1 Inverse problems

1.1 INTRODUCTION

Mathematical modelling consists in describing real-life problems in terms of mathematical equations, typically ordinary differential equations (ODEs) or partial differential equations (PDEs). These equations usually involve parameters, initial and/or boundary conditions, in order to be mathematically or numerically solved. Solving these equations is called the direct - or forward - problem: given the input and system parameters, compute the output (solution) of the model. But very often, the actual problem consists in recovering the parameters and input of the model from the output. This is the inverse problem: using actual measurements of the system, recover the values of the parameters that characterize it (Tarantola, 2005; Beck and Arnold, 1977; Hensel, 1991; Kirsch, 1996; Vogel, 2002; Tikhonov and Arsenin, 1977).

Using generic notations, let $x \in X$ be the input of a model, X being the set of admissible inputs. Let $p \in P$ be the model parameters, and let $y \in Y$ be the output of the model. We assume the existence of a model function $M : (X, P) \to Y$, that gives the output

$$y = M(x, p) \tag{1.1}$$

from the input x and the parameters p.

The direct model consists in solving Equation (1.1):

$$\text{Given } x \text{ and } p, \text{ calculate } y = M(x, p).$$

The inverse problem consists in identifying the input x, and/or the model parameters p, from the (partial) knowledge of the output y:

$$\text{Given (partial observations of) } M(x, p), \text{ calculate } x \text{ and/or } p.$$

Inverse problems arise typically when the system is (at least partially) not known but it can be observed. And from these partial observations, we want to recover some of the system characteristics, which are often not directly observable.

1.1.1 EXAMPLES OF INVERSE PROBLEMS

The list of inverse problems applications has significantly grown these last decades, with the expansion of image processing, but also non invasive mapping, non destructive control, ..., in almost all application fields (Tarantola, 2005; Hensel, 1991; Beck and Arnold, 1977; Tikhonov and Arsenin, 1977). Here is a non exhaustive list of common inverse problems:

- medical imaging: X-ray tomography, ultrasound tomography, elastrography, ...

- image analysis: image deblurring, denoising, inpainting, restoration, ...
- geosciences: data assimilation for weather forecast, radio-astronomical imaging, seismic tomography, ...
- signal processing: deconvolution, gridding, ...
- mechanical engineering: crack detection, non destructive control, ...

And also, in either general cases or in the particular framework of machine learning, model fitting and parameter identification are typical inverse problems. We now focus on two particular examples.

1.1.2 LINEAR REGRESSION

Let assume that some theoretical model relates the output quantity $y \in Y = \mathbb{R}^p$ to the input quantity $x \in X = \mathbb{R}^n$ via a linear equation:

$$y = Ax, \tag{1.2}$$

where $A \in \mathscr{M}_{n,p}(\mathbb{R})$ is an $n \times p$ matrix. Multivariate linear regression is the simplest way to model the relationship between responses y and explanatory variables x. The matrix A contains the model parameters, which are here the linear regression coefficients.

The direct problem is quite simple in this case: given the explanatory variables x and the regression coefficients A, the output can easily be computed from the matrix-vector product in Equation (1.2). The inverse problem is less obvious, and probably more interesting from the mathematical point of view: given a set of measurements - usually observations of the couple (x,y) - determine the regression coefficients A such that Equation (1.2) is satisfied, at least in the least square sense (Aster et al., 2008; Tarantola, 2005; Hensel, 1991).

Model fitting can be seen as a particular case of parameter estimation problems, as the goal is indeed to estimate the parameters of the regression model. Parameter estimation problems also occur in more complex machine learning algorithms (e.g. neural networks), but also in ODE or PDE models, for instance when designing digital twins (Tarantola, 2005; Chalmond, 2003; Bertero et al., 2021; Beck and Arnold, 1977).

