8 The SVD Applied to Signal and Image Deblurring

We will discuss the restoration of one-dimensional signals and two-dimensional gray-scale images that have been contaminated by blur and noise. After an introduction on the representation of images, we describe how the SVD can be applied to restore one-dimensional signals. However, this restoration approach cannot be applied in a straightforward manner to two-dimensional images that have been contaminated by blur and noise due to the large computational burden. We discuss how the problem can be simplified to allow deblurring of two-dimensional gray-scale images with the aid of the SVD.

8.1 The representation of gray-scale images

Gray-scale images are stored as matrices of picture elements, referred to as *pixels*. Each pixel is assigned a gray-level. Pixels are commonly represented by 8 binary digits (bits), say $a_0, a_1, \ldots, a_7$. Then each pixel can represent a nonnegative integer of the form

$$a_0 + 2a_1 + 2^2 a_2 + \ldots + 2^6 a_6 + 2^7 a_7, \qquad a_j \in \{0, 1\}.$$

The largest of these integers is obtained by setting all bits a_j to unity. This gives the value

$$1 + 2 \cdot 1 + 2^2 \cdot 1 + \ldots + 2^6 \cdot 1 + 2^7 \cdot 1 = \sum_{j=0}^{7} 2^j = \frac{2^8 - 1}{2 - 1} = 255.$$

The pixel value 0 represents black and the pixel value 255 represents white. Pixel values between 0 and 255 represent different shades of gray. For instance, the pixel value 240 represents light gray, very close to white.

Example 8.1

The image in Figure 1 is represented by a matrix of 512×512 pixels. Contaminated versions of this image often are used to illustrate the performance of numerical methods for image restoration. The image is one of several test images made available by the Institute of Electrical and Electronics Engineers (IEEE) for comparing the performance of restoration methods. □

The representation of images in terms of their pixels is convenient for analysis and modification of images. However, on the internet images are generally stored using a compressed storage format, such as gif, which requires less space. Images stored in the gif format, as well as in other formats, can be imported to MATLAB with the MATLAB command `imread('filename')`.

[0]Version December 4, 2013

Figure 1: Lena Söderberg.

Example 8.2

The image of Figure 1 is available on the course web page in the file lena.gif. It can be imported into MATLAB with the command X=imread('lena.gif'). Then X is a 512×512 matrix of pixels values. The largest entry of X is 249 and the smallest is 3. □

The MATLAB function imagesc can be used to view a matrix of pixels. The next couple of examples illustrate the use of this function.

Example 8.3

Let X be the matrix of Example 8.2. The MATLAB command imagesc(X) then can be used to display the picture of Figure 1. In order to obtain exactly the same picture a few additional MATLAB commands are required: colormap(gray) to get a gray-scale image, axis('tight') to obtain a square image, and axis off to remove the axes. □

Example 8.4

The MATLAB command $X =$ rand(512) generates a 512×512 matrix with uniformly distributed random entries in the interval $[0, 1]$. Figure 2 shows such a matrix. The function imagesc scales the matrix entries so that the largest entry represents white and the smallest one black. In order to

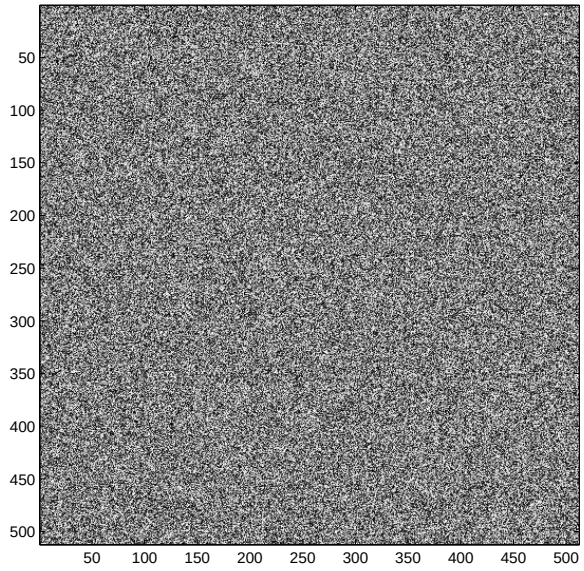


Figure 2: Random noise.

display this image, we issued the MATLAB commands imagesc(X), colormap(gray) to get a gray-scale image, and axis('tight') to make the image square. The axes are labeled with pixel coordinates. □

Example 8.5

Matrices whose entries are the same along every diagonal are known as *Toeplitz matrices*. They can be generated with the MATLAB command toeplitz. This command was used to define the matrix

$$T = \begin{bmatrix} 6 & 2 & 3 & 4 & 5 & 6 \\ 0 & 6 & 2 & 3 & 4 & 5 \\ -1 & 0 & 6 & 2 & 3 & 4 \\ -2 & -1 & 0 & 6 & 2 & 3 \\ -3 & -2 & -1 & 0 & 6 & 2 \\ -4 & -3 & -2 & -1 & 0 & 6 \end{bmatrix}.$$

