*Numerical Algorithms* **29:** 323–378, 2002.
© 2002 *Kluwer Academic Publishers. Printed in the Netherlands.*

# Deconvolution and regularization with Toeplitz matrices

Per Christian Hansen

*Department of Mathematical Modelling, Technical University of Denmark, Building 321, DK-2800 Lyngby, Denmark*
E-mail: pch@imm.dtu.dk

Received 2 August 2000; revised 5 November 2001

By deconvolution we mean the solution of a linear first-kind integral equation with a convolution-type kernel, i.e., a kernel that depends only on the difference between the two independent variables. Deconvolution problems are special cases of linear first-kind Fredholm integral equations, whose treatment requires the use of regularization methods. The corresponding computational problem takes the form of structured matrix problem with a Toeplitz or block Toeplitz coefficient matrix. The aim of this paper is to present a tutorial survey of numerical algorithms 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. Among other things, we present direct methods for regularization with Toeplitz matrices, and we show how Toeplitz matrix–vector products are computed by means of FFT, being useful in iterative methods. We also introduce the Kronecker product and show how it is used in the discretization and solution of 2-D deconvolution problems whose variables separate.

**Keywords:** deconvolution, regularization, Toeplitz matrix, Kronecker product, SVD analysis, image deblurring

**AMS subject classification:** 65F30

## 1. Introduction to deconvolution problems

The main purpose of this survey paper is to present modern *computational methods* for treating linear deconvolution problems, along with some of the underlying theory. Discretization of these problems leads to structured matrix problems with a Toeplitz or block Toeplitz coefficient matrix. The main focus is on numerical aspects, and we illustrate how mathematics helps to derive efficient numerical deconvolution algorithms that exploit the Toeplitz structure of the matrix problems. Our goal is to demonstrate that fast deconvolution can be performed without the artificial periodization of the data that is common in many popular FFT-based algorithms. The present paper can be viewed as an addendum to the monograph [20] where Toeplitz regularization algorithms are only mentioned briefly.

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 this introduction, we shall briefly describe both meaning of the term.

Before we present the algorithms, we make a detour in section 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 sections 3–6. We start in section 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 section 4 we discuss general regularization algorithms for computing stabilized solutions to the discretized linear systems, with no emphasis on matrix structure.

In the remaining sections we turn to numerical methods that exploit matrix structure. In section 5 we discuss various direct and iterative methods for one-dimensional problems, and show how we can compute the numerical solutions accurately and efficiently. The emphasis here is on numerically stable methods based on orthogonal transformations and FFT, and we also consider the use of stabilized hyperbolic rotations. Finally, in section 6 we turn to 2-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. The emphasis here is on Kronecker products, their properties, and their use in direct and iterative methods. In addition, we briefly discuss the use of the 2-D FFT algorithm.

Throughout the paper, the theory and the algorithms are illustrated with representative problems from gravity surveying and image deblurring.

We do not attempt to survey the vast area of deconvolution and its applications in science and engineering. The literature on this subject is rich, and presentations at all levels of sophistication can be found. The updated collection of papers in [23] provides both background material and surveys of applications, and is a good starting point.

Some acquaintance with numerical linear algebra is necessary; the most complete reference is the book [15] that discusses all state-of-the-art techniques, classical as well as modern. The monograph [5] contains a wealth of material on numerical least squares methods. We also wish to mention the recent monograph [24] devoted to theory and algorithms for structured matrices, including a wealth of material on Toeplitz matrices.

In our presentation we shall frequently use the elementwise product (Hadamard product), denoted by $\odot$, of two conforming matrices or vectors. Similarly we use the symbol $\oslash$ to denote the elementwise division of two conforming matrices or vectors. These operations are identical to the Matlab operations "`.*`" and "`./`", respectively.

We shall also make reference to the Matlab package REGULARIZATION TOOLS [19,21] from time to time. More details about the underlying theory that is briefly presented in this paper can be found in the "deconvolution primer" [35], while a recent survey of numerical methods for general linear inverse problems is given in the monograph [20].

### 1.1. Deconvolution in digital signal processing

Given two discrete (digital) real 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

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

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 convolution is used to represent many useful operations in digital signal processing. For example, if $h$ is the input signal then the output of a FIR filter with $n$ nonzero filter coefficients $f_0, \ldots, f_{n-1}$ is given by

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

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

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

If the two discrete 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

$$g_i = \sum_{j=0}^{N-1} f_j h_{(i-j) \bmod N}, \quad i = 0, 1, \ldots, N-1.$$

Note that the subscript of $h$ is taken modulo $N$. The discrete Fourier transform (DFT) of $g$ is defined as the sequence

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

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

$$\mathrm{DFT}(g) = \mathrm{DFT}(f) \odot \mathrm{DFT}(h), \tag{1}$$

i.e., the vector DFT($g$) equals the element-wise product of the two vectors DFT($f$) and DFT($h$). Consequently, $g = \text{IDFT}(\text{DFT}(f) \odot \text{DFT}(h))$, 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 $\text{O}(N \log_2 N)$ operations. A very thorough discussion of the DFT (and the FFT) is presented in [7].

*Deconvolution* is then defined as the process of computing the signal $f$ given the other two signals $g$ and $h$. For periodic discrete signals, equation (1) leads to a simple expression for computing $f$, considered as an $N$-vector:

$$f = \text{IDFT}\big(\text{DFT}(g) \oslash \text{DFT}(h)\big). \tag{2}$$

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 $\hat{f}(\omega) = \hat{g}(\omega)/\hat{h}(\omega)$, but there is no similar simple computational scheme for computing $f$ – although the FFT and equation (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 that resemble the classical deconvolution problem in signal processing, and these problems are also often called deconvolution problems. Given two real functions $f$ and $h$, the *general convolution* operation takes the generic form

$$g(s) = \int_0^1 h(s-t)\, f(t)\, \mathrm{d}t, \quad 0 \leqslant s \leqslant 1, \tag{3}$$

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 to focus our presentation on the principles of deconvolution we omit these problems here.

The general problem of deconvolution is now to determine either $f$ or $h$, given the other two quantities, and 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 $s_1, s_2, \ldots, s_m$:

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

Here, $e_i$ denotes the measurement noise associated with the $i$th 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 gravity surveying (presented as a geomagnetic prospecting problem in [35, p. 17]). Assume that a one-dimensional horizontal mass distribution lies at depth $d$ below the surface – see figure 1 for the geometry and the location of the $s$ and $t$ axes. From measurements of the vertical component of the gravity field, denoted $g(s)$, at the surface, we want to compute the mass

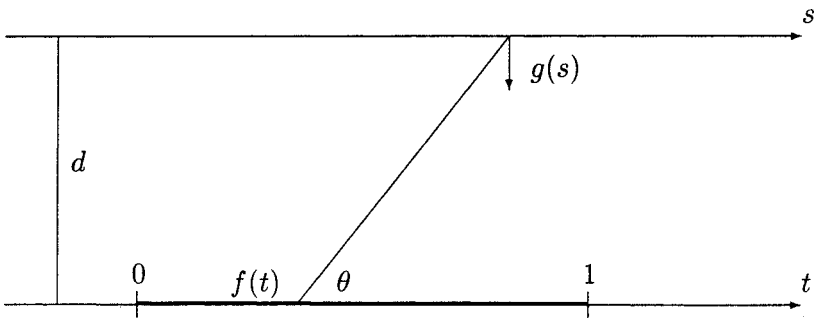


Figure 1. The geometry of the gravity surveying model problem. The measured signal $g(s)$ is the vertical component of the gravity field due to a 1-D mass distribution $f(t)$ at depth $d$.

distribution, denoted $f(t)$, along the $t$ axis. The contribution to $g$ from an infinitesimal part $\mathrm{d}t$ of the mass distribution at $t$ is given by

$$\mathrm{d}g = \frac{\sin\theta}{r^2} f(t)\,\mathrm{d}t,$$

where the angle $\theta$ is shown in figure 1, and the distance between the two points on the $s$ and $t$ axes is given by $r = \sqrt{d^2 + (s-t)^2}$. Using that $\sin\theta = d/r$, we get

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

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

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

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

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

where $\sigma$ 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.

The inverse heat equation [12] 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

$$\int_0^s h(s-t)\,f(t)\,\mathrm{d}t = g(s) \tag{4}$$

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 $\kappa$ 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. The Hubble Space Telescope is perhaps the most famous example of the need for image deblurring [1].

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

$$\int_0^1 K(s,t)\,f(t)\,\mathrm{d}t = g(s), \quad 0 \leqslant s \leqslant 1, \tag{5}$$

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 section are merely special cases of (5) 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 equation (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 (5) 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 "desmoothing process," so to speak.

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

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

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

$$g(s) = \int_0^1 K(s,t)\sin(2\pi pt)\,\mathrm{d}t, \quad p = 1, 2, \ldots,$$

then the Riemann–Lebesgue lemma[1] states that

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

In other words, the higher the frequency of $f$, the more $g$ is damped – and this is true independently of the kernel $K$; see figure 2 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 unavoidable data and rounding errors make

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

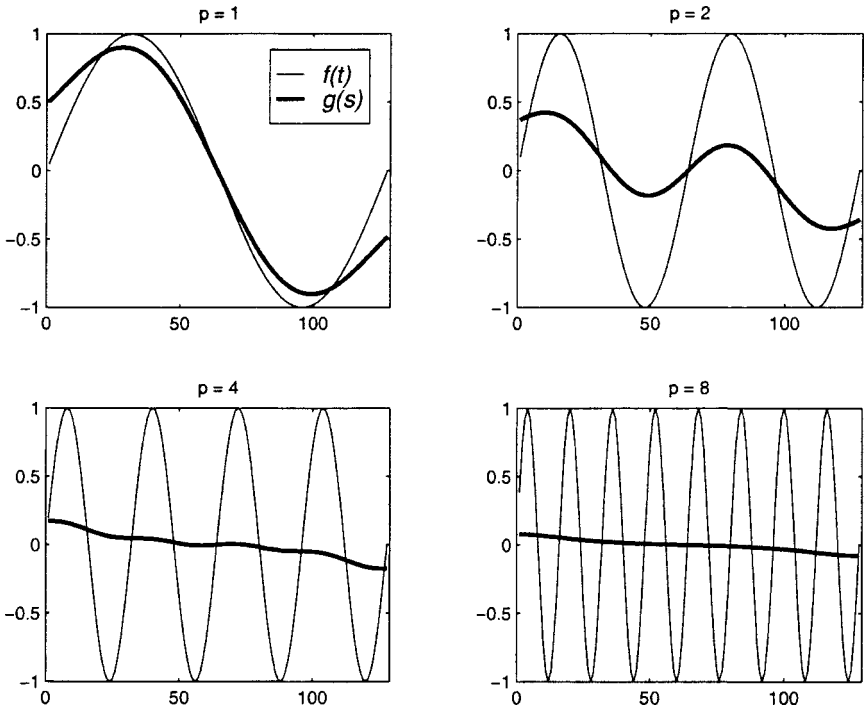


Figure 2. Illustration of the Riemann–Lebesgue lemma for the gravity surveying problem, with $f(t) = \sin(2\pi pt)$, $g(s) = \int_0^1 h(s-t)f(t)\,\mathrm{d}t$, and $p = 1, 2, 4$ and 8.

the practical inversion process a very unstable process indeed, and in practice 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 (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 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 $\sigma_{\text{noise}}$. Then the DFT of $\widetilde{g}$ is given by

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

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

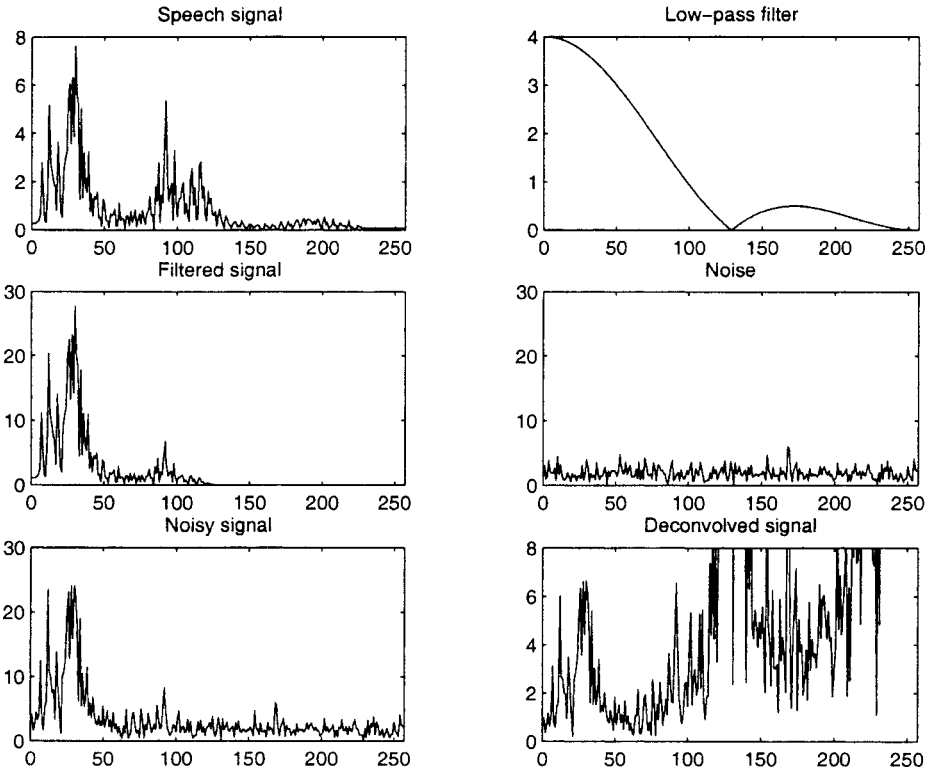


Figure 3. Power spectra of the various signals involved in the low-pass filtering example.

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

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. Figure 3 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 equation (2) is useless.

