F}}^2}. \tag{6.15}$$

6.5. Regularization with Kronecker Products

In this section we express the TSVD and Tikhonov solutions to the discretized 2-D problem (6.9) in terms of Kronecker products, thus illustrating how all computations with the large $n^2 \times n^2$ matrix $\overline{A} \otimes A$ can be avoided.

We begin with the TSVD solution which is defined in terms of the SVD of $\overline{A} \otimes A$. Let $\text{vec}(F_k)$ denote the TSVD solution, and F_k the corresponding 2-D version. Here, k is the TSVD truncation parameter, i.e., the number of singular values of $\overline{A} \otimes A$ to be included in the regularized solution. Then $\text{vec}(F_k)$ is given by

$$\text{vec}(F_k) = \sum_{i,j} \frac{(\overline{u}_j \otimes u_i)^T \text{vec}(G)}{\overline{\sigma}_j\, \sigma_i} (\overline{v}_j \otimes v_i),$$

where the summation is over the k largest quantities $\overline{\sigma}_j\, \sigma_i$. Using the relation (6.8) we arrive at the expression

$$F_k = \sum_{k} \frac{u_i^T G\, \overline{u}_j}{\sigma_i\, \overline{\sigma}_j} v_i\, \overline{v}_j^T \tag{6.16}$$

with the same summation as above. Note that we express the 2-D TSVD solution in terms of rank-one matrices $v_i\, \overline{v}_j$. Equation (6.16) is very convenient for computing 2-D TSVD solutions in terms of the two SVDs of A and $\overline{A}$.

Alternatively we can define a 2-D TSVD solution $F_{k,\overline{k}}$ by using two truncation parameters k and $\overline{k}$ associated with the SVDs of A and $\overline{A}$, respectively. This solution is given by the expression

$$F_{k,\overline{k}} = \sum_{i=1}^{k} \sum_{j=1}^{\overline{k}} \frac{u_i^T G\, \overline{u}_j}{\sigma_i\, \overline{\sigma}_j} v_i\, \overline{v}_j^T. \tag{6.17}$$

We remark in passing that if the 2-D integral equation collapses into a 1-D problem in the variables x and x', then $\overline{A} = 1$, i.e., a 1×1 matrix with SVD $\overline{u}_1 = \overline{\sigma}_1 = \overline{v}_1 = 1$, and (6.16) and (6.17) reduce to the standard expression (4.1) for the 1-D TSVD solution.

Consider now the Tikhonov solution. The underlying formulations of Tikhonov regularization are now

$$\min \left\{ \|(\overline{A} \otimes A)\,\mathrm{vec}(F) - \mathrm{vec}(G)\|_2^2 + \lambda^2 \|\mathrm{vec}(F)\|_2^2 \right\}$$

and

$$\min \left\{ \|A\,F\,\overline{A}^T - G\|_F^2 + \lambda^2 \|F\|_{\mathrm{F}}^2 \right\},$$

and the associated least squares problem is

$$\min \left\| \begin{pmatrix} \overline{A} \otimes A \\ \lambda\, I_{n^2} \end{pmatrix} \mathrm{vec}(F) - \begin{pmatrix} \mathrm{vec}(G) \\ 0 \end{pmatrix} \right\|_2,$$

where I_{n^2} is the identity matrix of order n^2. If $\mathrm{vec}(F_\lambda)$ denotes the Tikhonov solution, where F_λ is the corresponding 2-D version, then $\mathrm{vec}(F_\lambda)$ can also be expressed in terms of the SVD of $\overline{A} \otimes A$. We recognize that the 2-D filter factors are given by

$$f_{ij} = \frac{\sigma_i^2\,\overline{\sigma}_j^2}{\sigma_i^2\,\overline{\sigma}_j^2 + \lambda^2}, \qquad i, j = 1, \ldots, n,$$

and the expression for the 2-D Tikhonov solution becomes

$$F_\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}\, \frac{u_i^T G\,\overline{u}_j}{\sigma_i\,\overline{\sigma}_j}\, v_i\,\overline{v}_j^T. \tag{6.18}$$

Both algorithms described above require the computation of the SVD of the two matrices A and $\overline{A}$, and the algorithms are therefore of complexity $\mathcal{O}(n^3)$. In the computation of the 2-D Tikhonov solution, it is possible to replace one of the SVDs (of either A or $\overline{A}$) with a bidiagonalization of the matrix, as described in [7]; the complexity of the algorithm is the same. At this moment, it is not clear whether it is possible to derive an efficient 2-D Tikhonov regularization algorithm based on two bidiagonalizations of A and $\overline{A}$.