Toeplitz matrices arise in image and signal processing applications; see Section 8.2 below. Figure 3 is obtained with the MATLAB commands imagesc(T) and axis('tight'). □

Images of stars in the night sky are generally blurred because the light from the stars has to pass through the atmosphere before reaching us. Blurred images also can be obtained by taking pictures with a camera with an out-of-focus lens. Pictures taken of a moving object with too long

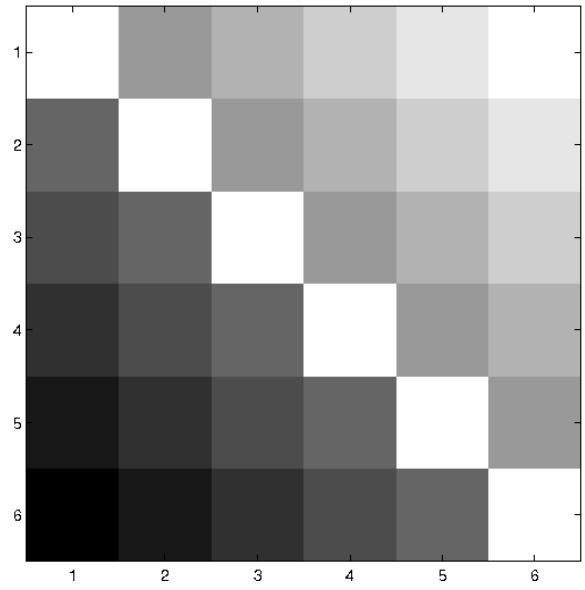


Figure 3: Toeplitz matrix.

exposure time suffer from motion blur. Mathematically, blur arises when pixel values of the original uncontaminated image are replaced by weighted averages of values of nearby pixels. We will in this lecture first discuss the blurring and deblurring of one-dimensional signals. Subsequently blurring and deblurring of gray-scale images will be considered.

8.2 One-dimensional signals

Consider the one-dimensional "signal"

$$\mathbf{u} = [0 \quad \ldots \quad 0 \quad 1/4 \quad 1/2 \quad 1/4 \quad 0 \quad \ldots \quad 0 \quad 1 \quad 0 \quad \ldots \quad 0]^T \in \mathbb{R}^{200}. \tag{1}$$

We are interested in the effect of blurring and subsequent deblurring of this signal. Introduce the "blurring matrix"

$$T = [t_{j,k}] \in \mathbb{R}^{200\times 200}, \qquad t_{j,k} = c\exp(-\frac{1}{10}(j-k)^2), \qquad 1 \leq j,k \leq 200. \tag{2}$$

Thus, T is a symmetric Toeplitz matrix. The constant $c \approx 1/5.6$ is chosen so that the entries of a typical row of T sum to one. Figure 5 displays the entries of this matrix. Most of the "mass" is located near the diagonal.

Multiplication of the signal (1) by the blurring matrix (2) yields the blurred signal $\mathbf{v} = T\mathbf{u}$. Blurring occurs because the entries of the vector $\mathbf{v}$ are weighted averages of nearby entries of $\mathbf{u}$.

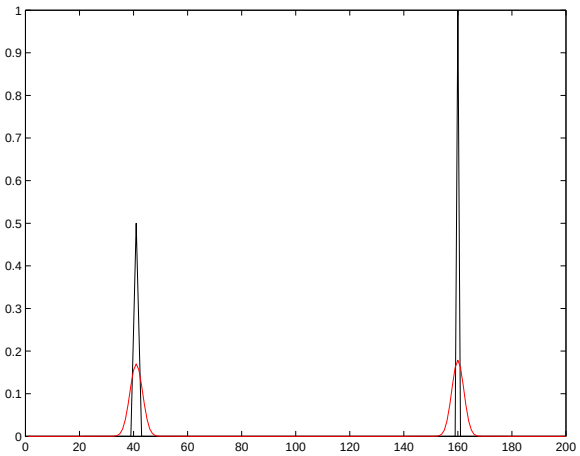


Figure 4: Original signal **u** (black) and blurred signal **v** (red).