## 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, most notably the singular value decomposition.

### *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. The most complete survey is probably [2]. Here, for pedagogical reasons, we limit or discussion to *quadrature methods* based on well-known quadrature rules. Recall that a quadrature rule for computing an approximation to a definite integral takes the following general form

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

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 (which is identical to the trapezoidal rule for periodic functions), we have

$$t_j = \frac{j - 0.5}{n}, \qquad w_j = \frac{1}{n}, \quad j = 1, \ldots, n,$$

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

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

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

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

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 $\psi$ defined above equals the right-hand side $g$ at given points $s_1, \ldots, s_m$:

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

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 paper, i.e., $m = n$. Inserting the expression for $\psi(s)$ into the collocation scheme, we arrive at the following equations

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

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

$$\begin{pmatrix} w_1 K(s_1, t_1) & w_2 K(s_1, t_2) & \ldots & w_n K(s_1, t_n) \\ w_1 K(s_2, t_1) & w_2 K(s_2, t_2) & \ldots & w_n K(s_2, t_n) \\ \vdots & \vdots & & \vdots \\ w_1 K(s_n, t_1) & w_2 K(s_n, t_2) & \ldots & w_n K(s_n, t_n) \end{pmatrix} \begin{pmatrix} \widetilde{f}(t_1) \\ \widetilde{f}(t_2) \\ \vdots \\ \widetilde{f}(t_n) \end{pmatrix} = \begin{pmatrix} g(s_1) \\ g(s_2) \\ \vdots \\ g(s_n) \end{pmatrix}$$

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

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

To illustrate the above scheme, we use the midpoint rule to discretize the gravity surveying problem from section 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

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

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 $b = Ax$. 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 practice the right-hand side is usually a perturbed version of this $b$. That is, we solve the system $Ax = \widetilde{b}$, 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

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

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

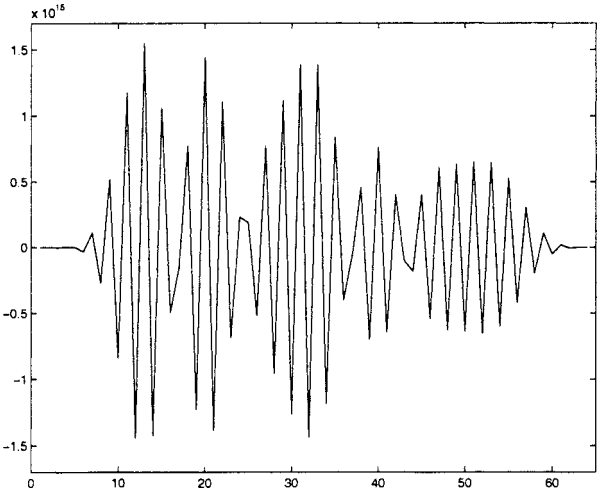


Figure 4. "Naive solution" to the gravity surveying model problem.

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 $Ax = b$ by standard tools in numerical analysis.

Figure 4, which shows the "naive" solution $x_{\text{naive}}$ to the gravity 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 section, and turn our attention to tools-of-the-trade in numerical linear algebra.

### 3.2. *The singular value decomposition*

One of the most versatile tools in linear algebra is probably the *singular value decomposition* (SVD) of a matrix [15], and we shall make heavy use of this decomposition throughout the paper. 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

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

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

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0.$$

It can be shown that the number of nonzero singular values is equal to the rank of $A$ and that the condition number of $A$ is $\mathrm{cond}(A) = \sigma_1/\sigma_n$. Software for computing the SVD is available in all modern software packages.

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

$$(u_i, \sigma_i, v_i) = \begin{cases} (w_i, \lambda_i, w_i) & \text{for } \lambda_i \geqslant 0, \\ (w_i, -\lambda_i, -w_i) & \text{for } \lambda_i < 0, \end{cases} \tag{6}$$

and this relation can be used to simplify the computation of the SVD for symmetric matrices. In particular, if $A$ is symmetric and positive definite then $(u_i, \sigma_i, v_i) = (w_i, \lambda_i, w_i)$ for all $i$.

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

$$Av_i = \sigma_i u_i, \qquad \|Av_i\|_2 = \sigma_i, \quad i = 1, \ldots, n. \tag{7}$$

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

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

The latter relation, combined with (7), leads to the expression

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

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

$$x_{\mathrm{naive}} = \sum_{i=1}^{n} \frac{u_i^{\mathrm{T}} b}{\sigma_i} v_i. \tag{8}$$

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.

### *3.3. 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, more can be said. We omit the underlying theory here (see, e.g., [20] 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 $\sigma_1$ (in infinite precision they would decay to zero).

2. Consequently, the condition number $\mathrm{cond}(A) = \sigma_1/\sigma_n$ 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 $\sigma_i \simeq \mathrm{e}^{-\alpha i}$, where $\alpha$ is a positive real constant.

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 $\sigma_i$ 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 gravity surveying problem from section 1, discretized as explained above with $n = 64$. The singular values are shown in figure 5, 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 figure 6. 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 $i$

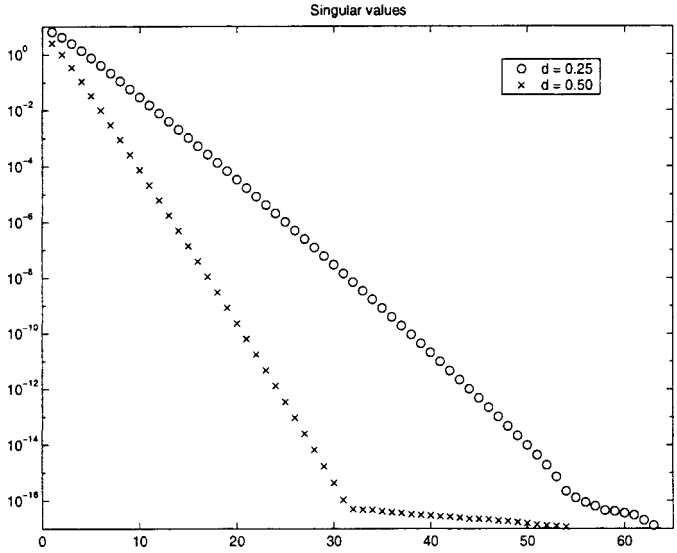


Figure 5. The singular values of the gravity surveying model problem, discretized with the midpoint quadrature rule and $n = 64$, for two values of the depth $d$.

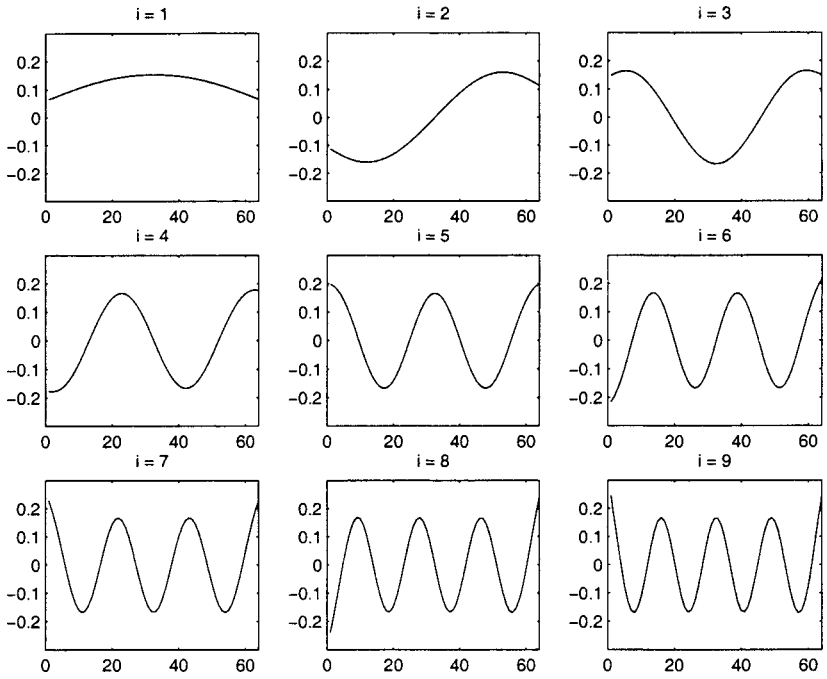


Figure 6. The first nine left singular vectors $u_i$ for the gravity surveying problem.

and the smaller the corresponding singular value $\sigma_i$, the more high-frequent the singular vectors $u_i$ and $v_i$.

From the above SVD analysis, it is clear that in connection with the analysis and solution of the problem $Ax = b$ we should monitor the behavior of the singular values $\sigma_i$, the SVD-components $u_i^{\mathrm{T}}b$ of the right-hand side, and the SVD components $u_i^{\mathrm{T}}b/\sigma_i$ of the computed "naive" solution. Figure 7 shows these quantities for the gravity model problem, and we see that the quantities $|u_i^{\mathrm{T}}b|$ decay until they, too, hit the level set by the machine precision. Note that in the decay region, the quantities $|u_i^{\mathrm{T}}b|$ decay strictly faster than the singular value, such that in the same region the quantities $|u_i^{\mathrm{T}}b/\sigma_i|$ also decay, only slower. In the region where $|u_i^{\mathrm{T}}b|$ levels off, the quantities $|u_i^{\mathrm{T}}b/\sigma_i|$ increase. The "naive" solution $x_{\mathrm{naive}}$ given by equation (8) is completely dominated by the SVD components corresponding to the smallest singular values, and therefore $x_{\mathrm{naive}}$ appears as a highly oscillatory solution with a large norm $\|x_{\mathrm{naive}}\|_2 = 6.1 \cdot 10^{15}$. Figure 4 illustrates this phenomenon.

It is not a coincidence that the absolute values of the SVD components $u_i^{\mathrm{T}}b$ 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 $\int_0^1 f(t)^2\,\mathrm{d}t < \infty$), then independently of the particular discretization scheme the quantities $|u_i^{\mathrm{T}}b|$ will decay faster than the singular values $\sigma_i$ (until they eventually hit the machine precision level). It is beyond the scope of theis paper to go further into the underlying theory; see [19,20] for details. Before attempting to solve any discretization

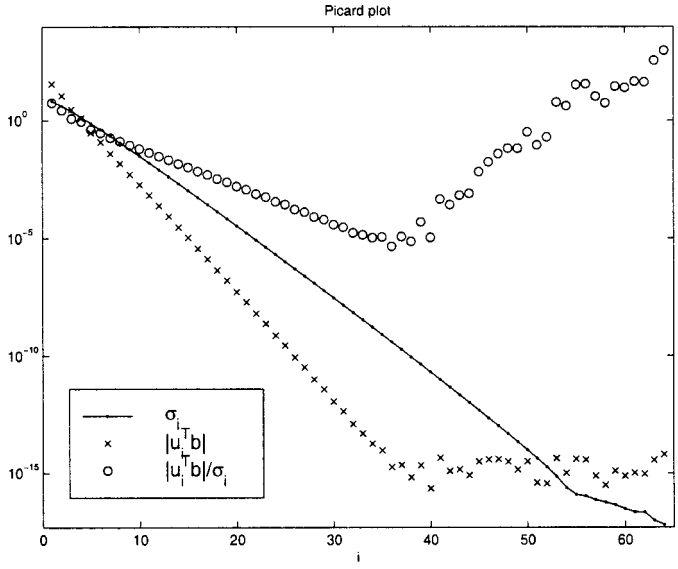


Figure 7. Singular values $\sigma_i$ and coefficients $|u_i^{\mathrm{T}} b|$ and $|u_i^{\mathrm{T}} b|/\sigma_i$ for the gravity surveying model problem. Apart from rounding errors, there is no noise in the right-hand side $b$.

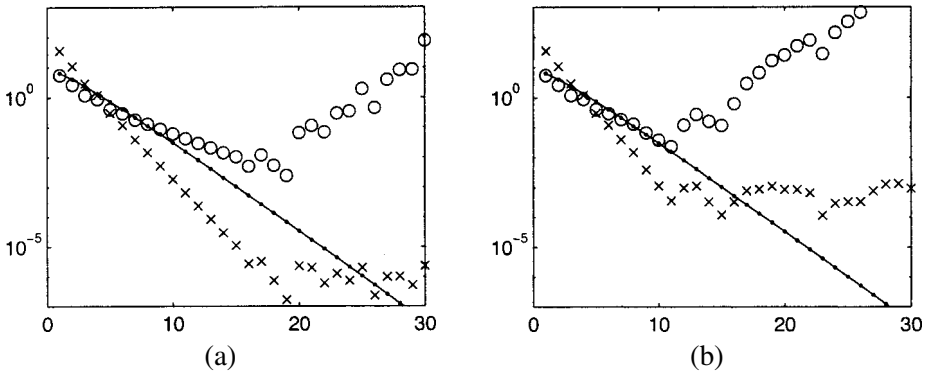


Figure 8. Same plots as before, except that noise with $\sigma_{\mathrm{noise}} = 10^{-6}$ (a) and $\sigma_{\mathrm{noise}} = 10^{-3}$ (b) is added to the right-hand side.

of a first-kind Fredholm equation, one should always try to check whether the quantities $|u_i^{\mathrm{T}} b|$ 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 $\sigma_{\mathrm{noise}}$. Figure 8 shows plots of the SVD quantities $\sigma_i$, $u_i^{\mathrm{T}} \widetilde{b}$, and $u_i^{\mathrm{T}} \widetilde{b}/\sigma_i$ for two noise levels, $\sigma_{\mathrm{noise}} = 10^{-6}$ and $\sigma_{\mathrm{noise}} = 10^{-3}$. We see that as the noise level increases, the decay region of the quantities $|u_i^{\mathrm{T}} \widetilde{b}|$ 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 leveled 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 $x_{\text{naive}}$ 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 section 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

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} v_i, \tag{9}$$

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 9 shows some TSVD solutions to the gravity model problem with $\sigma_{\text{noise}} = 10^{-3}$, along with the exact solution. Notice how $x_k$ improves as $k$ increases, until the noise contributions take over for $k \geqslant 14$.

