Chapter I
Introduction

Skills to acquire

- **How to represent images as matrices and vectors.**
- **How to model some fundamental image processing problems as linear systems and optimization problems: denoising, deconvolution, inpainting, approximation, compression.**
- **Understand the PSF function.**
- **Manipulate fundamental matrix properties and operations: norms, factorization, SVD, Differentiation.**

Many practical problems in applied mathematics can be formulated as optimization problems. We can cite image reconstruction and restoration(denoising, inpainting, deconvolution), supervised/unsupervised learning, parameter estimation, statistical inference, ... In this book, we consider only **finite dimensional** optimization problems, in which images are represented as matrices or vectors.

1 Image representation

A natural representation is to consider images having m rows and n columns as matrices of the space $X \in \mathbb{R}^{m\times n}$. But, it is usually convenient to transform images into vectors by stacking the columns of the images:

$$\boldsymbol{x} = \mathbf{vec}(X),\ e.g.,\quad \mathbf{vec}\left(\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{bmatrix}\right) = \begin{bmatrix}1\\2\\3\\4\\11\\22\end{bmatrix}.$$

With vector representation, entries of the matrix X are determined with *linear indices*. For instance, the linear index of 11 is 5.

Bases

Considered as vectors, images belong to the space $\mathbb{R}^N$, where $N = m \times n$. Thus, we can also write:

$$\boldsymbol{x} = \sum_{i=1}^{N} x_i \boldsymbol{e}_i,$$

where $(\boldsymbol{e}_i)_i$ is the canonical basis of $\mathbb{R}^N$.

Images can also be represented in more appropriate bases, frames, or dictionaries $(\boldsymbol{\psi}_i)_{i \in I}$:

$$\boldsymbol{x} = \sum_{i \in I} x_i \boldsymbol{\psi}_i.$$

For instance, the discrete 2-D Fourier atoms $(\boldsymbol{\psi_k})_{\boldsymbol{k}}$ are defined as:

$$\boldsymbol{\psi_k}(x) = \frac{1}{\sqrt{N}} e^{2i\pi(x_1 \frac{k_1}{m} + x_2 \frac{k_2}{n})},$$

where $\boldsymbol{k} = (k_1, k_2)$ are the frequency indexes, with $0 \leqslant k_1 < m$, $0 \leqslant k_2 < n$. Figures (I.1) and (I.2) show some basis functions of the Fourier and Wavelets bases.

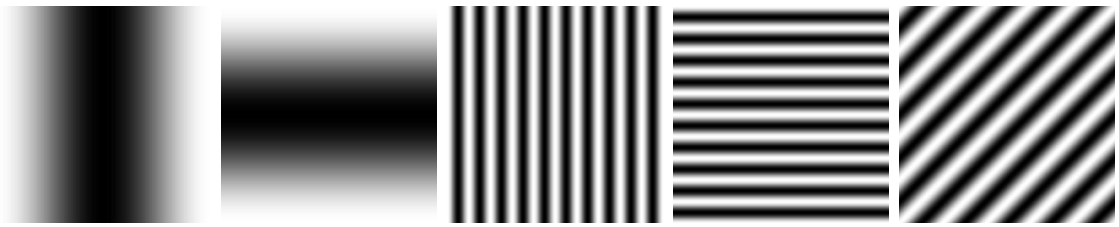


Fig. I.1: Some functions of a Fourier basis.

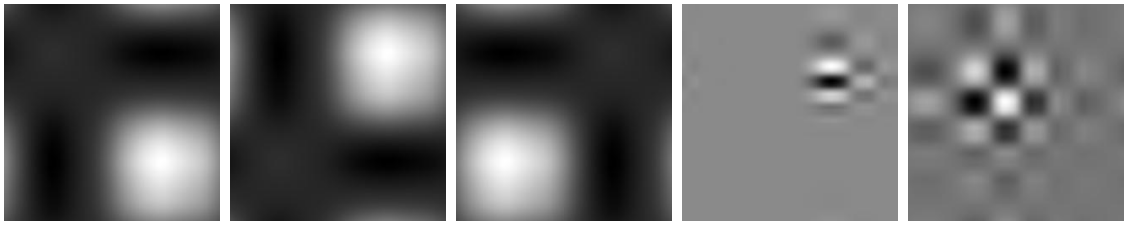


Fig. I.2: Some functions of a Wavelet (Daubechies) basis.