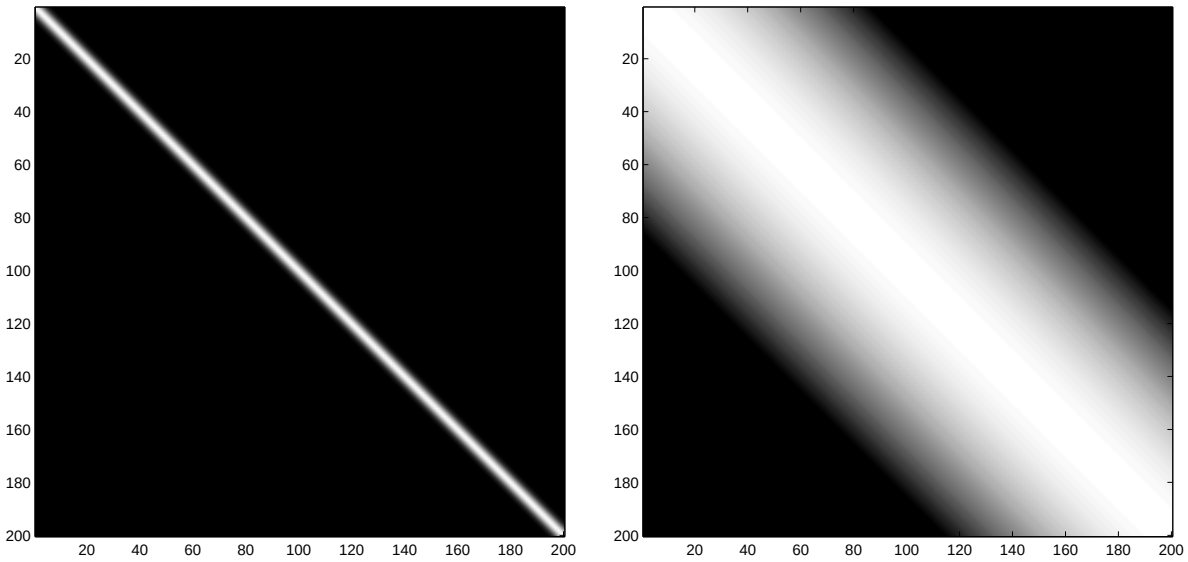


Figure 5: (a) Image of a blurring matrix with the gray-level proportional to the size of the entries, (b) image of the same blurring matrix with the gray-level proportional to the logarithm of the size of the entries.

Figure 4 shows the vectors $\mathbf{u}$ and $\mathbf{v}$. We remark that when looking at stars in the night sky, we observe a two-dimensional analog of the blurred signal $\mathbf{v}$. One-dimensional blurred signals typically stem from the use of poorly calibrated or inaccurate measurement equipment used to capture the signal. Thus, our best information about a measured process may be a blurred signal. In order to gain better understanding of the observed process, we would like to remove the blur. This is known as *deblurring* the signal.

In practice, available measured signals $\mathbf{v}$ are contaminated by blur as well as by noise. The noise stems from the measurement equipment, transmission errors when the signal is transferred from one device to another, and sometimes also from random perturbations in the medium through which we make our observations. For instance, the "twinkling" of stars in the night sky is caused by random fluctuations of the atmosphere. These fluctuations can be considered noise.

Thus, instead of having a blurred signal $\mathbf{v}$ at our disposal, we observe a blurred and noisy signal

$$\mathbf{w} = \mathbf{v} + \mathbf{e}. \tag{3}$$

Here the vector $\mathbf{e}$ represents the noise. In many applications, the noise is white Gaussian; it is normally distributed with zero mean. However, noise with other statistical properties can be found in some applications. For instance, images captured with a digital camera are contaminated by Poisson noise, and so are images of planets and stars taken with large telescopes.

MATLAB allows easy generation of "noise" vectors, whose entries are normally distributed with zero mean, with the command `randn`. This is described in Lecture 7. We recall that the MATLAB commands

$$\mathbf{e} = \mathsf{randn}(200, 1); \qquad \mathbf{e} = \mathbf{e}/\mathsf{norm}(\mathbf{e}); \qquad \mathbf{e} = \mathbf{e} * \mathsf{norm}(\mathbf{u}) * 1\mathsf{e} - 3;$$

yield a vector with 0.1% normally distributed "noise," i.e.,

$$\frac{\|\mathbf{e}\|}{\|\mathbf{u}\|} = 1 \cdot 10^{-3}. \tag{4}$$

Assume that the the blur- and noise-free vector $\mathbf{u}$ is not available. We would like to determine $\mathbf{u}$ from the available blur- and noise-contaminated vector $\mathbf{w}$ defined by (3). We remark that the vector $\mathbf{w}$ when plotted in Figure 4 is visually indistinguishable from the blurred, but noise-free, signal $\mathbf{v}$.

In our first attempt to determine the blur- and noise-free vector, we solve the linear system of equations

$$T\breve{\mathbf{u}} = \mathbf{w} \tag{5}$$

by issuing the MATLAB command

$$\breve{\mathbf{u}} = T\backslash\mathbf{w}. \tag{6}$$

Figure 6 displays the vector $\breve{\mathbf{u}}$ so determined. Note the scale on the vertical axis. Clearly, the vector $\breve{\mathbf{u}}$ has no resemblance to the desired signal $\mathbf{u}$!