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;

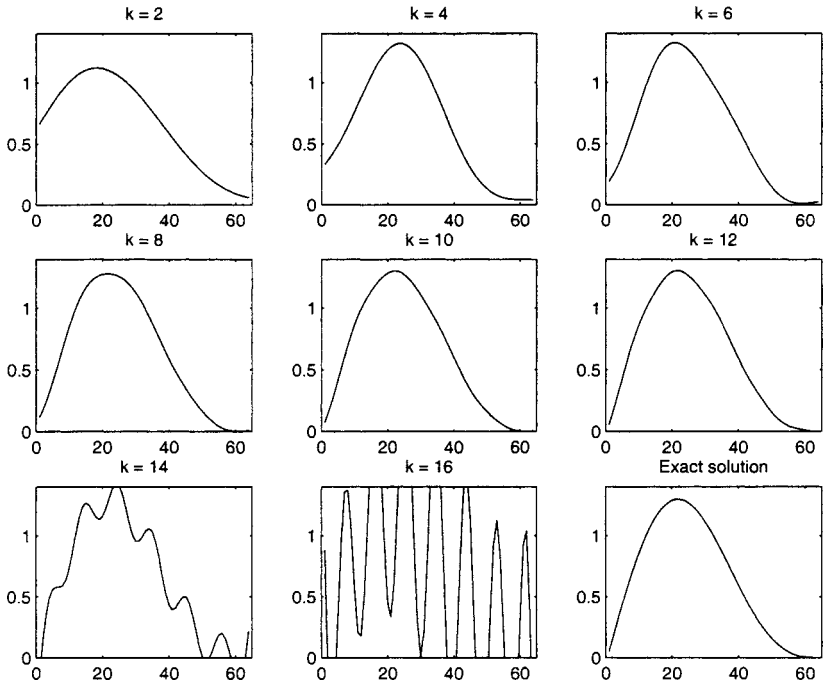


Figure 9. TSVD solutions $x_k$ to the gravity surveying model problem with $\sigma_{\text{noise}} = 10^{-3}$ for $k = 2, 4, \ldots, 16$, and the exact solution.

cf. the discussion in the previous section. There are also other techniques available that do not require monitoring of the $|u_i^{\mathrm{T}} b|$-quantities; but this topic lies outside the scope of this paper, see instead [20].

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 [31] and Tikhonov [33]. It is most commonly referred to as *Tikhonov regularization*, and occasionally as damped least squares. The key idea is to accept a nonzero residual $Ax - b$ and in return obtain a smaller solution norm. This problem can be formulated as follows

$$\min\left\{\|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2\right\}, \tag{10}$$

where the regularization parameter $\lambda$ controls the weight given to minimization of the solution norm $\|x\|_2$ relative to minimization of the residual norm $\|Ax - b\|_2$. It can be shown that there is always a unique solution $x_\lambda$ to the above problem, which we denote the Tikhonov solution. Note that as $\lambda$ approaches zero, the Tikhonov solution $x_\lambda$ approaches the "naive solution" $x_{\text{naive}}$ (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 $\lambda$-values for which there is a reasonable balance between the residual and solution norms, and for which the Tikhonov solution $x_\lambda$ 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 $x_\lambda$ in terms of the SVD of $A$ as follows:

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

The quantities $f_i = \sigma_i^2/(\sigma_i^2 + \lambda^2)$ are called the Tikhonov *filter factors*. They all satisfy $0 \leqslant f_i \leqslant 1$, and they control the damping of the individual SVD components of the solution $x_\lambda$. Specifically, if $\lambda$ is fixed somewhere between $\sigma_1$ and $\sigma_n$, then for $\sigma_i \gg \lambda$ we have $f_i = 1 + \mathrm{O}(\lambda^2/\sigma_i^2) \simeq 1$, while for $\sigma_i \ll \lambda$ we have $f_i = \sigma_i^2/\lambda^2 + \mathrm{O}(\sigma_i^4/\lambda^4) \simeq \sigma_i^2/\lambda^2$. For singular values $\sigma_i$ near $\lambda$, 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 $\lambda$, contribute with almost full strength to the Tikhonov solution $x_\lambda$. Similarly, the last SVD components corresponding to singular values smaller than $\lambda$ are damped considerably and therefore contribute very little to $x_\lambda$. Hence, we would expect that the Tikhonov solutions resemble the TSVD solutions when $k$ and $\lambda$ are chosen such that $\sigma_k \simeq \lambda$; more details can be found in [18].

Figure 10 shows various Tikhonov solutions to the gravity surveying problem with $\sigma_{\mathrm{noise}} = 10^{-3}$ for a large range of $\lambda$-values – notice the resemblance of these solutions with the TSVD solutions in figure 9.

Equation (10) 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

$$\min \left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \tag{11}$$

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

$$\left(A^{\mathrm{T}} A + \lambda^2 I\right) x = A^{\mathrm{T}} b.$$

While the latter is suited in certain situations, it is the least squares formulation (11) that is best suited for numerical computations.

When $A$ is symmetric and positive definite, there exists an alternative form of Tikhonov's method – usually attributed to Franklin [14] – that avoids the normal equations by instead working with the system

$$(A + \lambda I) x = b, \quad \lambda \geqslant 0, \tag{12}$$

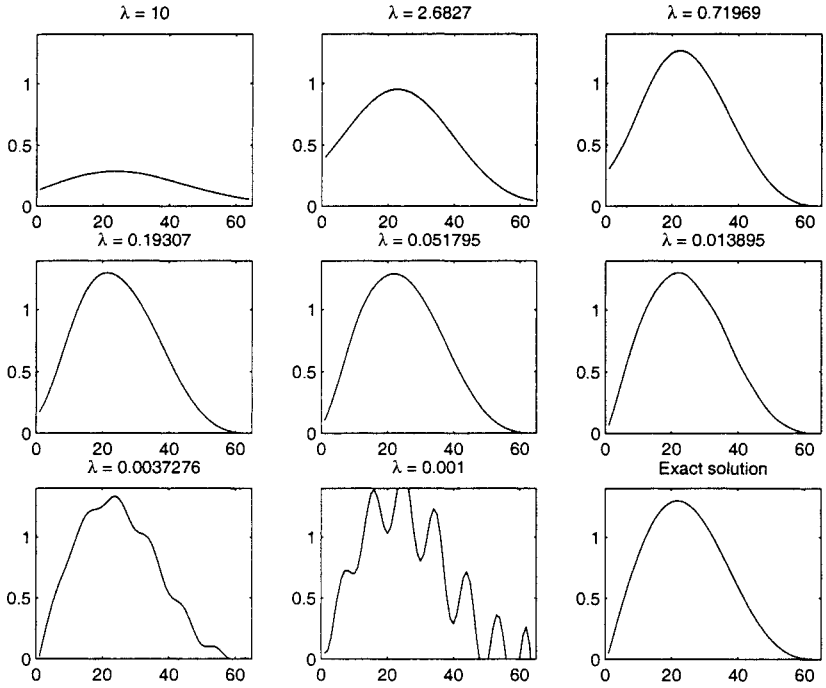


Figure 10. Tikhonov solutions $x_\lambda$ to the gravity surveying model problem with $\sigma_{\text{noise}} = 10^{-3}$ for $\lambda$ in the range $10^{-3}$–10, plus the exact solution.

which can be solved by means of Cholesky factorization. If $A = \sum_{i=1}^{n} w_i \alpha_i w_i^{\mathrm{T}}$ is the eigenvalue decomposition of $A$, then the regularized solution in Franklin's method is given by $x = \sum_{i=1}^{n} (w_i^{\mathrm{T}} b)/(\alpha_i + \lambda) w_i$. This method is particularly useful for Toeplitz matrices because it preserves the Toeplitz structure of the coefficient matrix.

At this stage, we emphasize 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

$$L_1 = n^{-1} \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix}, \qquad L_2 = n^{-2} \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix}. \tag{13}$$

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 following formulation of Tikhonov regularization in general form, as opposed to the standard form in (10):

$$\min \left\{ \|Ax - b\|_2^2 + \lambda^2 \|Lx\|_2^2 \right\}. \tag{14}$$

This general-form version of Tikhonov regularization is also available via the function tikhonov in REGULARIZATION TOOLS.

### 4.3. 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

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

where $\omega$ is a real positive parameter satisfying $0 < \omega < 2/\|A^{\mathrm{T}}A\|_2$, 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 $k$th iteration vector is given by

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

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 figure 11 for $k = 10$, 20, 40 and 80. We see that the filter factors resemble those for Tikhonov regularization, with

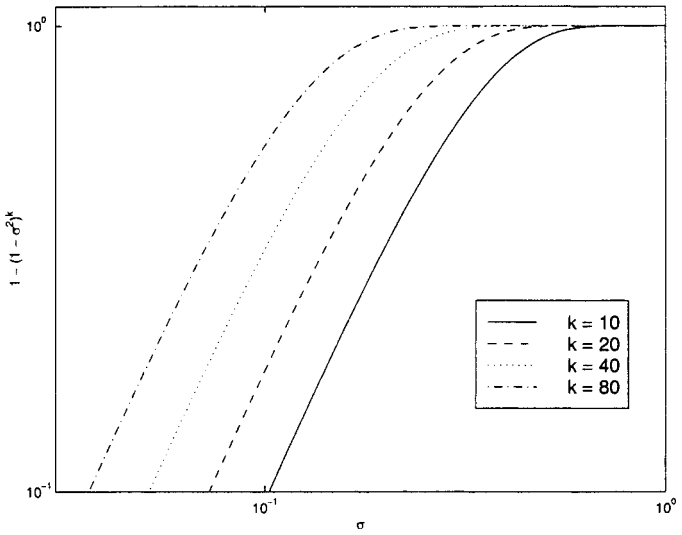


Figure 11. The function $1 - (1 - \sigma^2)^k$ which defines the Landweber filter factors, for $\omega = 1$ and $k = 10$, 20, 40 and 80.

$f_i^{(k)} \simeq 1$ 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 size of the small filter factors.

Similar stationary iterative methods can be found in the literature, 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 $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ associated with a least squares problem $\min \|Ax - b\|_2$. This variant is called CGLS, and it was described in the original paper by Hestenes and Stiefel [22] where the CG method was first published. Again, let $x^{(0)}$ denote the initial guess, and define the two auxiliary vectors $r^{(0)} = b - Ax^{(0)}$ and $d^{(0)} = A^{\mathrm{T}}r^{(0)}$. Then the CGLS iterations take the following form for $k = 1, 2, \ldots$

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

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

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 (11) with $A$ and $b$ replaced by $\binom{A}{\lambda I}$ and $\binom{b}{0}$, 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 (see [15, section 10.3]), but none of the preconditioners for general problems are suited for Tikhonov regularization problems, due to the spectral properties of the coefficient matrix $A^{\mathrm{T}}A + \lambda^2 I$. Development of specialized preconditioners for this matrix is a topic of current research, and a few algorithms for general unstructured matrices have appeared recently [17]. We return to preconditioners for structured matrices in section 5.3.

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

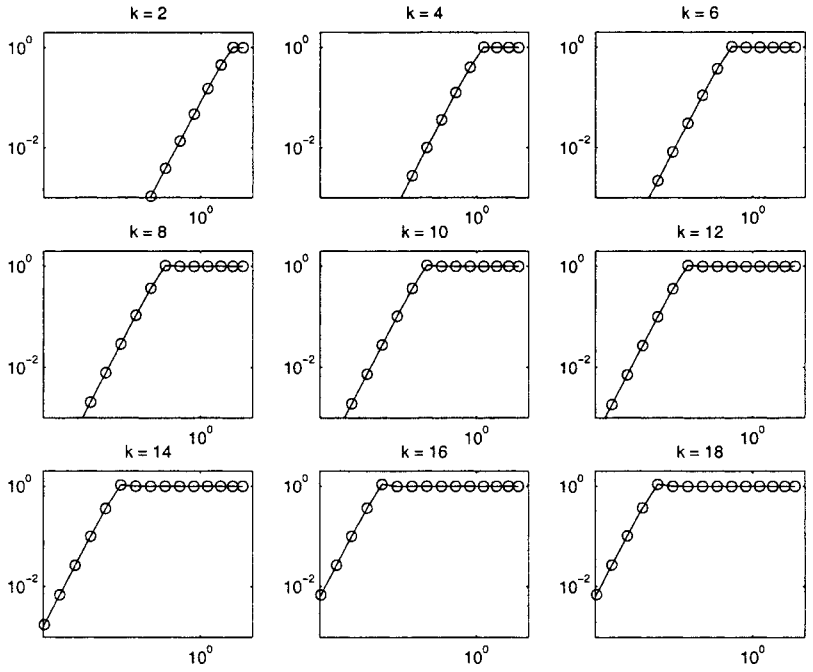


Figure 12. CGLS filter factors $f_i^{(k)}$ as functions of the singular values $\sigma_i$ for the gravity surveying test problem with $\sigma_{\text{noise}} = 10^{-3}$.

hard to analyze theoretically, because it is a nonstationary 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

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

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 $f_i^{(k)} = \mathrm{O}(\sigma_i^2)$ for the small singular values (see [20, section 6.3.2]). In the transition range, certain filter factors can become slightly larger than one.

Plots of typical filter factors are shown in figure 12 – 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 $\sigma_i$ 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 $\ell(k) \leqslant k$ of SVD components are captured in the $k$th iteration, cf. table 1.

Table 1
Corresponding values of $k$ and $\ell(k)$ in the CGLS algorithm for the problem in figure 12.

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

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 figure 12, with a flat region where $f_i^{(k)} \simeq 1$ and a decaying region where $f_i^{(k)} = \mathrm{O}(\sigma_i^2)$. A full analysis of this problem has not been developed yet – partial results can be found in [20], 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 $|u_i^{\mathrm{T}} b|$, 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 [27], 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 $k$th step of this method is often much larger than $k$.