6.6. Kronecker Products in Iterative Methods

We can also apply the iterative methods to the discretized 2-D problem in (6.9). For example, one step of Landweber's method becomes

$$\mathrm{vec}(F^{(k)}) = \mathrm{vec}(F^{(k-1)}) + \omega\,(\overline{A} \otimes A)^T \left(\mathrm{vec}(G) - (\overline{A} \otimes A)\,\mathrm{vec}(F^{(k-1)}) \right)$$

and by means of (6.5) and (6.8) we can rewrite this formula in the following, more convenient form

$$\begin{aligned} R^{(k-1)} &= G - A\,F^{(k-1)}\overline{A}^T \\ F^{(k)} &= F^{(k-1)} + \omega\,A^T R^{(k-1)}\overline{A}, \end{aligned}$$

where $R^{(k-1)}$ is the residual matrix corresponding to the iteration matrix $F^{(k-1)}$. We see that the basic "building blocks" in this algorithm are the multiplications of the iteration matrix $F^{(k-1)}$ and the residual matrix $R^{(k-1)}$ from the left and right by A and $\overline{A}$.

Similarly, we can rewrite the 2-D version of the CGLS algorithm in terms of the matrices A and $\overline{A}$. To do this, we use the fact that $\|\mathrm{vec}(X)\|_2 = \|X\|_F$, and the 2-D CGLS algorithm takes the form

$$\begin{aligned} \alpha_k &= \|A^T R^{(k-1)}\overline{A}\|_{\mathrm{F}}^2 \,/\, \|A\,D^{(k-1)}\overline{A}^T\|_{\mathrm{F}}^2 \\ F^{(k)} &= F^{(k-1)} + \alpha_k\,D^{(k-1)} \\ R^{(k)} &= R^{(k-1)} - \alpha_k\,A\,D^{(k-1)}\overline{A}^T \\ \beta_k &= \|A^T R^{(k)}\overline{A}\|_{\mathrm{F}}^2 \,/\, \|A^T R^{(k-1)}\overline{A}\|_{\mathrm{F}}^2 \\ D^{(k)} &= A^T R^{(k)}\overline{A} + \beta_k\,D^{(k-1)} \end{aligned}$$

with starting conditions $F^{(0)}$ = initial image, $R^{(0)} = G - A\,F^{(0)}\overline{A}^T$, and $D^{(0)} = A^T R^{(0)}\overline{A}$. The advantage of this formulation is that we work solely with the two-dimensional arrays that represent the underlying 2-D functions.

6.7. 2-D FFTs and Convolution

When both $\overline{A}$ and A are Toeplitz matrices, we can use the techniques from the previous chapter to derive an FFT-based algorithm for computing the matrix product $A\,X\,\overline{A}$, where X is any matrix that conforms with the other two matrices.

Our discussion takes its basis in the 2-D convolution of two general 2-D periodic discrete signals, represented by the two $m \times n$ matrices F and H. Then the convolution of F and H is another 2-D periodic signal which is represented by the matrix G whose elements are given by

$$g_{ij} = \sum_{k=1}^{m}\sum_{\ell=1}^{n} f_{k\ell}\,h_{i-k,j-\ell},$$

where the two subscripts of h are taken modulo m and n, respectively.

The matrix G can be computed in less than $2mn(m+n)$ flops by means of the 2-D FFT. We use the same notation as in the previous chapter, in which we write the DFT of an n-vector x as $\mathrm{DFT}(x) = F_n\,x$, where F_n is a complex symmetric matrix. Then the 2-D FFT of an $m \times n$ matrix X is defined as follows

$$\mathrm{DFT2}(X) = F_m\,X\,F_n. \tag{6.19}$$

In this expression, we could add transposition to the symmetric matrix F_m in order to make the expression appear like the previous 2-D expressions, but we shall refrain from this. We also define the 2-D inverse DFT as $\mathrm{IDFT2}(X) = F_m^{-1} X\,F_n^{-1}$. We can now express G as

$$\begin{aligned} G &= F_m^{-1}\left((F_m\,F\,F_n)\,.*(F_m\,H\,F_n)\right)F_n^{-1} \\ &= \mathrm{IDFT2}\left(\mathrm{DFT2}(F)\,.*\mathrm{DFT2}(H)\right). \end{aligned}$$

The 2-D DFT $\mathrm{DFT2}(X)$ requires the computation of the ordinary DFT of each of the n columns of X, followed by the computation of the ordinary DFT of each of the m rows of the matrix $F_m\,X$. Ignoring that data are real (because the intermediate results $F_m\,X$ is complex), all the $m+n$ DFTs in a single 2-D DFT can be computed by means of the FFT in $n\,(5\,m\,\log_2 m)+m\,(5\,n\,\log_2 n) = 5mn\,(\log_2 m+\log_2 n)$ flops. This is also the flop count for the 2-D inverse DFT. Hence, provided that $\mathrm{DFT2}(H)$ has been precomputed, G can be computed in $10mn\,(\log_2 m + \log_2 n)$ flops, which simplifies to $20\,n^2\,\log_2 n$ flops when $m = n$.