The are two major reasons for the poor "restoration" obtained:

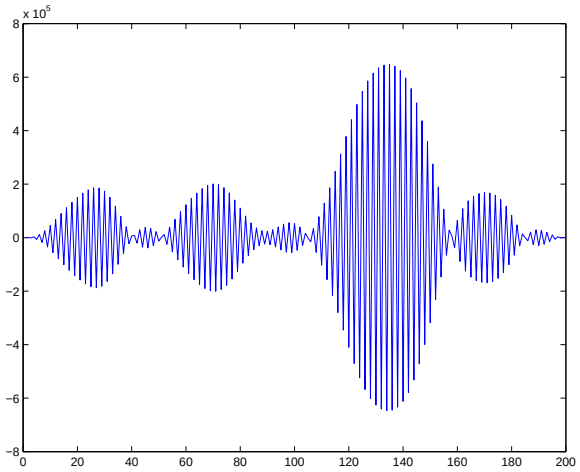


Figure 6: Signal obtained by solving for the deblurred signal with the backslash operation in MATLAB; see (6).

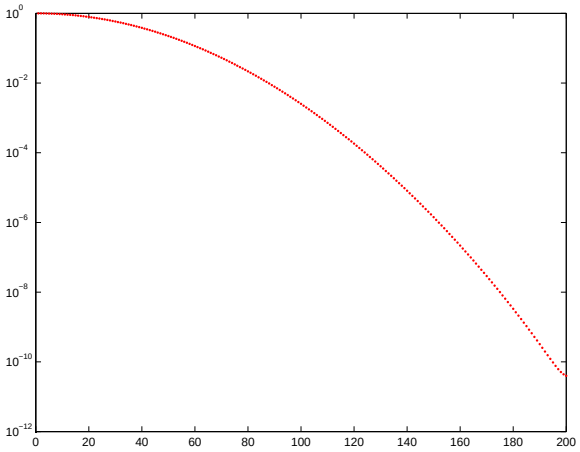


Figure 7: Singular values of the blurring matrix $T \in \mathbb{R}^{200\times 200}$.

- Equation (5) ignores the noise in $\mathbf{w}$. The desired restoration satisfies $T\mathbf{u} = \mathbf{v}$, but we solve (5) with the right-hand side vector defined by (3). Thus, the MATLAB command (6) solves a linear system of equations with the wrong right-hand side!

- The matrix T has "tiny" singular values in relation to its largest singular value. The smallest and largest singular values are $\sigma_{200} = 4.0 \cdot 10^{-11}$ and $\sigma_1 = 1.0$, respectively. The singular values are shown in Figure 7. Because of the large spread of the singular values, the computed solution is sensitive to errors in the right-hand side vector. This has already been illustrated in Lecture 7.

We are interested in determining the influence of the error $\mathbf{e}$ in the right-hand side $\mathbf{w}$ on the computed solution of $T\breve{\mathbf{u}} = \mathbf{w}$. A bound for this error can be derived as follows: subtracting $T\mathbf{u} = \mathbf{v}$ from $T\breve{\mathbf{u}} = \mathbf{w} = \mathbf{v} + \mathbf{e}$ gives $T(\breve{\mathbf{u}} - \mathbf{u}) = \mathbf{e}$. Therefore,

$$\breve{\mathbf{u}} - \mathbf{u} = T^{-1}\mathbf{e}.$$

Here $\breve{\mathbf{u}} - \mathbf{u}$ is the error in the computed restoration $\breve{\mathbf{u}}$ of $\mathbf{u}$. A bound for this error is given by

$$\|\breve{\mathbf{u}} - \mathbf{u}\| = \|T^{-1}\mathbf{e}\| \leq \|T^{-1}\| \, \|\mathbf{e}\|,$$

where the inequality is a consequence of the compatibility requirement (24) of Lecture 1. Moreover, we know from equation (15) of Lecture 7 that $\|T^{-1}\| = 1/\sigma_{200}$, and by (4) we have $\|\mathbf{e}\| = 1 \cdot 10^{-3}\|\mathbf{u}\|$ with

$$\|\mathbf{u}\|^2 = 1 + \left(\frac{1}{2}\right)^2 + 2\left(\frac{1}{4}\right)^2 = 1.375.$$

It follows that

$$\|\breve{\mathbf{u}} - \mathbf{u}\| \leq \frac{1}{\sigma_{200}} 1 \cdot 10^{-3}\|\mathbf{u}\| = 3.0 \cdot 10^{7}.$$

This bound indicates that it, indeed, is possible to obtain a completely useless restoration by solving (5).