## 5. Deconvolution in one dimension

We now return to the main theme of this paper, namely, deconvolution problems, and we show how the above general algorithms specialize – in 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

$$A = HW, \tag{15}$$

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.,

$$h_{ij} = h(s_i - t_j), \quad i, j = 1, \ldots, n. \tag{16}$$

At this stage we will assume that the quadrature points $s_i$ and the collocation points $t_j$ are identical and equidistantly spaced, 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 $\ell$. 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

$$T = \begin{pmatrix} t_0 & t_{-1} & t_{-2} & \ldots & t_{1-n} \\ t_1 & t_0 & t_{-1} & \ldots & t_{2-n} \\ t_2 & t_1 & t_0 & \ldots & t_{3-n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & t_{n-3} & \ldots & t_0 \end{pmatrix}. \tag{17}$$

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 equation (16) 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 $\ell$, 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

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

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

The persymmetry of $T$ can be used to derive certain symmetry relations between the singular vectors of $T$. From the symmetry of the matrix $TJ$, it follows from (6) that

$$TJ = \sum_{j=1}^{n} w_j \gamma_j \lambda_j (\gamma_j w_j)^{\mathrm{T}} \quad \Longleftrightarrow \quad T = \sum_{j=1}^{n} w_j \gamma_j \lambda_j (\gamma_j J w_j)^{\mathrm{T}},$$

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

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

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

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

In other words, the left and right singular vectors of a symmetric Toeplitz matrix are identical except perhaps for a sign change, and the sequence of elements in each vector is symmetric around the middle elements except perhaps for a sign change. We illustrate this by means of a small example:

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

$$U = \begin{pmatrix} 0.628 & 0.707 & -0.325 \\ 0.460 & 0 & 0.888 \\ 0.628 & -0.707 & -0.325 \end{pmatrix}, \qquad V = \begin{pmatrix} 0.628 & -0.707 & 0.325 \\ 0.460 & 0 & -0.888 \\ 0.628 & 0.707 & 0.325 \end{pmatrix}.$$

Here, $u_1 = Ju_1 = v_1 = Jv_1$, $u_2 = -Ju_2 = -v_2 = Jv_2$, and $u_3 = Ju_3 = -v_3 = -Jv_3$. The singular values of $T$ are $\sigma_1 = 2.73$, $\sigma_2 = 2$ and $\sigma_3 = 0.732$.

### 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

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

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 $g_i = \sum_{j=0}^{n-1} f_j h_{(i-j) \bmod n}$. Thus, if $h$ is the first column of $C_h$, then $g$ is obviously given by

$$g = C_h f. \tag{18}$$

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 $\hat{f} = \mathrm{DFT}(f) = F_n f$, where the elements of the *complex symmetric* matrix $F_n$ are given by

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

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 $f = \mathrm{IDFT}(\hat{f}) = F_n^{-1}\hat{f} = n^{-1}\mathrm{conj}(F_n)\hat{f}$. We now insert these two relations into expression (1) for $g$ to obtain

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

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 (18) 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 $5n \log_2 n$

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 $2.5n \log_2 n + 5n + \mathrm{O}(1)$ 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 $\mathrm{O}(n \log_2 n)$ flops – as opposed to $2n^2$ flops for a general matrix–vector multiplication. We derive the algorithm here, and more details can be found in [34].

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 equation (17), the first column of $C$ is constructed such that

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

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 $p \geqslant 2n$ in order to speed up the FFT computations. For example, if $T$ is the nonsymmetric $3 \times 3$ matrix

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

then $p = 8$ and $C$ takes the form

$$C = \begin{pmatrix} 3 & -2 & -1 & 0 & 0 & 0 & 1 & 2 \\ 2 & 3 & -2 & -1 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & -2 & -1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & -2 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & -2 & -1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & -2 & -1 \\ -1 & 0 & 0 & 0 & 1 & 2 & 3 & -2 \\ -2 & -1 & 0 & 0 & 0 & 1 & 2 & 3 \end{pmatrix},$$

where we recognize $T$ as the leading $3 \times 3$ principal submatrix of $C$.

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

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

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

$$\hat{z} = C\hat{x} = \begin{pmatrix} Tx \\ \hat{z}_2 \end{pmatrix}.$$

Due to the zeros in $\hat{x}$, the first $n$ components of $\hat{z}$ are identical to the desired matrix–vector product $Tx$. 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 $7.5p \log_2 p$ flops. Since $p$ is bounded above by $4n$ we obtain the approximate upper bound $30n \log_2 n$ 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 $15n \log_2 n$ flops.

Note that if $T$ is real and symmetric, then the vector $C(:, 1)$ is "real even" and 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 $2n_b n$ flops. When $n$ is a power of two, 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 13 illustrates this point.

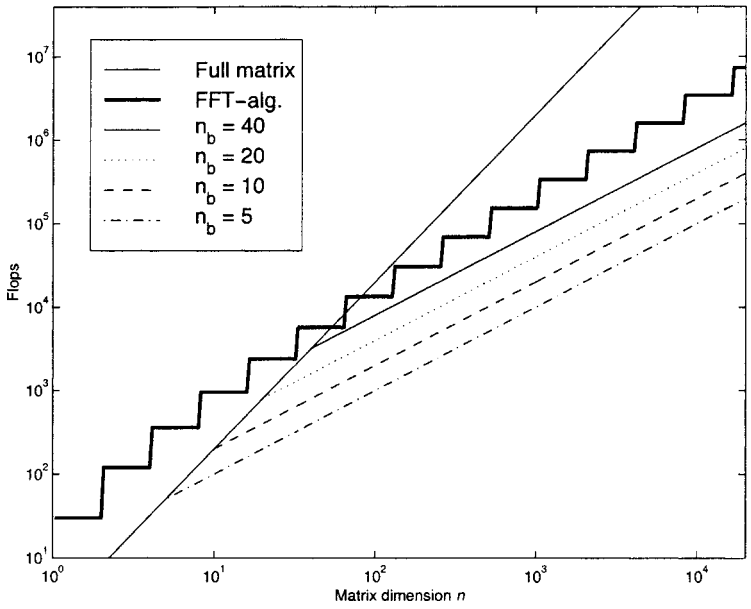


Figure 13. Flop counts for the FFT-based Toeplitz matrix–vector multiplication algorithm, compared to flop counts for ordinary matrix–vector multiplication with full and banded matrices.

### 5.4. Circulant preconditioners

We will here briefly summarize the main ideas of some current algorithms for computing circulant preconditioners specialized to regularization problems with a Toeplitz coefficient matrix. Circulant matrices are a natural choice as preconditioners for Toeplitz matrices because they are inexpensive to construct and use.

If $T$ is square then we can start by computing a circulant approximation $C_T$ to $T$. For example, we can use the closest circulant matrix to $T$ in the Frobenius norm [9], whose first column has elements given by

$$(C_T)_{i1} = \frac{(n-i)t_i + it_{i-n}}{n}, \quad i = 0, \ldots, n-1. \tag{19}$$

Next we decompose the circulant matrix $C_T$ as

$$C_T = F_n^{-1} \Lambda_T F_n,$$

where $F_n$ is the Fourier matrix and $\Lambda_T$ is a diagonal matrix with the eigenvalues of $C_T$. We also need the following expression for the transpose: $C_T^{\mathrm{T}} = \mathrm{conj}(C_T^{\mathrm{T}}) = \mathrm{conj}(F_n \Lambda_T F_n^{-1}) = \mathrm{conj}(F_n)\,\mathrm{conj}(\Lambda_T) n^{-1} F_n = F_n^{-1} \mathrm{conj}(\Lambda_T) F_n$, where we have used that $F_n^{-1} = n^{-1}\mathrm{conj}(F_n)$. Then it is natural to precondition the Tikhonov problem (10) with the following preconditioner

$$\begin{aligned} M = \left(C_T^{\mathrm{T}} C_T + \lambda^2 I\right)^{-1} &= \left(F_n^{-1}\mathrm{conj}(\Lambda_T)\Lambda_T F_n + \lambda^2 I\right)^{-1} \\ &= F_n^{-1}\left(|\Lambda_T|^2 + \lambda^2 I\right)^{-1} F_n. \end{aligned}$$

By its definition $M$ is real and symmetric, and if we choose $Q$ as the square root of $M$,

$$Q = M^{1/2} = F_n^{-1}\left(|\Lambda_T|^2 + \lambda^2 I\right)^{-1/2} F_n, \tag{20}$$

then it follows that $Q$ is also real and symmetric, and the preconditioned problem becomes $Q(T^{\mathrm{T}}T + \lambda^2 I)Qz = QT^{\mathrm{T}}b$ with $x = Qz$. This symmetric problem is solved by means of the CGLS algorithm. We emphasize that the matrix $Q$ is never formed explicitly, instead FFTs are used to perform all the matrix multiplications. Further implementation details and extensions to 2-D problems are described in [8], where the general-form regularization problem in (14) is also considered.

The same idea is used in [16] to derive a regularizing circulant preconditioner for symmetric matrices. Again we start with a circulant approximation $C_T = F_n^{-1}\Lambda_T F_n$ to $T$, but now we use the preconditioner

$$Q_\tau = F_n^{-1} D F_n \tag{21}$$

where $D = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with elements

$$d_i = \begin{cases} |\lambda_i|^{-1/2} & \text{for } |\lambda_i| \geqslant \tau, \\ 1 & \text{else} \end{cases} \tag{22}$$

in which $\lambda_i$ are the eigenvalues of $C_T$. The preconditioned symmetric problem $Q_\tau T Q_\tau z = Q_\tau b$ with $x = Q_\tau z$ can then be solved by means of CG if $T$ is positive

definite, or ORTHOMIN if $T$ is indefinite. The threshold $\tau$ plays the role of the regularization parameter, in the sense that eigenvalues of $C_T$ whose magnitude is less than $\tau$ are not affected by the preconditioner, while the remaining eigenvalues of $C_T$ are mapped to $\pm 1$. We refer to [16] for more details and extensions to the 2-D case.

Finally we mention an algorithm from [28] for a banded $(n+q)\times n$ Toeplitz matrix of the form

$$T = \begin{pmatrix} t_0 & & \\ \vdots & t_0 & \\ t_q & \vdots & \ddots \\ & t_q & \\ & & \ddots \end{pmatrix}.$$

The key idea here is to augment $T$ with additional $q$ columns – in the obvious way – to make it square and circulant,

$$C_{\text{aug}} = \left(T, \breve{T}\right) = F_n^{-1} \Lambda_{\text{aug}} F_n.$$

Following the previously mentioned techniques, we now formally introduce the real symmetric matrix

$$Q_{\text{aug}} = F_n^{-1} \left( |\Lambda_{\text{aug}}|^2 + \lambda^2 I \right)^{-1/2} F_n.$$

Finally, we partition $Q_{\text{aug}}$ as

$$Q_{\text{aug}} = \left(Q, \breve{Q}\right)$$

such that $Q$ has the same dimensions as $T$, and solve the preconditioned problem $Q(T^{\mathrm{T}}T + \lambda^2 I)Q^{\mathrm{T}}z = QT^{\mathrm{T}}b$ with $x = Q^{\mathrm{T}}z$. Again we stress that $Q_{\text{aug}}$ or $Q$ are never formed explicitly, and we refer to the original paper for details and extensions to 2-D problems.

We mention in passing that instead of using a circulant preconditioner, one can transform the Toeplitz matrix into a Cauchy-like matrix (which is numerically stable) and then construct a preconditioner from a rank-revealing pivoted LU decomposition of the Cauchy-like matrix; see [26] for details.

### 5.5. *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 (15) can always be "absorbed" into the solution vector $x$, we can assume without loss of generality that the coefficient matrix $T$ is a Toeplitz matrix.

We consider first the special case where the coefficient matrix $T$ is triangular and Toeplitz, and $L$ is a $p \times n$ Toeplitz matrix with dimensions $p \leqslant n$ and with zeros below

the main diagonal. Triangular Toeplitz matrices arise, e.g., in connection with Volterra-type deconvolution problems (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 $T$ 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 $L$):

$$\begin{pmatrix} T \\ \lambda L \end{pmatrix} = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 & t_4 & t_5 \\ & t_0 & t_1 & t_2 & t_3 & t_4 \\ & & t_0 & t_1 & t_2 & t_3 \\ & & & t_0 & t_1 & t_2 \\ & & & & t_0 & t_1 \\ & & & & & t_0 \\ l_0 & l_1 & l_2 & l_3 & l_4 & l_5 \\ & l_0 & l_1 & l_2 & l_3 & l_4 \\ & & l_0 & l_1 & l_2 & l_3 \\ & & & l_0 & l_1 & l_2 \end{pmatrix}$$

We shall now demonstrate how to transform this matrix into upper triangular form by means of $n$ Givens transformations. The algorithm is due to Eldén [11]. First, we apply a Givens rotation to rows 1 and $n + 1$ in order to annihilate the element $l_0$; 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 $a'_0$. 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 $T$ and $L$ the new rows are just shifted versions of rows 1 and $n + 1$. Thus, we obtain the first intermediate matrix

$$M' = \begin{pmatrix} t'_0 & t'_1 & t'_2 & t'_3 & t'_4 & t'_5 \\ & t'_0 & t'_1 & t'_2 & t'_3 & t'_4 \\ & & t'_0 & t'_1 & t'_2 & t'_3 \\ & & & t'_0 & t'_1 & t'_2 \\ & & & & t'_0 & t'_1 \\ & & & & & t'_0 \\ & l'_1 & l'_2 & l'_3 & l'_4 & l'_5 \\ & & l'_1 & l'_2 & l'_3 & l'_4 \\ & & & l'_1 & l'_2 & l'_3 \\ & & & & l'_1 & l'_2 \end{pmatrix}.$$

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

$$M'' = \begin{pmatrix} t_0' & t_1' & t_2' & t_3' & t_4' & t_5' \\ & t_0'' & t_1'' & t_2'' & t_3'' & t_4'' \\ & & t_0'' & t_1'' & t_2'' & t_3'' \\ & & & t_0'' & t_1'' & t_2'' \\ & & & & t_0'' & t_1'' \\ & & & & & t_0'' \\ & & l_2'' & l_3'' & l_4'' & l_5'' \\ & & & l_2'' & l_3'' & l_4'' \\ & & & & l_2'' & l_3'' \\ & & & & & l_2'' \end{pmatrix}.$$

Obviously, we can repeat this process on the submatrix $M''(3{:}n+p, 3{:}n)$, and so on, until we end up with an upper triangular matrix after $n$ steps. The algorithm uses solely Givens rotations and is therefore numerically stable. The total amount of work is approximately $8n^2$, and the storage requirement is dominated by the need to store $n(n+1)/2$ elements of the final triangular matrix.