1.1.3 INVERSE HEAT CONDUCTION

In a physical (e.g. thermal, electrostatical or acoustical) framework, non-destructive testing of materials usually consists in detecting interior cracks in a given body, without degrading it. This can be done by applying a known thermal source to the boundary of the object, and by measuring the thermal flux, still on the boundary of the object. Cracks being partially (or almost totally) insulating, they will modify the temperature map inside the body, and then also the heat flux on the boundary.

Let Ω be a bounded open set of $\mathbb{R}^3$ that represents the object, Γ its boundary, and $\sigma \in \Omega$ some perfectly insulating crack. The direct model is then the following heat

equation:

$$\begin{cases} \Delta u = 0 & \text{in } \Omega \setminus \sigma, \\ u = T & \text{on } \Gamma, \\ \partial_n u = 0 & \text{on } \sigma, \end{cases} \tag{1.3}$$

where $T \in H^{\frac{1}{2}}(\Gamma)$ represents the heat source applied to the boundary of the object. Equation (1.3) can be solved in order to get the heat map $u \in H^1(\Omega \setminus \sigma)$. And as consequence, we can compute the output flux $\varphi = \partial_n u$ at the boundary Γ (n being the normal to the boundary) (Beck et al., 1985; Kohn and Vogelius, 1984).

There are many associated inverse problems, one of them is the following: knowing overdetermined boundary values (temperature T and heat flux φ at the boundary Γ), identify the cracks $\sigma \in \Omega$. This inverse problem typicaly arises in mechanical engineering for cracks identification in non destructive control, but also has applications to image processing, the Laplacian operator acting as a blurring (and then denoising for Gaussian white noise) operator (Friedman and Vogelius, 1989; Kohn and Vogelius, 1984; Beck et al., 1985; Kirsch, 1996; Auroux and Masmoudi, 2009).

1.2 WELL-POSED AND ILL-POSED INVERSE PROBLEMS

Hadamard introduced in 1902 the notion of well-posedness (Hadamard, 1902). According to his definition, an inverse problem is well-posed if the three following properties hold:

- a solution exists,
- the solution is unique,
- the solution depends continuously on the data.

Otherwise, the inverse problem is said to be ill-posed.

Existence is of course a prerequisite to solving the inverse problem. But uniqueness also: non uniqueness can lead to situations where even with perfect data, it may not be possible to recover the exact quantities to be identified. The last condition, stability, is also necessary in the following sense: it ensures that a small change (typically errors) in data leads to only a small modification of the reconstructed solution. Unfortunately, because direct problems are usually stable, inverse problems often involve irreversibility or causality issues, leading to ill-posedness.

Let us see some examples of ill-posed inverse problems.

1.2.1 PARAMETER IDENTIFICATION

The identification of parameters in differential equations is often an ill-posed inverse problem. We consider here the case of a simple one dimensional stationary heat equation, where the diffusion coefficient a is unknown:

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right) = f(x), \quad x \in (0,1), \tag{1.4}$$

with boundary conditions $u(0) = u_0$ and $u(1) = u_1$. In Equation (1.4), u denotes the temperature, a the thermal conductivity of the material, and f the heat source. Solving the heat equation consists in finding the temperature u from the knowledge of the conductivity a and the source term f. This is a very well known and studied inverse problem (Friedman and Vogelius, 1989; Kohn and Vogelius, 1984; Beck et al., 1985). Sometimes, the inverse problem consists in identifying the parameter a from the knowledge of the source f and (at least partially) of the temperature u. In dimension 1, it is easy to integrate Equation (1.4) in order to determine the coefficient:

$$-a(x)\frac{du}{dx}(x) + a(0)\frac{du}{dx}(0) = \int_0^x f(s)\,ds,$$

which leads to

$$a(x) = \frac{-\int_0^x f(s)\,ds + a(0)\frac{du}{dx}(0)}{\frac{du}{dx}(x)}. \tag{1.5}$$

Assuming we can measure the conductivity on the boundary $a(0)$, and the temperature everywhere, Equation (1.5) allows us to recover the conductivity in the whole material. But as you can see, in particular situations, it will be an ill-posed problem. For instance, if $\frac{du}{dx}(x) = 0$ for some x, then there is no solution: it is impossible to recover the conductivity.