In order to compute a useful approximation of the blur- and noise-free signal $\mathbf{u}$, we have to modify our approach. Section 7.5 of Lecture 7 provides a clue for how one may be able to proceed. There we removed oscillations in a computed polynomial approximant by setting some small singular values to zero. The flaw of the "restored" signal $\breve{\mathbf{u}}$ shown in Figure 6 is that it oscillates too much. Therefore, we may be able to achieve a better approximation of $\mathbf{u}$ by first setting some of the smallest singular values of T to zero, and then determining the solution of minimal norm of the least-squares problem so obtained.

To fix ideas, introduce the singular value decomposition

$$T = U\Sigma V^T,$$

consisting of the 200×200 matrices

$$U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{200}], \qquad V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{200}], \qquad \Sigma = \mathrm{diag}[\sigma_1, \sigma_2, \ldots, \sigma_{200}],$$

with $U^T U = V^T V = I$ and $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_{200} > 0$. Define the matrices

$$\Sigma_k = \mathrm{diag}[\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0], \qquad 1 \leq k \leq 200,$$

obtained by setting the last $200 - k$ diagonal entries of Σ to zero, and introduce the associated modifications of T,

$$T_k = U \Sigma_k V^T, \qquad 1 \leq k \leq 200. \tag{7}$$

We would like to determine an approximation of the desired blur- and noise-free signal $\mathbf{u}$ by solving the least-squares problem

$$\min_{\mathbf{x} \in \mathbb{R}^{200}} \|T_k \mathbf{x} - \mathbf{w}\|^2 \tag{8}$$

for a suitable value of k. Substituting (7) into the above expression yields, similarly to equation (18) of Lecture 7,

$$\min_{\mathbf{x} \in \mathbb{R}^{200}} \|U \Sigma_k V^T \mathbf{x} - \mathbf{w}\|^2 = \min_{\mathbf{y} \in \mathbb{R}^{200}} \|\Sigma_k \mathbf{y} - \tilde{\mathbf{w}}\|^2, \tag{9}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_{200}]^T = V^T \mathbf{x}$ and $\tilde{\mathbf{w}} = [\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{200}]^T = U^T \mathbf{w}$. The right-hand side of (9) can be expressed as

$$\min_{\mathbf{y} \in \mathbb{R}^{200}} \sum_{j=1}^{k} (\sigma_j y_j - \tilde{w}_j)^2 + \sum_{j=k+1}^{200} \tilde{w}_j^2. \tag{10}$$

This representation shows that the solution of the minimization problem is

$$y_j = \frac{\tilde{w}_j}{\sigma_j}, \quad 1 \leq j \leq k, \tag{11}$$

with the components y_j for $j > k$ arbitrary. We set the latter to zero. This yields the solution of (10) of minimal Euclidean norm.

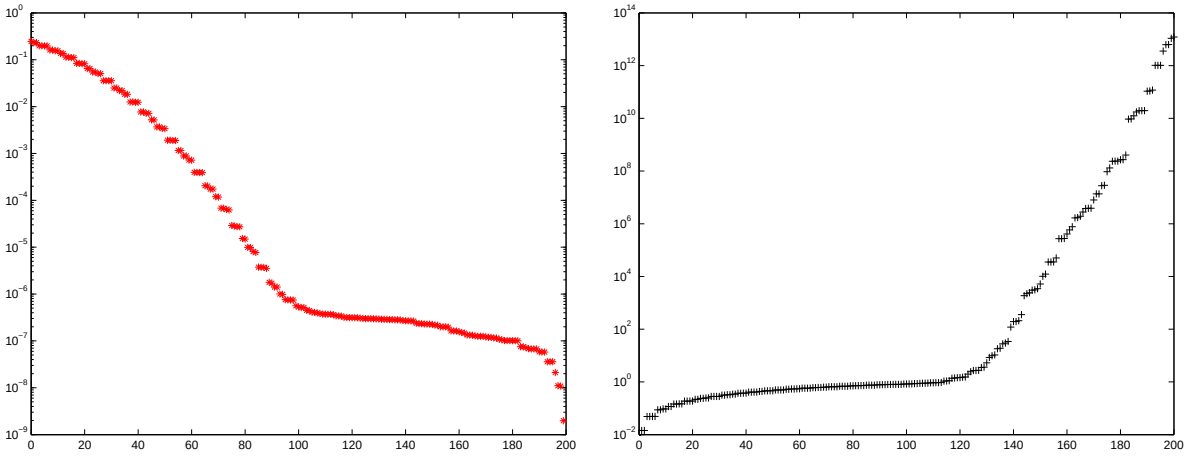
Define the vectors

$$\mathbf{y}_k = [y_1, y_2, \ldots, y_k, 0, \ldots, 0]^T \in \mathbb{R}^{200}, \qquad 1 \leq k \leq 200, \tag{12}$$

and the associated least-squares solutions

$$\mathbf{x}_k = V \mathbf{y}_k, \qquad 1 \leq k \leq 200, \tag{13}$$

of (8). Each of the vectors $\mathbf{x}_k$ is an approximation of the desired blur- and noise-free signal $\mathbf{u}$. We have to determine which one to select. Figure 6 shows the vector $\mathbf{x}_{200}$, which is useless. The vector $\mathbf{x}_1$ also is unlikely to be an accurate approximant of $\mathbf{u}$, since its computation uses very little