Next we consider the more general case where $T$ is a general $m \times n$ Toeplitz matrix, and we present an algorithm due to Bojanczyk et al. [6] (with improvements by Park and Eldén [29]) for computing the triangular matrix $R$ in the QR factorization

$$\begin{pmatrix} T \\ \lambda I \end{pmatrix} = QR.$$

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

$$\begin{pmatrix} T \\ \lambda I \end{pmatrix} = \begin{pmatrix} t_0 & u^{\mathrm{T}} \\ v & \widehat{T} \\ \lambda & 0^{\mathrm{T}} \\ 0 & \lambda \hat{I} \end{pmatrix} = \begin{pmatrix} \widehat{T} & \widetilde{u} \\ \widetilde{v}^{\mathrm{T}} & t_{m-n} \\ \lambda \hat{I} & 0 \\ 0^{\mathrm{T}} & \lambda \end{pmatrix},$$

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

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

By comparing the submatrices in the equation

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

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

$$r_{11}^2 = t_0^2 + v^{\mathrm{T}} v + \lambda^2, \qquad z^{\mathrm{T}} = \frac{t_0 u^{\mathrm{T}} + v^{\mathrm{T}} \widehat{T}}{r_{11}}, \qquad R_b^{\mathrm{T}} R_b = R_t^{\mathrm{T}} R_t + u u^{\mathrm{T}} - \widetilde{v}\,\widetilde{v}^{\mathrm{T}} - z z^{\mathrm{T}}.$$

Notice that $\lambda$ 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, $(r_{11}, z^{\mathrm{T}})$. 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., [15, section 12.5].

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 $u^{\mathrm{T}}$, annihilates the first element of $u$, i.e.,

$$G_1 \begin{pmatrix} R_t \\ u^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \widehat{R}_t \\ (u')^{\mathrm{T}} \end{pmatrix}, \quad u'(1) = 0.$$

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 $\widetilde{v}^{\mathrm{T}}$ and $z^{\mathrm{T}}$, annihilates the first element of $z$, i.e.,

$$\widehat{G}_1 \begin{pmatrix} \widetilde{v}^{\mathrm{T}} \\ z^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \hat{v}^{\mathrm{T}} \\ (z')^{\mathrm{T}} \end{pmatrix}, \quad z'(1) = 0.$$

Finally, we determine a stabilized hyperbolic rotation $H_1$ which, when applied to the first row of $\widehat{R}_t$ and $\hat{v}^{\mathrm{T}}$, annihilates the first element of $\hat{v}$, i.e.,

$$H_1 \begin{pmatrix} \widehat{R}_t \\ \hat{v}^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} R_t' \\ (\widetilde{v}')^{\mathrm{T}} \end{pmatrix}, \quad \widetilde{v}'(1) = 0.$$

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:

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

where $u''(1) = z''(1) = 0$. Then we use a stabilized hyperbolic rotation $H_2$ to annihilate the second element of $\hat{v}'$ by means of the second row of $R_t''$:

$$H_2 \begin{pmatrix} \widehat{R}_t' \\ (\hat{v}')^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} R_t'' \\ (\widetilde{v}'')^{\mathrm{T}} \end{pmatrix}, \quad \widetilde{v}''(1) = 0.$$

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 compute the Tikhonov solution via the semi-normal equations (see [5, section 6.6.5]):

$$R^{\mathrm{T}} R x_\lambda = T^{\mathrm{T}} b.$$

This requires additional $(m + 2n)n$ flops, which can be reduced if the FFT is used to compute $T^{\mathrm{T}} b$. For practical issues as well as an error analysis, see [30].

A word about the numerical stability of the algorithm. It is proved in [29] that the forward error in the computed $R$ is proportional to the condition number of the matrix $T^{\mathrm{T}} T + \lambda^2 I$. Although $T$ is very ill conditioned, $\lambda$ is always chosen such that $T^{\mathrm{T}} T + \lambda^2 I$ is more well conditioned, and hence $R$ is normally computed with sufficient accuracy. The paper [29] also describes an algorithm to improve the accuracy of $R$.

If $T$ is symmetric and positive definite then as an alternative we can use Franklin's method (12) and solve the system $(T + \lambda I)x = b$ for positive $\lambda$. This system can be solved in $6n^2$ flops by means of Levinson's algorithm, cf. [15, section 4.7]. Franklin's method may also be applied to symmetric indefinite or general problems, but then care must be taken to ensure that $\lambda$ is sufficiently different from all eigenvalues of $T$, and it may be necessary to use a look-ahead Levinson algorithm [10].

We emphasize that all algorithms in this section must be started over if a regularized solution with a new value of $\lambda$ is desired.

### 5.6. *Periodic versus nonperiodic deconvolution problems*

The periodic deconvolution problem (1) corresponds to a linear system of equations with a circulant coefficient matrix $C$, and if regularization is not needed then the solution is given by equation (2). If regularization is needed, then we can easily modify this formula to include Franklin's version of regularization (12). Since $C = W \Lambda W^{\mathrm{T}}$ with $W = n^{-1/2} F_n$ and $\Lambda = \mathrm{diag}(\mathrm{DFT}(h))$, it is easy to see that the regularized solution can be computed as

$$f = \mathrm{IDFT}\big(\mathrm{DFT}(g) \oslash \big(\mathrm{DFT}(h) + \lambda\big)\big). \tag{23}$$

Although Franklin's method is not guaranteed to work for general matrices, it will work for any circulant $C$ because its eigenvectors (the columns of $F_n$) have the characteristic oscillation property mentioned in section 3.3.

In nonperiodic deconvolution problems where the Toeplitz coefficient matrix $T$ is not circulant, i.e., where the signal $h$ is not periodic with period $n$, equation (23) 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 embed $T$ 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 $\hat{x} = C^{-1} \hat{b}$, where $\hat{b}$ is the augmented version of $b$. Then as the approximate solution $\widetilde{x}$ we use those elements of $\hat{x}$ that correspond to the locations of the $b$-element in $\hat{b}$. We note that if a cyclic shift is applied to $\hat{b}$ then the same cyclic shift

applies to $\hat{x}$, and thus the approximate solution $\widetilde{x}$ is independent on the locations of the zeros in $\hat{b}$.

Unfortunately, this approach is not guaranteed to yield a good approximation to the desired solution. To illustrate this, we assume that $n$ is a power of 2 and introduce the notation

$$C = \begin{pmatrix} T & \Upsilon \\ \Upsilon & T \end{pmatrix}, \qquad \hat{x} = \begin{pmatrix} \widetilde{x} \\ z \end{pmatrix}, \qquad \hat{b} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

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

$$C^{-1} = \begin{pmatrix} T^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} T^{-1}\Upsilon S^{-1}\Upsilon T^{-1} & -T^{-1}\Upsilon S^{-1} \\ -S^{-1}\Upsilon T^{-1} & S^{-1} \end{pmatrix}, \tag{24}$$

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

$$\widetilde{x} = x + T^{-1}\Upsilon S^{-1}\Upsilon x$$

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

The error component $T^{-1}\Upsilon S^{-1}\Upsilon x$ 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 $b = Tx$ is padded with 32 zeros on top and 32 zeros below, and the computed vector $\hat{x}$ of length $2n = 128$ is shown in figure 14 with the exact solution $x$ superimposed at the correct position. Notice the large, undesired ringing.

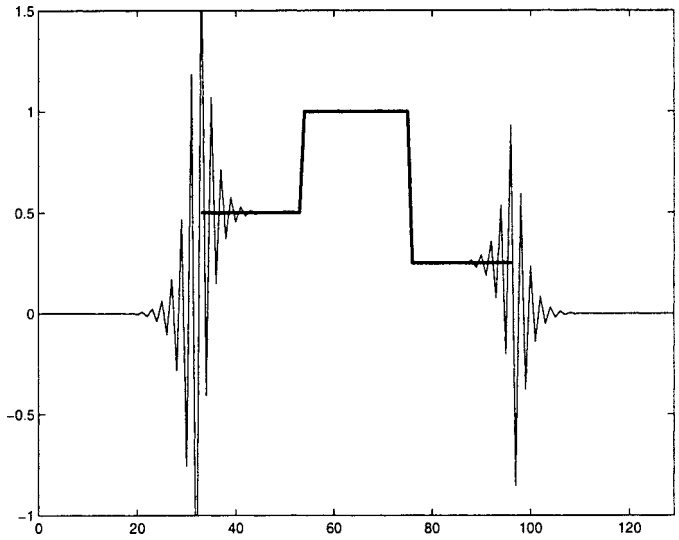


Figure 14. The approximate solution $\widetilde{x}$ is embedded in the vector $\hat{x}$, and the exact solution $x$ is superimposed. Notice the large oscillations ("ringing").

Alternatively, if the underlying signal $h$ is peaked at index 0 and decaying away from this index, thus making $T$ as effectively banded matrix, then one could replace $h$ with a periodic signal of the same length, i.e., with the sequence $h_0, h_1, \ldots, h_{n/2-1}, h_{n/2}, h_{n/2-1}, \ldots, h_2, h_1$. This correspond to replacing the Toeplitz matrix $T$ with a circulant matrix of the same order whose first column – in Matlab notation – is given by $[T(1{:}n/2, 1); T(n/2 + 1{:}-1{:}2)]$. The solution to this modified problem has exactly the same difficulties with ringing as the solution in the previously described approach.

The ringing effect is caused by the sharp transitions in the padded vector $\hat{b}$ between the nonzero elements in $b$ and the padded zeros. Figure 15 illustrates this: the top plot shows $\hat{b}$ while the solid line in the bottom plot is the correct right-hand side vector given by

$$C \begin{pmatrix} x \\ 0 \end{pmatrix} = \begin{pmatrix} Tx \\ \Upsilon x \end{pmatrix} = \begin{pmatrix} b \\ \Upsilon x \end{pmatrix}. \tag{25}$$

Both plots are circularly shiftet (as in figure 14) in order to emphasize the error, which is shown as the black filled areas. It is precisely this error, caused by the "missing"

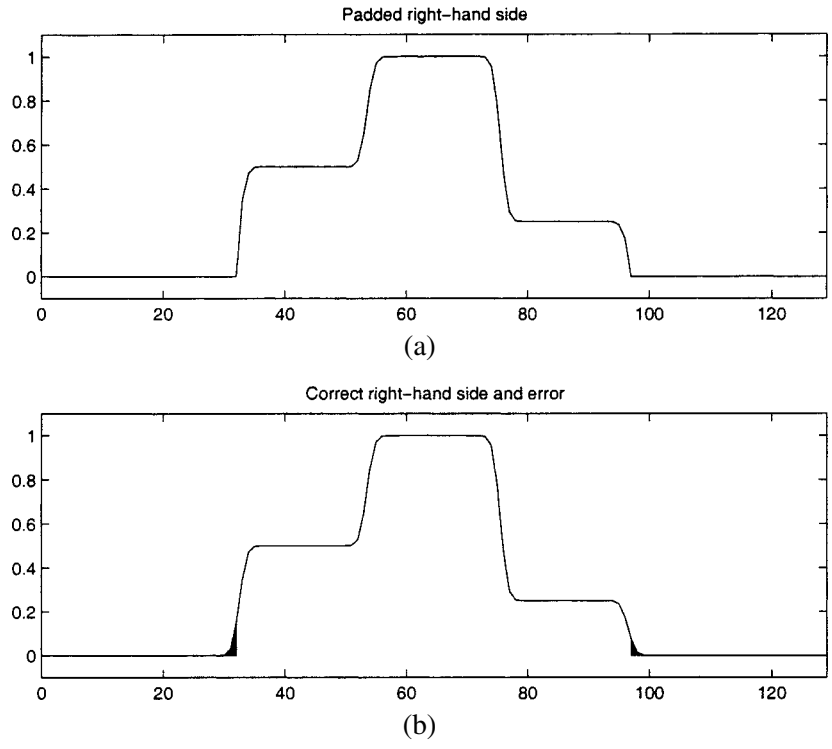


Figure 15. (a) The padded right-hand side $\hat{b}$, and (b) the correct right-hand side together with their difference. Both figures are circularly shifted to emphasize the error which occurs at the borders of the padded parts.

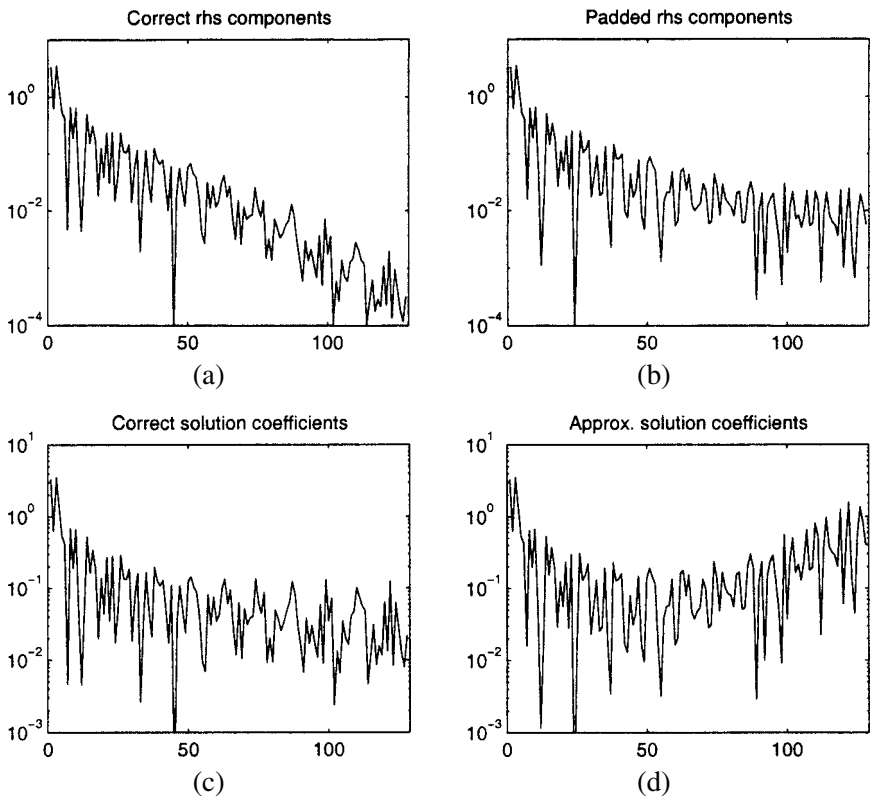


Figure 16. Figures (a) and (c) show the coefficients when expressing the correct right-hand side and the corresponding solution in terms of the eigenvectors of $C$ (i.e., the columns of $F_n$); (b) and (d) show the same coefficients for the padded right-hand side $\hat{b}$ and the corresponding approximate solution $\hat{x}$.

subvector $\Upsilon x$ in $\hat{b}$, that causes the oscillations in $\hat{x}$, because high-frequency components are needed in order to represent the error and because these high-frequncy components are amplified by the inversion process, thus causing the ringing effect.