The special case of convolution that we focus on in this chapter is when the kernel K in the integral equation (6.1) separates into a product of two functions as in (6.2). In connection with discrete data, this corresponds to a situation where the matrix H is a rank-one matrix given by the outer product of two vectors v and w of length m and n, respectively, i.e.

$$H = v\,w^T.$$

The corresponding expression for G in this case becomes

$$G = C_v\,F\,C_w^T, \tag{6.20}$$

where C_v and C_w are two circulant matrices defined by setting their first columns equal to v and w, respectively. In this case, the 2-D DFT of H is particularly easy to compute and represent, since

$$\mathrm{DFT2}(H) = F_m\,H\,F_n = F_m\,v\,w^T F_n^T = (F_m\,v)\,(F_n\,w)^T = \mathrm{DFT}(v)\,\mathrm{DFT}(w)^T,$$

i.e., we merely need to compute and store the DFTs of the two vectors v and w. This leads to an efficient algorithm for computing the matrix product in (6.20):

$$G = \mathrm{IDFT2}\left(\mathrm{DFT2}(F) .* \left(\mathrm{DFT}(v)\,\mathrm{DFT}(w)^T\right)\right). \tag{6.21}$$

Note that the element-wise matrix multiplication should be computed without forming the out product of the two DFTs explicitly.

The above relations can now be used to efficiently compute the matrix product $A\,F\,\overline{A}^T$ when A and $\overline{A}$ are Toeplitz matrices. The key idea is again to imbed the two Toeplitz matrices in two larger circulant matrices C and $\overline{C}$ as described in the previous chapter, and similarly augment X with zero blocks such that the augmented matrix

$$\widetilde{X} = \begin{pmatrix} X & 0 \\ 0 & 0 \end{pmatrix}$$

conforms with C and $\overline{C}$. Then the desired matrix product is the upper left submatrix in

$$\widetilde{Z} = C\,\widetilde{X}\,\overline{C}^T = \begin{pmatrix} A\,X\,\overline{A}^T & \widetilde{Z}_{12} \\ \widetilde{Z}_{21} & \widetilde{Z}_{22} \end{pmatrix},$$

while the other three submatrices are discarded. The multiplications with C and $\overline{C}$ could be computed by means of the fast algorithms from the previous chapter, by first working on the columns of $\widetilde{X}$ and then on the rows of $C\,\widetilde{X}$, but we can avoid the many DFT-IDFT pairs by instead using Eq. (6.21). The complete amount of work in this approach is approximately $2 \cdot 5 \cdot 2m\,2n\,(\log_2(2m) + \log_2(2n)) \simeq 40mn\,(\log_2 m + \log_2 n)$ flops – which should be compared with the $2mn(m+n)$ flops in a standard matrix-matrix multiplication algorithm for computing $A\,X\,\overline{A}^T$. And if $m = n$ then the two numbers are $80n^2 \log_2 n$ and $4n^3$.

Bibliography

[1] M. R. Banham and A. K. Katsaggelos, *Digital image restoration*, IEEE Signal Proc. Magazine, 14 (1997), pp. 24–41.

[2] M. Bertero, P. Brianzi, and E. R. Pike, *Super-resolution in confocal scanning microscopy*, Inverse Problems, 3 (1987), pp. 195–212.

[3] A. W. Bojanczyk, R. P. Brent, and F. R. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 210–221.

[4] W. L. Briggs and V. E. Henson, *The DFT. An Owner's Manual for the Discrete Fourier Transform*, SIAM, Philadelphia, 1995.

[5] L. Eldén, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 18 (1977), pp. 134–145.

[6] L. Eldén, *An efficient algorithm for the regularization of ill-conditioned least squares problems with triangular Toeplitz matrices*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 229–236.

[7] L. Eldén and I. Skoglund, *Algorithms for the regularization of ill-conditioned least squares problems with tensor product structure, and applications to space-invariant image restoration*, Report LiTH-MAT-R-1982-48, Dept. of Mathematics, Linköping University, Sweden, 1982.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3. Ed., Johns Hopkins University Press, 1996.

[9] P. C. Hansen, *Truncated SVD solutions to discrete ill-posed problems with ill-determined numerical rank*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 503–518.

[10] P. C. Hansen, *Regularization Tools: a Matlab package for analysis and solution of discrete ill-posed problems*, Numer. Algorithms, 6 (1994), pp. 1–35. Software and manual is avaiable from Netlib as well as from the author's home page.

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

[12] P. C. Hansen and K. Mosegaard, *Piecewise polynomial solutions without a priori break points*, Numer. Algo., 3 (1996), pp. 513–524.

[13] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[14] M. Kilmer and G. W. Stewart, *Iterative regularization and MINRES*, Report CMSC TR-3949, Dept. of Computer Science, Univ. of Maryland, 1999; to appear in SIAM J Matrix Anal. Appl.

[15] H. Park and L. Eldén, *Stability analysis and fast algorithms for triangularization of Toeplitz matrices*, Numer. Math., 76 (1997), pp. 383–402.

[16] 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.

[17] 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, 51 (1963), pp. 501–504.

[18] C. F. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.

[19] G. M. Wing and J. D. Zahrt, *A Primer on Integral Equations of the First Kind*, SIAM, Philadelphia, 1991.