(a) (b)

Figure 8: (a) The square of the residual norm (14) as a function of k, (b) the square norm of the minimal norm solution $\mathbf{x}_k$ of (8) as a function of k.

information about the matrix T; only the triplet $\{\sigma_1, \mathbf{u}_1, \mathbf{v}_1\}$ is used. In order to gain some insight into how to select a suitable value of k, we plot the squared norm of the residual error

$$\|T\mathbf{x}_k - \mathbf{w}\|^2 = \|\Sigma\mathbf{y}_k - \tilde{\mathbf{w}}\|^2 = \sum_{j=k+1}^{200} \tilde{w}_j^2 \tag{14}$$

as a function of k. Figure 8(a) displays this function in logarithmic scale. The graph shows the residual norm to plateau at about $k = 110$. Figure 8(b) plots the square norm of the minimal norm solutions $\mathbf{x}_k$ of the least-squares problems (8) as a function of k in a logarithmic scale. The norm of $\mathbf{x}_k$ increases with k; see Exercise 8.2. The rapid increase for $k \geq 110$ indicates that the vectors $\mathbf{x}_k$ for these k-values are likely to contain substantial propagated noise (stemming from the error $\mathbf{e}$) and therefore are poor approximations of the blur- and noise-free signal $\mathbf{u}$. Both graphs of Figure 8 suggest that $\mathbf{x}_{110}$ may be a fairly accurate approximation of $\mathbf{u}$.

It is also illustrative to study the graph of the points

$$\{\|\mathbf{x}_k\|^2, \|T\mathbf{x}_k - \mathbf{w}\|^2\}, \qquad k = 1, 2, \ldots, 200, \tag{15}$$

in log-log scale. Figure 9(a) displays this graph. The circles ∘ at the top-left of the figure correspond to points (15) with small k-values; circles ∘ associated with large k-values can be found at the lower-right of the figure. Due to its shape, the graph is commonly referred to an *L-curve*. Least-squares solutions $\mathbf{x}_k$ corresponding to circles ∘ close to the "vertex" of the L-curve are likely to be fairly accurate approximations of $\mathbf{u}$, because increasing k further gives least-squares solution $\mathbf{x}_k$ of larger

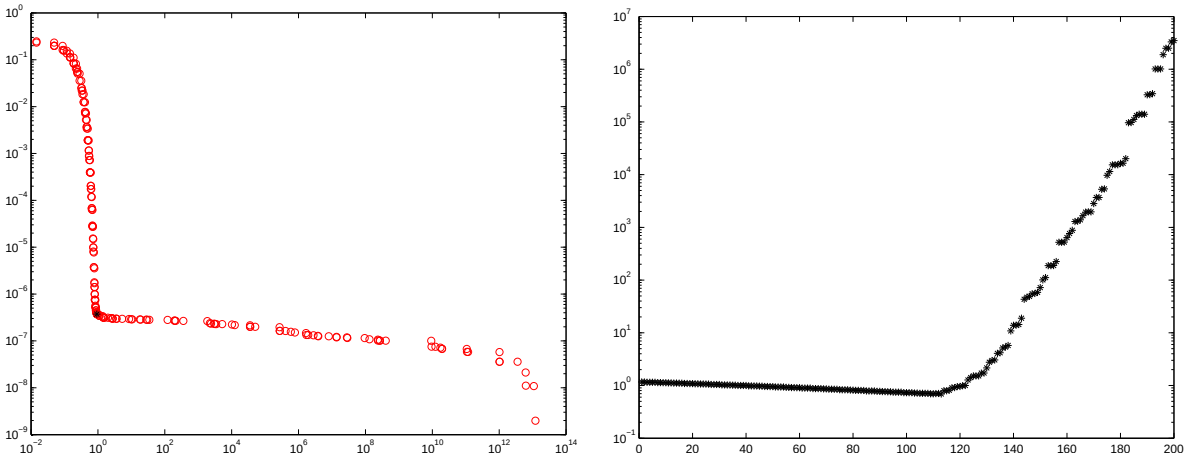


Figure 9: (a) The L-curve made up by the points (15), (b) errors (16) as a function of k.

norm without reducing the the residual error $\|T\mathbf{x}_k - \mathbf{w}\|$ substantially. This, suggest that the increase in norm of the $\mathbf{x}_k$ with k primarily stems from propagated error. Thus, we would like to choose a least-squares solution $\mathbf{x}_k$, such that the associated point $\{\|\mathbf{x}_k\|^2, \|T\mathbf{x}_k - \mathbf{w}\|^2\}$ corresponds to a circle $\circ$ close to the vertex. The black $*$ at the vertex of Figure 9(a) corresponds to the point $\{\|\mathbf{x}_{110}\|^2, \|T\mathbf{x}_{110} - \mathbf{w}\|^2\}$.