Figure 16 illustrates the same phenomenon from a different perspective, namely, in terms of the coefficients in an eigenvector expansion. Recall that the eigenvectors of $C$ are the columns of the DFT matrix $F_n$. Figures 16(a) and (c) show the absolute values of the coefficients for the correct right-hand side and the corresponding correct solution, given by the vectors

$$F_n \begin{pmatrix} b \\ \Upsilon x \end{pmatrix} \quad \text{and} \quad F_n \begin{pmatrix} x \\ 0 \end{pmatrix} = \Lambda^{-1} F_n \begin{pmatrix} b \\ \Upsilon x \end{pmatrix}.$$

Note that the correct right-hand side coefficients decay fast enough to ensure that the inversion (i.e., the multiplication with $\Lambda^{-1}$) produces a solution with decaying coefficients.

Figures 16(b) and (d) show the absolute values of the coefficient vectors $F_n\hat{b}$ and $F_n\hat{x} = \Lambda^{-1}F_n\hat{x}$ for the padded right-hand side $\hat{b}$ and the corresponding approximate solution $\hat{x}$. The perturbation of the high-frequent components is clearly visible. The decay of the coefficients $F_n\hat{b}$ is now slower than the decay of the eigenvalues for indices greater than 50. As a consequence the coefficients to the high-frequency components of the approximate solution $\hat{x}$ increase and give rise to a severe perturbation of $\hat{x}$.

### 5.7. *An iterative solver based on embedding*

We shall here describe a new iterative algorithm[2] for solving Toeplitz systems, based on the augmented matrix $C$ and using the fact that a system involving this matrix can be solved in $O(n \log_2 n)$ operations. Again, we assume that the order $n$ is a power of two.

Our outset is the relation (25) which shows that if we could somehow "plug in" the vector $\Upsilon x$ in the bottom half of the augmented right-hand side, then the top half of the solution vector would consist of the solution $x$. Now if $\widetilde{x}_0$ denotes an initial guess for the solution (which may be the zero vector), and if we insert $\Upsilon\widetilde{x}_0$ in the right-hand side, then it follows from (24) that the corresponding solution is given by

$$\hat{x}_1 = \begin{pmatrix} \widetilde{x}_1 \\ z_1 \end{pmatrix} = C^{-1}\begin{pmatrix} b \\ \Upsilon\widetilde{x}_0 \end{pmatrix} = \begin{pmatrix} x \\ 0 \end{pmatrix} + \begin{pmatrix} T^{-1}\Upsilon S^{-1}\Upsilon\big(x - \widetilde{x}_0\big) \\ -S^{-1}\Upsilon\big(x - \widetilde{x}_0\big) \end{pmatrix}.$$

Recall that $S$ is the Schur complement given by $S = T - \Upsilon T^{-1}\Upsilon$.

Assuming that $\widetilde{x}_1$ is a better approximation to $x$ than $\widetilde{x}_0$ we can repeat the process, i.e., insert $\Upsilon\widetilde{x}_1$ in the right-hand side and solve the augmented system, which produces the next approximation $\widetilde{x}_2 = x + T^{-1}\Upsilon S^{-1}\Upsilon(x - \widetilde{x}_1)$. Repeating this scheme in an iterative fashion – each time solving the augmented system by means of FFT – leads to a sequence of iterates formally given by

$$\widetilde{x}_k = x + T^{-1}\Upsilon S^{-1}\Upsilon\big(x - \widetilde{x}_{k-1}\big) = x - \big(-T^{-1}\Upsilon S^{-1}\Upsilon\big)^k\big(x - \widetilde{x}_0\big).$$

The error in the $k$th iterate is therefore given by

$$x - \widetilde{x}_k = (-M)^k\big(x - \widetilde{x}_0\big), \quad k = 1, 2, \ldots,$$

where we have introduced the matrix

$$M = T^{-1}\Upsilon S^{-1}\Upsilon.$$

Obviously, a sufficient condition for the iterates to converge to $x$ is that the spectral radius $\rho(M)$, i.e., the largest of the absolute values of the eigenvalues of $M$, is strictly less than one.

In order to analyze this condition in detail, we write $S$ as

$$S = T - \Upsilon T^{-1}\Upsilon = T\big(I - T^{-1}\Upsilon T^{-1}\Upsilon\big)$$

[2] The algorithm is based on an idea by J. Hald, Brüel and Kjær, Denmark.

to obtain

$$S^{-1} = \left(I - \left(T^{-1}\Upsilon\right)^2\right)^{-1} T^{-1}.$$

Inserting this into the expression for $M$ we get

$$M = T^{-1}\Upsilon\left(I - \left(T^{-1}\Upsilon\right)^2\right)^{-1} T^{-1}\Upsilon.$$

The next step is to introduce the eigenvalue decomposition of $T^{-1}\Upsilon$, given by

$$T^{-1}\Upsilon = WDW^{-1}, \quad D = \mathrm{diag}(\delta_1, \ldots, \delta_n).$$

Inserting this into the above formula for $M$ leads to

$$M = WD^2\left(I - D^2\right)^{-1} W^{-1}$$

showing that the eigenvalues of $M$ are $\delta_i^2(1 - \delta_i^2)^{-1}$ for $i = 1, \ldots, n$. The requirement $\rho(M) < 1$ thus leads to the requirement

$$\left|\frac{\delta_i^2}{1 - \delta_i^2}\right| < 1, \quad i = 1, \ldots, n.$$

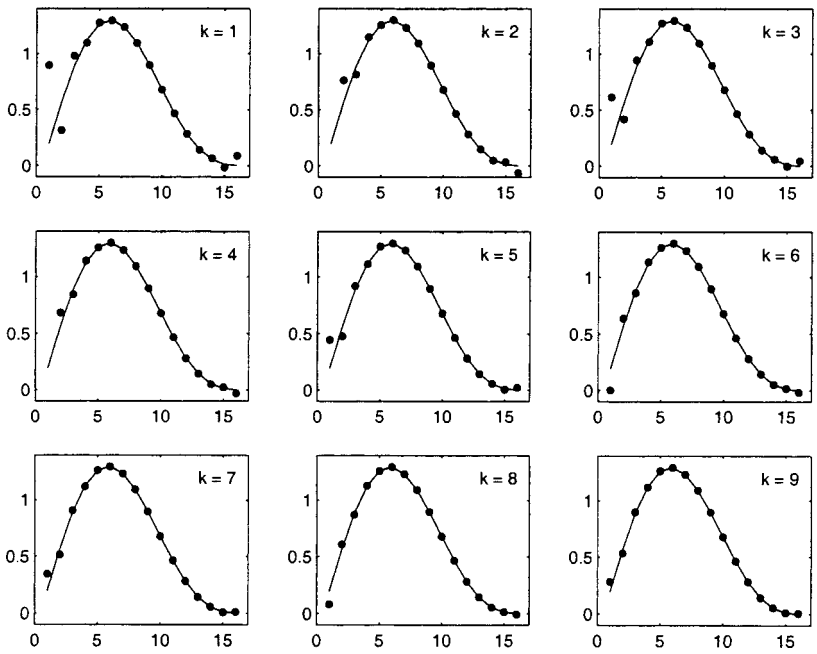


Figure 17. The first nine iterates from the iterative algorithm based on embedding the Toeplitz matrix into a larger circulant matrix.

A straightforward computation shows that if $\delta_i = \Re_i + \hat{\imath}\Im_i$ then

$$\left| \frac{\delta_i^2}{1-\delta_i^2} \right|^2 = \frac{|\delta_i|^4}{|\delta_i|^4 + 1 - 2(\Re_i^2 - \Im_i^2)}$$

and for this quantity to be less than one we must require that $1 - 2(\Re_i^2 - \Im_i^2) > 0$, leading to the requirement $\Re_i^2 - \Im_i^2 < 1/2$.

We have thus proved that the iteration described above will converge to $x$ provided that all the eigenvalues $\delta_i = \Re_i + \hat{\imath}\Im_i$ of the matrix $T^{-1}\Upsilon$ satisfy the requirement

$$\text{Real}\big(\delta_i^2\big) = \Re_i^2 - \Im_i^2 < \frac{1}{2}, \quad i = 1, \ldots, n.$$

It remains to give an interpretation of the matrix $T^{-1}\Upsilon$ and its eigenvalue; we have not found such an interpretation yet.

We illustrate the above iterative algorithm with the gravity surveying problem using $n = 16$ and $d = 0.1$, for which $\rho(M) = 0.78$. The first nine iterates are shown in figure 17; notice that the middle entries in the solution vector $\widetilde{x}_k$ converge faster than the entries at the ends.

## 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, $1024 \times 1024$ 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

$$\int_0^1 \int_0^1 K\big(x, y, x', y'\big) f\big(x', y'\big)\, \mathrm{d}x'\, \mathrm{d}y' = g(x, y). \tag{26}$$

We shall limit our discussion here to the important case where the kernel $K$ is a real 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

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

where $\kappa$ and $\omega$ are real 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 $y'$:

$$\int_0^1 \kappa\big(x - x'\big)\left(\int_0^1 \omega\big(y - y'\big) f\big(x', y'\big)\,\mathrm{d}y'\right)\mathrm{d}x' = g(x, y). \tag{27}$$

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 [4], 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 $w$, then the 2-D object $f$ is related to its image $g$ via a 2-D convolution equation whose kernel separates as in (27), with functions $\kappa$ and $\omega$ given by

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

Precisely the same 2-D integral equation arises in connection with extrapolation of band-limited signals [32]. Other 2-D convolution problems with separable kernels arise in connection with image restoration problems, and we return to this subject shortly.

### 6.2. *Kronecker products*

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

$$\overline{A} \otimes A = \begin{pmatrix} \overline{a}_{11}A & \overline{a}_{12}A & \dots & \overline{a}_{1\bar{n}}A \\ \overline{a}_{21}A & \overline{a}_{22}A & \dots & \overline{a}_{2\bar{n}}A \\ \vdots & \vdots & & \vdots \\ \overline{a}_{\overline{m}1}A & \overline{a}_{\overline{m}2}A & \dots & \overline{a}_{\overline{m}\bar{n}}A \end{pmatrix}. \tag{28}$$

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

$$\overline{A} \otimes A = \big(\overline{a}_1 \otimes a_1, \dots, \overline{a}_1 \otimes a_n, \overline{a}_2 \otimes a_1, \dots, \overline{a}_{\bar{n}} \otimes a_1, \dots, \overline{a}_{\bar{n}} \otimes a_n\big). \tag{29}$$

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

$$(A \otimes B)^{\mathrm{T}} = A^{\mathrm{T}} \otimes B^{\mathrm{T}} \tag{30}$$

and

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD). \tag{31}$$

Some norm relations for Kronecker products are

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

More details about Kronecker products can be found in [34].

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

$$\mathrm{vec}(X) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{32}$$

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

$$\big(\overline{A} \otimes A\big)\mathrm{vec}(X) = \mathrm{vec}\big(AX\overline{A}^{\mathrm{T}}\big). \tag{33}$$

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

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

then it follows from (29) and (31) that

$$\overline{A} \otimes A = \big(\overline{U} \otimes U\big)\big(\overline{\Sigma} \otimes \Sigma\big)\big(\overline{V} \otimes V\big)^{\mathrm{T}} = \sum_{i=1}^{\bar{n}} \sum_{j=1}^{n} \big(\overline{u}_i \otimes u_j\big)\big(\overline{\sigma}_i \sigma_j\big)\big(\overline{v}_i \otimes v_j\big)^{\mathrm{T}}$$

which, except for the ordering of the singular values and vectors, constitutes the SVD of $\overline{A} \otimes A$. We see that the $n\bar{n}$ 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 computational savings can always 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 problem in (27). 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 $y'_\ell$, $\ell = 1, \ldots, n$, thus arriving at the function $\phi$:

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

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 $\psi$:

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

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,

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

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

$$\begin{aligned} A_{ik} &= n^{-1}\kappa\big(x_i - x'_k\big), & \overline{A}_{j\ell} &= n^{-1}\omega\big(y_j - y'_\ell\big) \\ F_{k\ell} &= \widetilde{f}\big(x'_k, y'_\ell\big), & G_{ij} &= g(x_i, y_j), \end{aligned}$$

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

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

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

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

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:

$$AF\overline{A}^{\mathrm{T}} = G \quad \Longleftrightarrow \quad \big(\overline{A} \otimes A\big)\mathrm{vec}(F) = \mathrm{vec}(G). \tag{34}$$

The rightmost equation is a system of $n^2$ linear equations in $n^2$ unknowns, and corresponds to the linear system $Ax = b$ in the 1-D case. This system is useful for theoretical considerations, while the leftmost expression in (34) is more suited for numerical computations. We note that if both $A$ and $\overline{A}$ are Toeplitz matrices then the coefficient matrix $\overline{A} \otimes A$ is a block Toeplitz matrix with Toeplitz blocks.