Figure (I.3) shows examples of reconstructions using the Fourier basis. In the Fourier decomposition,small coefficients bellow a chosen threshold are neglected. This can be considered as an **image compression** as we obtained a good image representation with 6554 coefficients only (13108 coefficients when adding also their linear indices), instead of $256 \times 256 = 65536$.

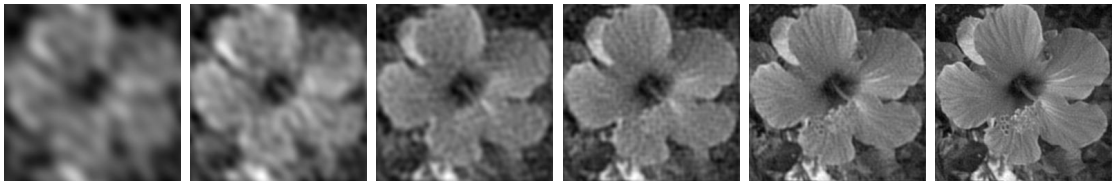


Fig. I.3: Image reconstruction using Fourier decomposition. From right to left, reconstructions with increasing number of coefficients, 109, 218, 655, 1311, 6554, chosen in descending order.

Convolution with vector representation

Convolution is an important operation frequently used in image processing. In the following, we give explicit formula of 2D Convolution computed with the 1D vector representation of images.

One-Dimensional convolution

We first consider one-dimensional convolution $\boldsymbol{b} = \boldsymbol{p} * \boldsymbol{x}$. For the sake of example, we consider $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^5$. It is easy to see that the vector $\boldsymbol{b}$ can be written as a matrix product:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_5\ p_4\ p_3\ p_2\ p_1 & & & & \\ & p_5\ p_4\ p_3\ p_2\ p_1 & & & \\ & & p_5\ p_4\ p_3\ p_2\ p_1 & & \\ & & & p_5\ p_4\ p_3\ p_2\ p_1 & \\ & & & & p_5\ p_4\ p_3\ p_2\ p_1 \end{bmatrix} \begin{bmatrix} w1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline y_1 \\ y_2 \end{bmatrix},$$

where the extra-variables (w_i and y_i) are chosen to fit the boundary conditions (**BC**). More precisely, for **zero BC**, we shall choose $w_i = y_i = 0$. Thus, we get:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \underbrace{\begin{bmatrix} p_3\ p_2\ p_1 & & & & \\ p_4\ p_3\ p_2\ p_1 & & & & \\ p_5\ p_4\ p_3\ p_2\ p_1 & & & & \\ & p_5\ p_4\ p_3\ p_2 & & & \\ & & p_5\ p_4\ p_3 & & \end{bmatrix}}_{\text{Toeplitz matrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}.$$

For **periodic BC**, we shall choose $w_1 = x_4, w_2 = x_5, y_1 = x_1, y_2 = x_2$. We get:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \underbrace{\begin{bmatrix} p_3 & p_2 & p_1 & p_5 & p_4 \\ p_4 & p_3 & p_2 & p_1 & P_5 \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ p_1 & p_5 & p_4 & p_3 & p_2 \\ p_2 & p_1 & p_5 & p_4 & p_3 \end{bmatrix}}_{\text{circulant matrix}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}.$$

For **reflexive BC**, we have $w_1 = x_2, w_2 = x_1, y_1 = x_5, y_2 = x_4$ and

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_3 + p_4 & p_2 + p_5 & p_1 & & \\ p_4 + p_5 & p_3 & p_2 & p_1 & \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ & p_5 & p_4 & p_3 & p_2 + p_1 \\ & & p_5 & p_4 + p_1 & p_3 + p_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}.$$

Two-Dimensional convolution

The convolution operation for two-dimensional images is similar to the one-dimensional case. For example, let

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}; \quad \mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}.$$

For the two-dimensional convolution $\mathbf{B} = \boldsymbol{b} * \mathbf{B}$ and with zero boundary conditions we obtain:

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \hline b_{12} \\ b_{22} \\ b_{32} \\ \hline b_{13} \\ b_{23} \\ b_{33} \end{bmatrix} = \underbrace{\left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & & p_{21} & p_{11} & & & & \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & & & \\ & p_{32} & p_{22} & & p_{31} & p_{21} & & & \\ \hline p_{23} & p_{13} & & p_{22} & p_{12} & & p_{21} & p_{11} & \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ & p_{33} & p_{23} & & p_{32} & p_{22} & & p_{31} & p_{21} \\ \hline & & & p_{23} & p_{13} & & p_{22} & p_{12} & \\ & & & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ & & & & p_{33} & p_{23} & & p_{32} & p_{22} \end{array}\right]}_{\text{Toeplitz matrix with Toeplitz blocks (BTTB)}} \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \hline x_{12} \\ x_{22} \\ x_{32} \\ \hline x_{13} \\ x_{23} \\ x_{33} \end{bmatrix},$$

2 Matrices

2.1 Norms

The standard norms on $\mathbb{R}^n$ are the ℓ_p-norms ($p \geqslant 1$).

The *dual norm* $\|\cdot\|_*$ of a norm $\|.\|$ is defined on $\mathbb{R}^n$ as follows:

$$\|\boldsymbol{x}\|_* = \sup_{\boldsymbol{y} \neq 0} \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{y}\|} = \sup_{\|\boldsymbol{y}\|=1} \boldsymbol{x}^T \boldsymbol{y}.$$

Dual norms satisfy a generalized Cauchy-Schwarz inequality

$$\left|\boldsymbol{x}^T\boldsymbol{y}\right| \leq \|\boldsymbol{x}\|_*\|\boldsymbol{y}\|; \quad \forall\, \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n.$$

Dual norms are also defined for matrices:

$$\|A\|_{p,q} = \sup_{x\in\mathbb{R}^n, x\neq 0} \frac{\|Ax\|_p}{\|x\|_q}.$$

For example, we get the following norms.

$$\|A\|_{1,1} = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}|\,, \quad \|A\|_{\infty,\infty} = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|\,,$$

$$\|A\|_{2,2} = \sqrt{\rho(A^*A)} = \sqrt{\rho(AA^*)} = \sigma_{max}(A),$$

where ρ denotes the spectral radius and σ_{max} the largest singular value of A.

We can also consider *mixed norms*:

$$\|A\|_{\ell_{p,q}} = \left(\sum_{j=1}^{n}\left(\sum_{i=1}^{n}|a_{ij}|^p\right)^{q/p}\right)^{1/q}.$$

For example, we have the Frobenius norm:

$$\|A\|_F = \|A\|_{\ell_{2,2}} = \left(\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2\right)^{1/2} = \sqrt{tr(A^*A)} = \sqrt{tr(AA^*)}.$$

2.2 Differentiation

Let $\boldsymbol{x}$ and $\boldsymbol{c}$ are $(n\times 1)$ vectors sand A an $(m\times n)$ matrix. Then $\boldsymbol{c}^T\boldsymbol{x}$ and $A\boldsymbol{x}$ are scalar and vector functions of $\boldsymbol{x}$, respectively, and we have

$f(\boldsymbol{x})$	$\nabla f(\boldsymbol{x})$
$\boldsymbol{c}^T x$	$\boldsymbol{c}$
$A\boldsymbol{x}$	A
$\boldsymbol{x}^T A\boldsymbol{x}$	$\boldsymbol{x}^T(A^T+A)$
$Tr(XA)$	A^T
$Tr(X^TA)$	A
$Tr(X^TAX)$	$(A+A^T)X$

2.3 Singular value decomposition (SVD)

The *singular value decomposition (SVD)* is a factorization of a real or complex matrix that generalizes the eigendecomposition Any real matrix X of size $n \times p$ can be decomposed as

$$\underbrace{X}_{n\times p} = \underbrace{U}_{n\times n}\,\underbrace{\Sigma}_{n\times p}\,\underbrace{V^T}_{p\times p}, \tag{I.1}$$

- U (respect. V) is orthogonal; its columns are normalized eigenvectors of XX^T (respec. X^TX).
- The columns $\mathbf{u}_j$ and $\mathbf{v}_j$ of the matrices U and V respectively share the same eigenvalue λ_j.
- $\sigma_j = \sqrt{\lambda_j}$ are the singular values, $\sigma_1 \geqslant \cdots \geqslant \sigma_r$, $k = \min\{m, n\}$, and we have

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & \bigcirc & \end{pmatrix}, \quad \text{or} \quad \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \bigcirc \\ & & \sigma_r \end{pmatrix}.$$

- Every matrix X of rank s has exactly s nonzero singular values.