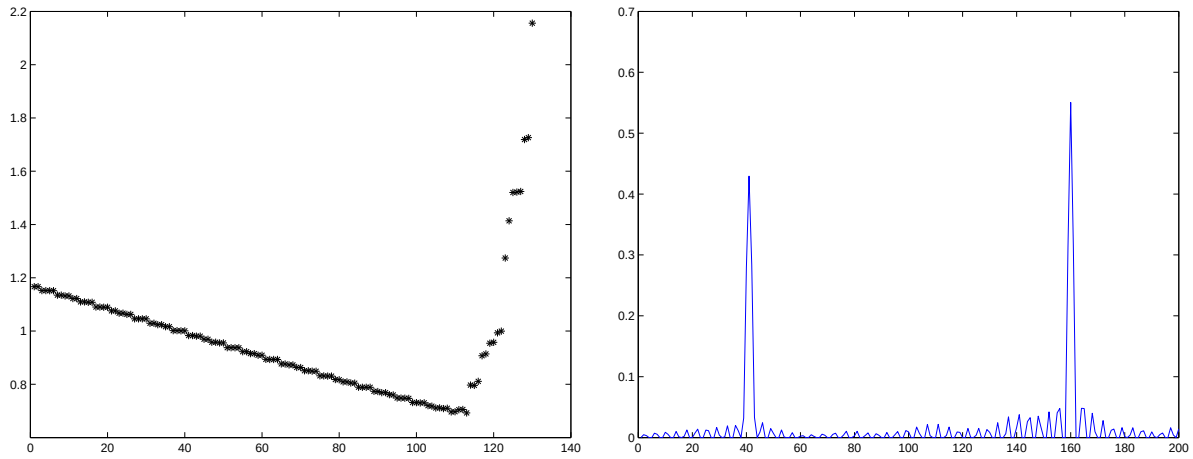
Figure 9(b) displays the norm of the errors

$$\|\mathbf{u} - \mathbf{x}_k\|, \qquad k = 1, 2, \ldots, 200. \tag{16}$$

The figure indicates that, indeed, $\mathbf{x}_{110}$ is a near-best approximation of $\mathbf{u}$. The fast increase of the error with k, when k is larger than its optimal value is illustrated by Figure 10(a), which shows part of Figure 9(b) with linear scaling of the vertical axis. It is obvious from Figures 9(b) and 10(a) that the determination of a value of k, which is not too large, is important.

The computed solution $\mathbf{x}_{110}$ is displayed in Figure 10(b) after setting the negative entries to zero. The latter is justified because we know that the desired signal is nonnegative. The locations of the peaks of the restored signal $\mathbf{x}_{110}$ are accurate and so is the height of the left peak; the right peak, however, is not high enough. We remark that in real signal processing applications, Figure 10(a) of course is not available.

The L-curve is a valuable tool for determining a suitable approximation of an unavailable blur- and noise-free signal from an available blur- and noise-contaminated version. The L-curves works particularly well when there is substantial noise in the available contaminated signal.



(a) (b)

Figure 10: (a) The errors (16) as a function of k, (b) restored signal $\mathbf{x}_{110}$.

Exercise 8.1

Show that the residual error $\mathbf{r}_k = \mathbf{w} - T\mathbf{x}_k$ decreases in norm as k increases. Hint: See (14). □

Exercise 8.2

Show that the least-squares solutions $\mathbf{x}_k$ of (8) increase in norm with k. □

Exercise 8.3

Consider the "exact" signal

$$\mathbf{u} = [u_1, u_2, \ldots, u_{200}]^T, \qquad u_k = \sin\left(\pi \frac{k-1}{199}\right).$$

Blur this signal with the blurring matrix (2) and add 1% noise to obtain the contaminated signal $\mathbf{w}$. Determine an approximation of $\mathbf{u}$ using the L-curve. Plot also the computed solutions $\mathbf{x}_k$ of (8) for larger and smaller values of k than suggested by the L-curve. How do the computed approximations of $\mathbf{u}$ compare? □

8.3 The Pseudoinverse

It is sometimes convenient to express the minimal norm solution of the least-squares problem (8) in terms of a generalized inverse of T_k, similarly as the solution of a linear system of equations with a square nonsingular matrix can be expressed in terms of the inverse of the matrix.