### 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[3] – 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 [3]; here we limit our 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 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.

*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

$$K(x, y, x', y') = h_L(x - x') = \begin{cases} (2L)^{-1}, & |x - x'| \leqslant L, \\ 0, & \text{else}, \end{cases} \tag{35}$$

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

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

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 $I$

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

is the identity matrix and $B_\ell$ is a banded Toeplitz matrix of appropriate dimensions with elements given by

$$(B_\ell)_{ij} = \begin{cases} (2\ell)^{-1}, & |i-j| \leqslant \ell, \\ 0, & \text{else}, \end{cases} \tag{36}$$

where the parameter $\ell$ 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. A good model of the point spread function for out-of-focus blur is

$$K(x, y, x', y') = \begin{cases} (\pi R^2)^{-1}, & \sqrt{(x-x')^2 + (y-y')^2} \leqslant R, \\ 0, & \text{else}, \end{cases} \tag{37}$$

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

$$\begin{aligned} K(x, y, x', y') &= \frac{1}{2\pi\sigma\bar{\sigma}} \exp\left(-\frac{1}{2}\left(\frac{x-x'}{\bar{\sigma}}\right)^2 - \frac{1}{2}\left(\frac{y-y'}{\sigma}\right)^2\right), \\ &= \eta_{\bar{\sigma}}(x - x')\eta_\sigma(y - y'), \end{aligned} \tag{38}$$

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

$$(T_\sigma)_{ij} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{i-j}{\sigma}\right)^2\right), \tag{39}$$

then the discretization of the atmospheric turbulence point spread function takes the form $\overline{A} \otimes A = T_{\bar{\sigma}} \otimes T_\sigma$, where the two Toeplitz matrices should have the appropriate dimensions. Depending on the dimensions and the value of $\sigma$, the matrix $T_\sigma$ may be a full matrix or an effectively banded matrix. Figure 18 illustrates this point.

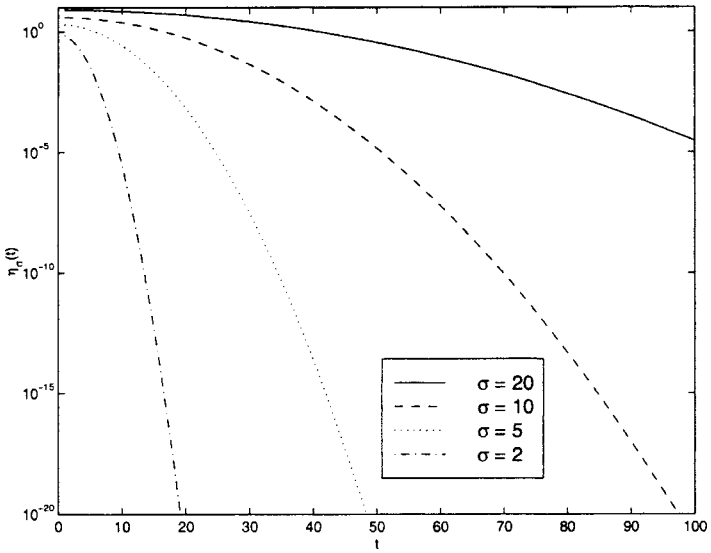


Figure 18. The decay of the function $\eta_\sigma(t)$ in the atmospheric turbulence blur model.

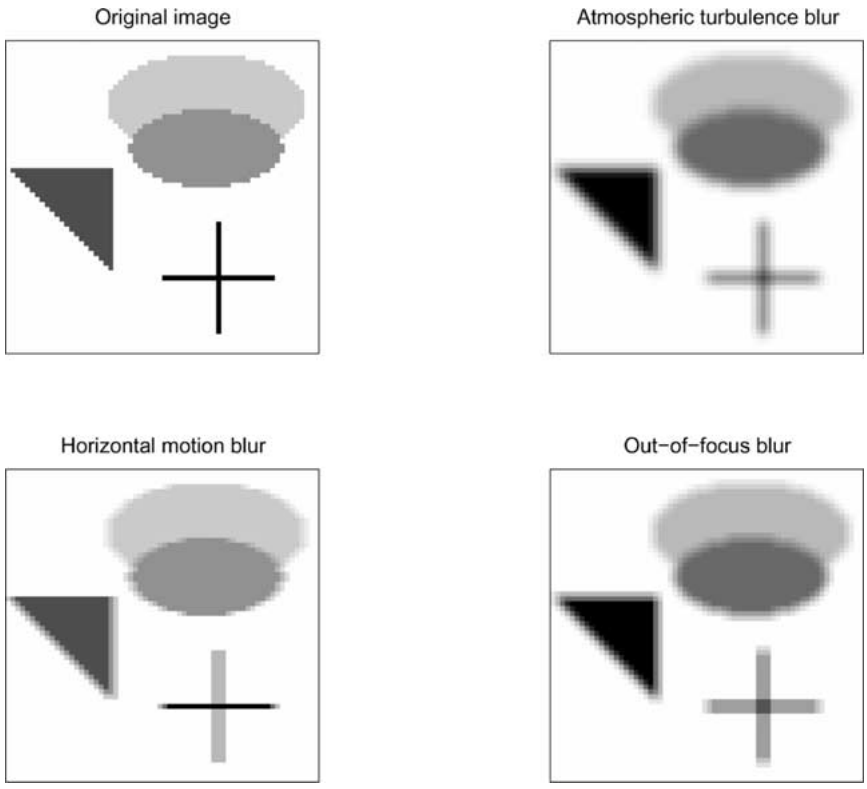


Figure 19. An exact image, and three blurred versions of the same image.

The above three blurring models are illustrated in figure 19 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 [3] to measure the power of an image by first subtracting the image's mean value from all pixels; i.e., if $\mu_F$ 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

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

in which $\Xi$ is an $m \times n$ matrix of all ones. Along this line, if $G_{\mathrm{pure}} = A F \overline{A}^{\mathrm{T}}$ is the "pure" blurred image and $E$ is additive Gaussian noise with zero mean and standard deviation $\sigma_{\mathrm{noise}}$, then the *blurred signal-to-noise ratio* (BSNR) of the blurred noisy image $G = G_{\mathrm{pure}} + E$ is given by

$$\mathrm{BSNR} = 10 \log_{10} \frac{P(G_{\mathrm{pure}})}{P(E)} = 10 \log_{10} \frac{\| G_{\mathrm{pure}} - \mu_{G_{\mathrm{pure}}} \Xi \|_{\mathrm{F}}^2}{mn \sigma_{\mathrm{noise}}^2}. \tag{40}$$

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

$$\mathrm{ISNR} = 10 \log_{10} \frac{\| F - G \|_{\mathrm{F}}^2}{\| F - F_{\mathrm{res}} \|_{\mathrm{F}}^2}. \tag{41}$$

### 6.5. *Regularization with Kronecker products*

In this section we express the TSVD and Tikhonov solutions to the discretized 2-D problem (34) 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 $\mathrm{vec}(F_k)$ 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 $\overline{A} \otimes A$ that are included in the regularized solution. Then $\mathrm{vec}(F_k)$ is given by

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

where the summation is over the $k$ largest quantities $\overline{\sigma}_j \sigma_i$. Using the relation (33) we arrive at the expression

$$F_k = \sum_{i,j} \frac{u_i^{\mathrm{T}} G \overline{u}_j}{\sigma_i \overline{\sigma}_j} v_i \overline{v}_j^{\mathrm{T}} \tag{42}$$

with the same summation as above. Note that we express the 2-D TSVD solution in terms of rank-one matrices $v_i \overline{v}_j^{\mathrm{T}}$. Equation (42) is very convenient for computing 2-D TSVD solutions in terms of the two SVDs of $A$ and $\overline{A}$. 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 $1 \times 1$ matrix with SVD $\overline{u}_1 = \overline{\sigma}_1 = \overline{v}_1 = 1$, and (42) reduces to the standard expression (9) for the 1-D TSVD solution.

We illustrate the 2-D TSVD algorithm with a $100 \times 75$ image deblurring problem. The original image shows J. H. Wilkinson, the founder of rounding error analysis. The blurring matrices $A$ and $\overline{A}$ are generated by means of equation (39) with $\sigma = 2.5$ and then truncated to a total bandwidth of 21 elements. Figure 20 shows the original im-

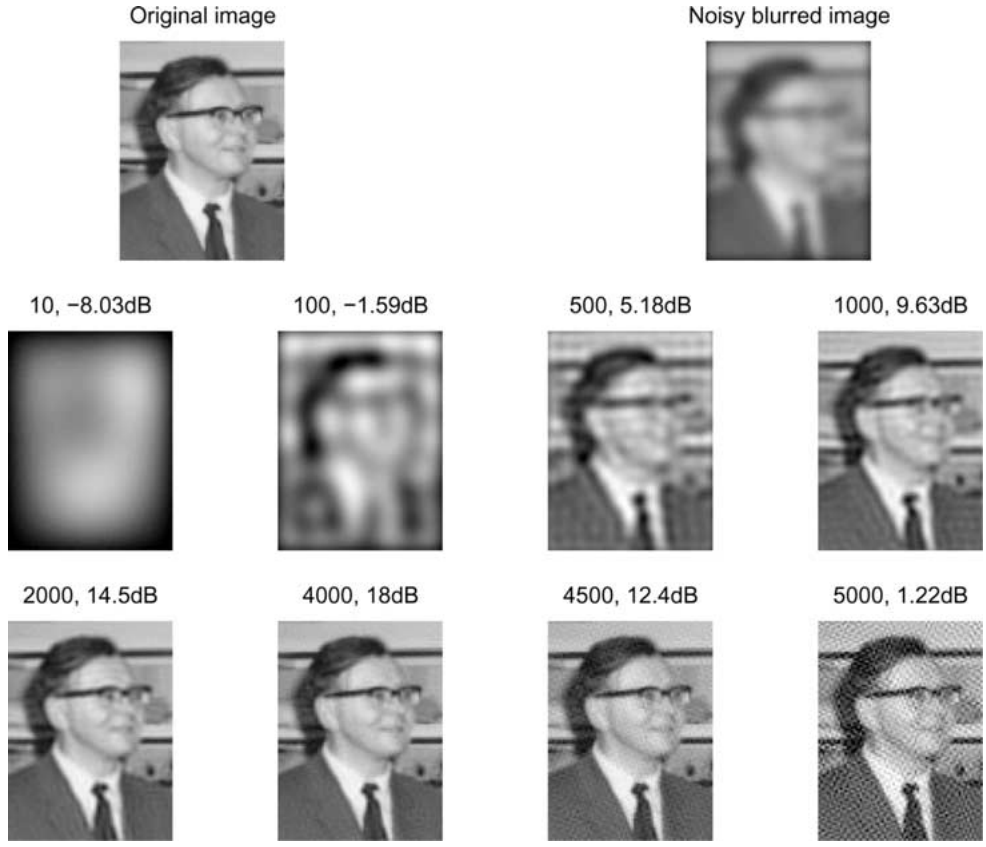


Figure 20. Top row: original and noisy blurred images of dimension $100 \times 75$. Middle and bottom rows: 2-D TSVD solutions $F_k$; the numbers above each image are $k$ and the ISNR.

age $F$, the noisy blurred image $G = A F \overline{A}^T + E$, and eight reconstructed images $F_k$. Above each plot is given $k$ and the corresponding ISNR. Notice how the image quality increases with $k$ until the influence of the noise $E$ becomes visible.

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

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

and

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

and the associated least squares problem is

$$\min \left\| \begin{pmatrix} \overline{A} \otimes A \\ \lambda I \end{pmatrix} \mathrm{vec}(F) - \begin{pmatrix} \mathrm{vec}(G) \\ 0 \end{pmatrix} \right\|_2, \tag{43}$$

where $I$ is the identity matrix of order $n^2$. If $\mathrm{vec}(F_\lambda)$ denotes the Tikhonov solution, where $F_\lambda$ 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

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

and the expression for the 2-D Tikhonov solution becomes

$$F_\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \frac{u_i^{\mathrm{T}} G \overline{u}_j}{\sigma_i \bar{\sigma}_j} v_i \overline{v}_j^{\mathrm{T}}. \tag{44}$$

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 $\mathrm{O}(n^3)$. 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 [13]; the complexity of the algorithm is the same. 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. Direct methods for 2-D problems*

Unfortunately, there is not much hope that we can develop efficient direct algorithms for the general 2-D Tikhonov problem that can exploit the Toeplitz structure, because the above least squares formulation (43) involving $\mathrm{vec}(F)$ cannot be turned into an equivalent linear problem that involves $F$ itself. For example, the corresponding "normal equations" take the form of a generalized Sylvester equation

$$A^{\mathrm{T}} A F \overline{A}^{\mathrm{T}} \overline{A} + \lambda^2 F = A^{\mathrm{T}} G \overline{A}$$

which is not a standard linear system of equations for computing $F$.

The situation is better when $A$ and $\overline{A}$ are symmetric Toeplitz matrices, for then we can use Franklin's method (12) to arrive at the problem

$$\big(\overline{A} \otimes A + \lambda^2 I\big)\mathrm{vec}(F) = \mathrm{vec}(G).$$

For this method, the filter factors that appear in (44) are given by

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

Due to the Toeplitz structure of $A$ and $\overline{A}$, the coefficient matrix $\overline{A} \otimes A + \lambda I$ is a block Toeplitz matrix with Toeplitz blocks (but it is not a Kronecker product), and the system can be solved efficiently by means of a block version of the Levinson algorithm from [25].

As an alternative to the Tikhonov and Franklin solutions, one can use the 1-D algorithms as building blocks for computing the solutions to the two problems

$$\big(A^{\mathrm{T}}A + \lambda^2 I\big)\, F\, \big(\overline{A}^{\mathrm{T}}\overline{A} + \overline{\lambda}^2 I\big) = A^{\mathrm{T}} G \overline{A}$$

and

$$(A + \lambda I)\, F\, (\overline{A} + \overline{\lambda} I) = G.$$

The filter factors for these soltuions are, respectively,

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

and

$$f_{ij} = \frac{\sigma_i \bar{\sigma}_j}{(\sigma_i + \lambda)(\bar{\sigma}_j + \overline{\lambda})}, \quad i, j = 1, \ldots, n.$$

They resemble those for Franklin's method, but they decay in a different way.