The SVD has the following geometric interpretation. The matrix X can be seen as a linear transformation which can be decomposed in three sub-transformations: 1. rotation U, 2. re-scaling D, 3. rotation V.

The decomposition (I.1) is called *Full SVD*. Truncating the full SVD till the rank s, we can write a *compact* or *reduced SVD* as

$$X = U_1\Sigma_1 V_1^T = \sum_{i=1}^{s} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T. \tag{I.2}$$

where $U_1 = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_s]$ and $V_1 = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_s]$.

2.3.1 Computing the compact SVD

It is computationally inefficient to compute a full SVD. Algorithm 1 explains the steps to compute a compact SVD.

Example 1 *Compute the SVD of* $A = \begin{bmatrix} 2 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & -2 & 0 \end{bmatrix}$

- a. Find the eigenvalues of A^TA and the singular values for A.

$$\det(A^TA - \lambda I) = \begin{vmatrix} 8-\lambda & 2 & 0 \\ 2 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{vmatrix} = \lambda(\lambda-4)(\lambda-9)$$

Thus, $\lambda_1 = 9$, $\lambda_2 = 4$ and $\lambda_3 = 0$. So the singular values are 3,2 and 0.

Algorithm 1: Computing the compact SVD

Data: Matrix A

/* Calculate the eigenvalues and eigenvectors of $A^{\mathsf{H}}A$. */

1 $\boldsymbol{\lambda}, V \leftarrow \text{eig}(A^{\mathsf{H}}A)$

/* Calculate the singular values of A. */

2 $\boldsymbol{\sigma} \leftarrow \sqrt{\boldsymbol{\lambda}}$

/* Sort the singular values **from greatest to least**. */

3 $\boldsymbol{\sigma} \leftarrow \text{sort}(\boldsymbol{\sigma})$

/* Sort the eigenvectors **the same way**. */

4 $V \leftarrow \text{sort}(V)$

/* Keep only the positive singular values. */

5 $\boldsymbol{\sigma}_1 \leftarrow \boldsymbol{\sigma}_{:r}$

/* Keep only the corresponding eigenvectors. */

6 $V_1 \leftarrow V_{:,:r}$

/* Construct U colum vy column (Can we compute full SVD?). */

7 $U_1 \leftarrow AV_1/\boldsymbol{\sigma}_1$

/* Output */

8 ***return*** $U_1, \boldsymbol{\sigma}_1, V_1^{\mathsf{H}}$

$$\Sigma = \begin{bmatrix} 3\,0\,0 \\ 0\,2\,0 \\ 0\,0\,0 \end{bmatrix}$$

- b. Find $V = (e_1, e_2, e_3)$. e_j is the unit vector corresponding to the jth eigenvalue.

$$V = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Find U by using the formula $AV = U\Sigma$. We can write $\boldsymbol{u}_j = \frac{1}{\sigma_j} A\boldsymbol{v}_j$

  Finally, we get:

$$A = U\Sigma V^T = \begin{bmatrix} \frac{4}{3\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ \frac{\sqrt{5}}{3} & 0 & 0 \\ -\frac{2}{3\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \end{bmatrix} \begin{bmatrix} 3\,0\,0 \\ 0\,2\,0 \\ 0\,0\,0 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.3.2 Applications

Solving linear systems In the next chapter, we will show how the SVD can be used to solve linear systems in various situations.

Computing Pseudoinverse For $A \in \mathbb{R}^{m \times n}$, a *pseudoinverse* of A is defined as a matrix $A^+ \in \mathbb{R}^{n \times m}$ satisfying the following Moore-Penrose conditions:

$$AA^+A = A,\ A^+AA^+ = A^+,\ (AA^+)^* = AA^+,\ (A^+A)^* = A^+A.$$

The pseudoinverse A^+ exists for any matrix A, but when the latter has full rank, A^+ can be expressed using simple algebraic formula.