Combining the formulas (8)-(13) shows that

$$\mathbf{x}_k = V\Sigma_k^\dagger U^T\mathbf{w},$$

where

$$\Sigma_k^\dagger = \mathrm{diag}[\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_k^{-1}, 0, \ldots, 0].$$

The matrix $\Sigma_k^\dagger$ is known as the Moore-Penrose pseudoinverse of Σ_k, or briefly as the pseudoinverse of Σ_k. Define

$$T_k^\dagger = V\Sigma_k^\dagger U^T. \tag{17}$$

Then

$$\mathbf{x}_k = T_k^\dagger\mathbf{w}.$$

The matrix $T_k^\dagger$ is referred to as the (Moore-Penrose) pseudoinverse of T_k. Thus, multiplying the data vector, here $\mathbf{w}$, of a least-squares problem by the pseudoinverse yields the least-squares solution of minimal norm.

We remark that the Moore-Penrose pseudoinverse of a matrix $A \in \mathbb{R}^{m\times n}$ can be defined independently of the SVD of A. The matrix $A^\dagger$ is said to be the Moore-Penrose Pseudoinverse of A if it satisfies the four properties:

$$\begin{array}{ll} AA^\dagger A = A, & A^\dagger AA^\dagger = A^\dagger, \\ (AA^\dagger)^T = AA^\dagger, & (A^\dagger A)^T = A^\dagger A. \end{array} \tag{18}$$

These properties determine $A^\dagger$ uniquely. We will apply the pseudoinverse in the following section.

Exercise 8.4

Show that the matrix $T_k^\dagger$ given by (17) satisfies the four properties (18). □

8.4 Two-dimensional signals: better than Photoshop

This section considers deblurring of gray-scale images. For definiteness, we focus on the image of Figure 1. Let the matrix $X \in \mathbb{R}^{256\times 256}$ represent this image, cf. Example 8.2. Assume that the blurring process acts in the horizontal and vertical directions only, and that the blurring is the same in both directions. Then the blurred image can be represented by

$$Y = TXT, \tag{19}$$

where the symmetric matrix $T \in \mathbb{R}^{256\times 256}$ is the blurring operator. The blur is chosen to make the computations simple. However, we remark that in many applications of image deblurring, the blurring can be approximated well by an expression of the form

$$Y = T_1XT_2. \tag{20}$$

Our restoration method easily can be applied to restore images with the latter kind of blur.

Let the available image also be contaminated by noise. We represent the noise by the matrix $E \in \mathbb{R}^{256\times 256}$ with normally distributed random entries with zero mean. The available blur- and noise-contaminated image is given by

$$Z = TXT + E. \tag{21}$$

Our task is to determine an accurate approximation of X from Z. Similarly as in Section 8.2, the blurring matrix T has tiny singular values, which should be set to zero. Let T_k be the rank-k approximation of T obtained by setting all but the first k singular values to zero. Consider the approximations

$$X_k = T_k^\dagger Z T_k^\dagger, \qquad k = 1, 2, \ldots \ . \tag{22}$$

They can be computed fairly easily by computing the SVD of T. We do not explicitly form the matrices T_k for different values of k, but use the representation (17). A suitable value of k can be chosen by visual inspection with the aid of the L-curve.

We remark that color images can be deblurred in the same manner as gray-scale images. However, a color image requires three times as much storage space as a gray-scale image with the same resolution. For each pixel three "channels" are provided to represent the colors red, green, and blue.

The following exercises require the file lena.gif and MATLAB/Octave m-files for three common kinds of blur: Gaussian, out-of-focus, and linear motion blur. They can be found on the course web page.

Exercise 8.5

Let $T = \text{gblur}(512,48,6)$, and compute a blurred image Y of Lena using equation (19). Display the blurred image. □

Exercise 8.6

Add 1% normally distributed noise to the blurred image to determine the image Z; cf. (21). □

Exercise 8.7

Try to recover the original image by "inverting" the blur:

$$\breve{X} = T^\dagger Z T^\dagger.$$

Display your result. Explain your findings. □

Exercise 8.8

(Optional): If you have access to Photoshop or Gimp, then save the blurred and noisy image represented by the vector $\mathbf{w}$ and use the "Unsharp Mask" filter to sharpen the image as best you can. This is the best available tool that Photoshop has for deblurring. □

Exercise 8.9

We can of course do better than Photoshop by using the SVD of the blurring matrix T. Determine a suitable value of k in (22), e.g., by visual inspection or by using the L-curve and determine an approximation X_k of X similarly as in Subsection 8.2. □

Exercise 8.10

Assume that the blurring is only in one direction, i.e.,

$$Y = TX,$$

and let the available image be contaminated by 1% noise. Restore the available image similarly as described above. □

Exercise 8.11

Blur the original image by out-of-focus blur and 1% noise and try to restore it as described above. Use the MATLAB function oblur.m with bandwidth= 5 to define the blurring matrix T. □