### 6.7. *Kronecker products in iterative methods*

In the light of the discussion in the previous section, it is of particular interest to apply iterative methods to the discretized 2-D problem in (34). For example, one step of Landweber's method becomes

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

and by means of (30) and (33) we can rewrite this formula in the following, more convenient form

$$R^{(k-1)} = G - AF^{(k-1)}\overline{A}^{\mathrm{T}}, \qquad F^{(k)} = F^{(k-1)} + \omega A^{\mathrm{T}} R^{(k-1)} \overline{A},$$

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 $\|\mathrm{vec}(X)\|_2 = \|X\|_{\mathrm{F}}$, and one step of the 2-D CGLS algorithm takes the form

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

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

Figure 21 shows eight CGLS solutions $F^{(k)}$ to the same image deblurring problem as in figure 20. Initially the CGLS solutions $F^{(k)}$ are better approximations than the corresponding TSVD solutions $F_k$. But as $k$ increases the influence of finite precision sets in, and after $k = 10000$ iterations the CGLS solution $F^{(k)}$ resembles the TSVD solution with 3000 components.

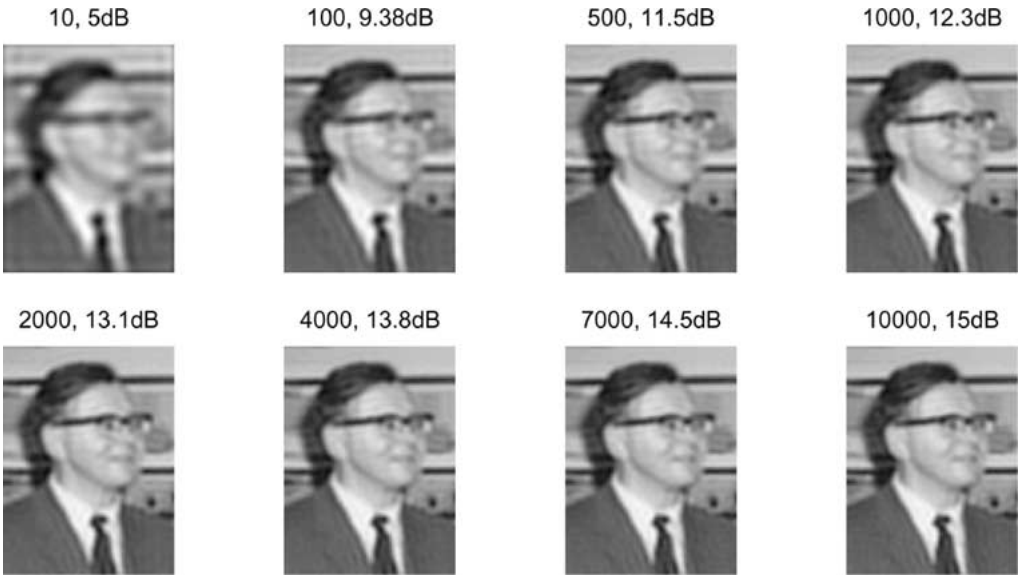


Figure 21. Eight CGLS solutions $F^{(k)}$ to the $100 \times 75$ image deblurring problem. Above each image is shown the iteration count $k$ and the ISNR.

### 6.8. 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 $AX\overline{A}$, 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 for $i = 0, \ldots, m-1$ and $j = 0, \ldots, n-1$ are given by

$$g_{ij} = \sum_{k=0}^{m-1} \sum_{\ell=0}^{n-1} f_{k\ell} h_{(i-k) \bmod m, (j-\ell) \bmod n},$$

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

The matrix $G$ can be computed in less than $2m^2n^2$ 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

$$\mathrm{DFT2}(X) = F_m X F_n. \tag{45}$$

In this expression, we could add transposition to the complex 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

$$G = F_m^{-1}\big((F_m F F_n) \odot (F_m H F_n)\big) F_n^{-1} = \mathrm{IDFT2}\big(\mathrm{DFT2}(F) \odot \mathrm{DFT2}(H)\big).$$

The 2-D DFT (45) 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 $X$ is real (because the intermediate result $F_m X$ is complex), all the $m + n$ DFTs in a single 2-D DFT can be computed by means of the FFT in $n(5m \log_2 m) + m(5n \log_2 n) = 5mn(\log_2 m + \log_2 n)$ 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 $20n^2 \log_2 n$ 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 (26) separates into a product of two functions as in (27). 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,

$$H = vw^{\mathrm{T}} = v \otimes w^{\mathrm{T}}.$$

The elements of $H$ are thus $h_{ij} = v_i w_j$ and we see that $H$ can be considered as a sampling of the function $\kappa(x)\omega(y)$ on a 2-D grid.

To derive the corresponding expression for $G$ we write its elements as

$$g_{ij} = \sum_{k=0}^{m-1} \sum_{\ell=0}^{n-1} f_{k\ell} v_{(i-k) \bmod m} w_{(j-\ell) \bmod n}$$
$$= \sum_{k=0}^{m-1} \left( \sum_{\ell=0}^{n-1} f_{k\ell} w_{(j-\ell) \bmod n} \right) v_{(i-k) \bmod m} = \sum_{k=0}^{m-1} \gamma_{kj} v_{(i-k) \bmod m},$$

where we have introduced $\gamma_{kj} = \sum_{\ell=0}^{n-1} f_{k\ell} w_{(j-\ell) \bmod n}$. These quantities define a matrix $\Gamma$ whose rows are convolutions of $w$ with the rows of $F$, i.e., $\Gamma^{\mathrm{T}} = C_w F^{\mathrm{T}} \Leftrightarrow \Gamma = F C_w^{\mathrm{T}}$. Moreover, the columns of $G$ are convolutions of $v$ with the columns of $\Gamma$, i.e., $G = C_v \Gamma$. We have thus shown that $G$ is given by

$$G = C_v F C_w^{\mathrm{T}}, \tag{46}$$

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

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

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 (46):

$$G = \mathrm{IDFT2}\big(\mathrm{DFT2}(F) \odot \big(\mathrm{DFT}(v)\mathrm{DFT}(w)^{\mathrm{T}}\big)\big). \tag{47}$$

Note that the elementwise matrix multiplication should be computed without forming the outer product of the two DFTs explicitly.

The above relations can now be used to efficiently compute the matrix product $AF\overline{A}^{\mathrm{T}}$ when $A$ and $\overline{A}$ are Toeplitz matrices. The key idea is again to embed 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

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

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 equation (47). The complete amount of work in this approach is approximately $2 \cdot 5 \cdot 2m2n \big(\log_2(2m) + \log_2(2n)\big) \simeq 40mn(\log_2 m + \log_2 n)$ flops – which should be compared with the $2mn(m+n)$ flops in a standard matrix–matrix

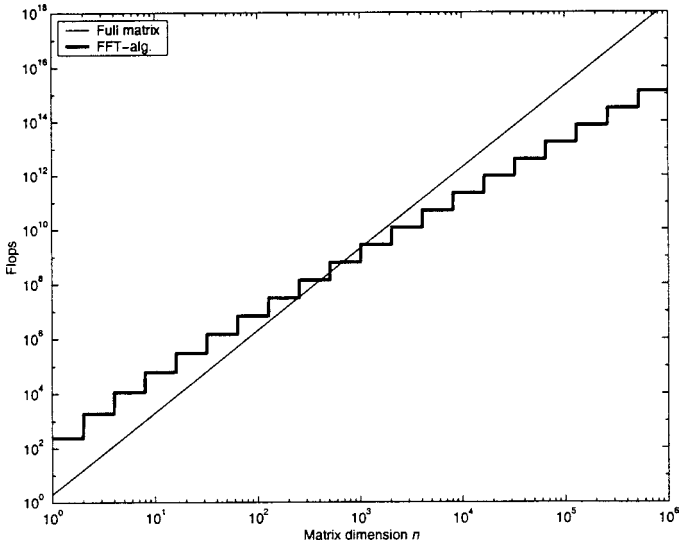


Figure 22. Comparison of the flop count $4n^3$ for ordinary matrix multiplication with the flop count $60n^2 \log_2 n$ for the FFT-based algorithm.

multiplication algorithm for computing $AX\overline{A}^{\mathrm{T}}$. If $m = n$ then the two numbers are $80n^2 \log_2 n$ and $4n^3$. And if special care is taken to exploit the real data and result, then the factor can be reduced to 60. Figure 22 compares the two flop counts.

## References

[1] H.-M. Adorf, Hubble space telescope image restoration in its fourth year, Inverse Problems 11 (1995) 639–653.
[2] C.T.H. Baker, *The Numerical Treatment of Integral Equations* (Clarendon Press, Oxford, 1977).
[3] M.R. Banham and A.K. Katsaggelos, Digital image restoration, IEEE Signal Proc. Mag. 14 (1997) 24–41.
[4] M. Bertero, P. Brianzi and E.R. Pike, Super-resolution in confocal scanning microscopy, Inverse Problems 3 (1987) 195–212.
[5] Å. Björck, *Numerical Methods for Least Squares Problems* (SIAM, Philadelphia, PA, 1996).
[6] A.W. Bojanczyk, R.P. Brent and F.R. de Hoog, QR factorization of Toeplitz matrices, Numer. Math. 49 (1986) 210–221.
[7] W.L. Briggs and V.E. Henson, *The DFT. An Owner's Manual for the Discrete Fourier Transform* (SIAM, Philadelphia, PA, 1995).
[8] R.H. Chan, J.G. Nagy and R.J. Plemmons, FFT-based preconditioners for Toeplitz-block least squares problems, SIAM J. Numer. Anal. 30 (1993) 1740–1768.
[9] T. Chan, An optimal circulant preconditioner for Toeplitz systems, SIAM J. Sci. Statist. Comput. 9 (1988) 766–771.
[10] T.F. Chan and P.C. Hansen, A look-ahead Levinson algorithm for general Toeplitz systems, IEEE Trans. Signal Processing 40 (1992) 1079–1090.
[11] L. Eldén, An efficient algorithm for the regularization of ill-conditioned least squares problems with triangular Toeplitz matrices, SIAM J. Sci. Statist. Comput. 5 (1984) 229–236.

[12] L. Eldén, Numerical solution of the sideways heat equation, in: *Inverse Problems in Diffusion Processes*, eds. H.W. Engl and W. Rundell (SIAM, Philadelphia, PA, 1995).
[13] 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, Department of Mathematics, Linköping University, Sweden (1982).
[14] J.N. Franklin, Minimum principles for ill-posed problems, SIAM J. Math. Anal. 9 (1978) 638–650.
[15] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins Univ. Press, Baltimore, MD, 1996).
[16] M. Hanke and J.G. Nagy, Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques, Inverse Problems 12 (1996) 157–173.
[17] M. Hanke and C.R. Vogel, Two-level preconditioners for regularized inverse problems I: Theory, Numer. Math. 83 (1999) 385–402.
[18] P.C. Hansen, Truncated SVD solutions to discrete ill-posed problems with ill-determined numerical rank, SIAM J. Sci. Statist. Comput. 11 (1990) 503–518.
[19] P.C. Hansen, Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems, Numer. Algorithms 6 (1994) 1–35.
[20] P.C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion* (SIAM, Philadelphia, PA, 1998).
[21] P.C. Hansen, Regularization tools version 3.0 for Matlab 5.2, Numer. Algorithms 20 (1999) 195–196; software and manual is available from Netlib in directory numeralgo/na4.
[22] M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards 49 (1952) 409–436.
[23] P.A. Jansson, ed. *Deconvolution of Images and Spectra*, 2nd ed. (Academic Press, San Diego, 1997).
[24] T. Kailath and A.H. Sayed, eds., *Fast Reliable Algorithms for Matrices with Structure* (SIAM, Philadelphia, PA, 1999).
[25] N. Kalauptsidis, D. Manolakis and G. Carayannis, A family of computationally efficient algorithms for multichannel signal processing – a tutorial review, Signal Processing 5 (1983) 5–19.
[26] M. Kilmer and D.P. O'Leary, Pivoted Cauchy-like preconditioners for regularized solution of ill-posed problems, SIAM J. Sci. Comput. 21 (1999) 88–110.
[27] M. Kilmer and G.W. Stewart, Iterative regularization and MINRES, SIAM J. Matrix Anal. Appl. 21 (1999) 613–628.
[28] J.G. Nagy, R.J. Plemmons and T.C. Torgersen, Iterative image restoration using approximate inverse preconditioning, IEEE Trans. Image Processing 5 (1996) 1151–1162.
[29] H. Park and L. Eldén, Stability analysis and fast algorithms for triangularization of Toeplitz matrices, Numer. Math. 76 (1997) 383–402.
[30] H. Park and L. Eldén, Schur-type methods for solving least squares problems with Toeplitz structure, SIAM J. Sci. Comput. 22 (2000) 406–430.
[31] D.L. Phillips, A technique for the numerical solution of certain integral equations of the first kind, J. Assoc. Comput. Mach. 9 (1962) 84–97.
[32] B.J. Sullivan and B. Liu, On the use of singular value decomposition and decimation in discrete-time band-limited signal extrapolation, IEEE Trans. Acoust. Speech Signal Processing 32 (1984) 1201–1212.
[33] A.N. Tikhonov, Solution of incorrectly formulated problems and the regularization method, Soviet Math. Dokl. 4 (1963) 1035–1038; English translation of Dokl. Akad. Nauk. SSSR 51 (1963) 501–504.
[34] C.F. Van Loan, *Computational Frameworks for the Fast Fourier Transform* (SIAM, Philadelphia, PA, 1992).
[35] G.M. Wing and J.D. Zahrt, *A Primer on Integral Equations of the First Kind* (SIAM, Philadelphia, PA, 1991).