When A has linearly independent columns (A^*A is invertible), A^+ constitutes a left inverse ($A^+A = I$) and we can write:

$$A^+ = (A^*A)^{-1}A^*.$$

On the other hand, when A has linearly independent rows (AA^* is invertible), A^+ is a right inverse and we have:

$$A^+ = A^*(AA^*)^{-1}.$$

Using the SVD decomposition, we get:

$$X^+ = V \begin{pmatrix} d_1^{-1} & & \\ & \ddots & \\ & & d_p^{-1} \end{pmatrix} \bigcirc \; U^T$$

Low-Rank Approximations: Image compression. The SVD makes easy the construction of low-rank approximations of matrices and is therefore the basis of several data compression algorithms.

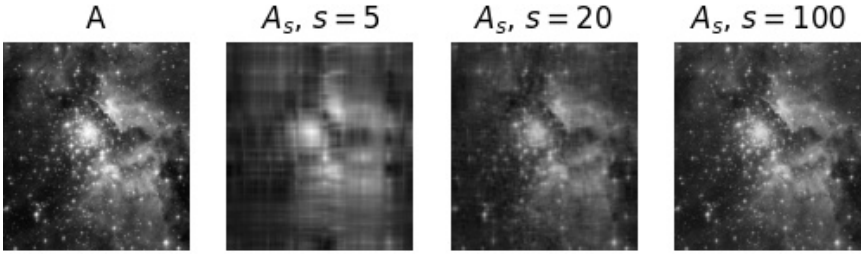
Let A be a $m \times n$ matrix of rank $r < min\{m, n\}$ and $A = U\Sigma V^T$. We can perform an image compression by storing matrices U_1, Σ_1 and V_1 instead of storing the matrix A. In the three matrices, we keep only the first k values and vectors.

Indeed, A contains mn values. But, storing the matrices U_1, Σ_1 and V_1 require $mr + r + nr$ values only. For instance, for a 100×200 matrix A of rank 20, storing A requires $20,000$ values while storing the matrices U_1, Σ_1 and V_1 requires $6,020$ entries only.

We can achieve better compression performance by using a *truncated SVD*, i.e., by keeping only the first $s < r$ singular values, plus the corresponding columns of U and V. The matrix A can be approximated by

$$A_s \approx \sum_{i=1}^{s} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{H}}.$$

The resulting matrix A_s has rank s and is only an approximation to A, since $r - s$ nonzero singular values are neglected.



3 Algebraic methods for image reconstruction

3.1 Degradation model

We would like to model some linear image degradation operators. We suppose that there exists a large $N \times N$ matrix $\mathbf{A}$ such that we can write

$$\boldsymbol{x} = \mathbf{A}\boldsymbol{y} + \boldsymbol{\epsilon}, \tag{I.3}$$

where $\boldsymbol{y}$ is the observed image, $\boldsymbol{x}$ is the unknown "ideal" image, and $\boldsymbol{\epsilon}$ is a noise (usually Gaussian). Note that we distinguish between the degradation $N \times N$ matrix $\mathbf{A}$ and the $m \times n$ image arrays $X, Y \in \mathbb{R}^{n \times m}$.

The model (I.3) can describe various types of degradation.

Noise

$\mathbf{A} = \mathbb{1}$.

Bluring

$\mathbf{A}$ = convolution with a Gaussian mask.

Inpainting

$\mathbf{A}$ = Multiplication with a binary matrix.

Estimation a PSF

$\mathbf{A}$ can be defined and estimated as the *point spread function (PSF)*. Each column of

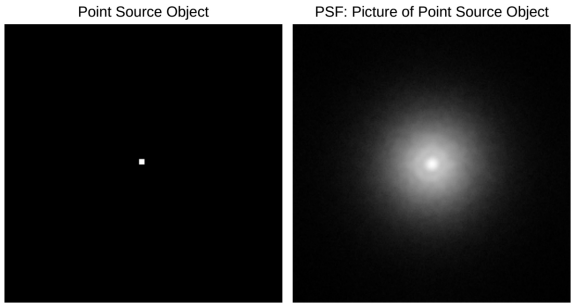


Fig. I.4

$\mathbf{A}$ can be written as $\mathbf{A}\mathbf{e}_j = \mathbf{a}_j$. One can construct $\mathbf{A}$ by generating all its columns, i.e., it can be estimated by imaging "point source" objects, for all different positions of the source point, corresponding to image pixels.

3.2 Algebraic solutions

A *naive approach* fo image reconstruction consists in inverting the matrix $\mathbf{A}$:

$$\hat{\mathbf{x}} = \mathbf{A}^{-1}\mathbf{y}.$$

This approach fails because of the noise. We explain here an *improved SVD-based method*.

We have

$$\begin{aligned} A &= U\Sigma V^T, \\ &= u_1\sigma_1 v_1^T + \cdots + u_N\sigma_N v_k^T, \\ &= \textstyle\sum_{i=1}^{k} \sigma_i u_i v_i^T. \end{aligned}$$

Similarly,

$$A^{-1} = V\Sigma^{-1}U^T = \sum_{i=1}^{k} \frac{1}{\sigma_i} v_i u_i^T$$

The error term of the naive solution becomes

$$A^{-1}\boldsymbol{\epsilon} = V\Sigma^{-1}U^T e = \sum_{i=1}^{k} \frac{u_i^T \boldsymbol{\epsilon}}{\sigma_i} v_i$$

For all i, the error components $|u_i^T e|$ are small and roughly of the same order of magnitude. Dividing by small singular values increases the corresponding error component. In addition, singular vectors corresponding to small singular values represent high frequency information. Thus, we discard the terms corresponding to small singular values to get better reconstruction results.

4 Optimization problems

Most image processing mathematical models considered here in this book are based on energy minimization. In such approaches, given an observed image $\boldsymbol{y}$, we "estimate" $\boldsymbol{x}$ using the following criterion:

$$\begin{aligned} \hat{\boldsymbol{x}} &= \operatorname*{argmin}_{\boldsymbol{x}} \; J(\boldsymbol{x}, \boldsymbol{y}), \\ &= \operatorname*{argmin}_{\boldsymbol{x}} \; h(\boldsymbol{x}, \boldsymbol{y}) + \tau\varphi(\boldsymbol{x}), \\ &= \operatorname*{argmin}_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \tau\varphi(\boldsymbol{x}). \end{aligned}$$

- The **data term** $h(\boldsymbol{x}, \boldsymbol{y})$ describes how well $\boldsymbol{x}$ "fits"/"explains" the data $\boldsymbol{y}$. A typical example is when there exists a theoretical functional relation between $\mathbf{x}$ and $\mathbf{y}$, of the form: $\boldsymbol{y} = F(\boldsymbol{x})$. In this case, one can choose $h(\boldsymbol{x}, \boldsymbol{y}) = \|F(\boldsymbol{x}) - \boldsymbol{y}\|$.
- The **regularization term** $\varphi(\mathbf{x})$ encodes some knowledge/constraints/structure. Common regularizers impose/encourage one, or a combination of the following characteristics: small norm (vector or matrix), sparsity (few nonzeros), low-rank (matrix), smoothness or piece-wise smoothness, ... Adding a regularization term

to the data term to from one single criterion is called *Tikhonov regularization*. We shall also point out other forms or regularization, "equivalent" under mild conditions, which are:

- Morozov regularization: $\underset{\mathbf{x}}{\operatorname{argmin}}\ \varphi(\boldsymbol{x}),\ s.t.,\ f(\boldsymbol{x}) \leq \varepsilon$
- Ivanov regularization: $\underset{\boldsymbol{x}}{\operatorname{argmin}}\ f(\boldsymbol{x}),\ s.t.,\ \varphi(\boldsymbol{x}) \leq \delta.$

Morozov and Ivanov can be written as Tikhonov using indicator functions.

- The **regularization weight** $\tau \geq 0$ controls the importance given to each term of the function J.

For instance, let $Y \in \mathbb{R}^{n \times m}$ be a given noisy image. A model for image denoising consists in solving the following optimization problem:

$$\underset{x}{\text{minimize}} \quad \|Y - X\|_{2,2}^2 + \lambda\varphi(X)$$

where $\varphi(X) = \sum_{i=1}^{n-1} \sum_{j=1}^{m-1} ((x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2)$. We should mention here that we used the matrix representation of images.