1.2.2 INTEGRAL EQUATIONS

We consider here Fredholm integral equations of the first kind associated to a kernel $K(x,y) \in \mathscr{L}^2((0,1),(0,1))$ (set of functions, the square of which are integrable), and to data $f \in \mathscr{L}^2(0,1)$:

$$\text{Find } \varphi \in \mathscr{L}^2(0,1) \text{ such that } \int_0^1 K(x,y)\varphi(y)\,dy = f(x), \quad \forall x \in (0,1). \tag{1.6}$$

Let assume that the kernel K is continuously differentiable ($\mathscr{C}^1$). Then $\forall \varphi \in \mathscr{L}^2(0,1)$, from Equation (1.6), f will also be continously differentiable. The inverse problem of recovering φ from f might then be ill-posed: if the input data f is not continuously differentiable, then Equation (1.6) has no solution.

Note also that ill-posedness can be a consequence of the violation of more than one Hadamard conditions. Typically here, when the solution of (1.6) exists, it may also not depend continuously on the data (Groetsch, 1984, 2007; Bôcher, 1909; Tikhonov and Arsenin, 1977).

1.2.3 DIFFERENTIATION

Differentiation can be seen as an inverse problem corresponding to the direct problem of integration. We consider here a simple example in dimension 1: let $f \in \mathscr{C}^1(0,1)$

a continuous and differentiable function on the interval $(0,1) \subset \mathbb{R}$, the derivative of which being also continuous.

Let now consider the following perturbed function

$$f_{\delta,n}(x) = f(x) + \delta \sin\left(\frac{nx}{\delta}\right), \tag{1.7}$$

where $x \in (0,1)$, $\delta \in (0,1)$ and $n \in \mathbb{N}^*$ is a positive integer.

Then it is obvious that $f_{\delta,n}$ also belongs to $\mathscr{C}^1(0,1)$ and that

$$f'_{\delta,n}(x) = f'(x) + n\cos\left(\frac{nx}{\delta}\right).$$

If we look at the $\mathscr{L}^\infty$ norm, defined for continuous functions as:

$$\|f\|_\infty = \sup_{x\in(0,1)} |f(x)|,$$

then we can easily see that

$$\|f - f_{\delta,n}\|_\infty = \sup_{(0,1)} \left|\delta \sin\left(\frac{nx}{\delta}\right)\right| = \delta, \qquad \text{and} \qquad \|f' - f'_{\delta,n}\|_\infty = n.$$

As we can take δ arbitrarily small and n arbitrarily large, it shows that even infinitely close functions (input data) f and $f_{\delta,n}$ can lead to arbitrarily different derivatives (solutions of the inverse problem) f' and $f'_{\delta,n}$. These functions and their derivatives are plot on Figure 1.1: on the left plot, there is no visual difference between the two functions; on the right plot, the derivatives are obviously very different. Therefore, the solution does not depend continuously on the data: from the third condition of Hamadard, differentiation is an ill-posed problem. This can also be understood from the fact that integration is a smooth (well-posed) process, where the regularity of functions increases, and thus leading to a loss of regularity in the inverse process (differentiation) (Lavrent'ev and Savel'ev, 2006; Engl and Groetsch, 1987; Engl, 1987).

1.2.4 OTHER EXAMPLES

It is often not too difficult to handle existence and uniqueness issues, but regularity is the most difficult point to deal with, making inverse problems ill-posed and hard to solve.

Image deblurring is a standard (ill-posed) inverse problem in image processing. The presence of noise in the blurred image can lead to dramatic errors in the reconstructed image, as shown in Figure 1.2. Blurring (direct problem) can be easily modeled by e.g. a convolution with a Gaussian kernal, so that deblurring (inverse problem) consists in a deconvolution, which is usually ill-posed (similarly to solving a heat equation backwards in time) (Hansen et al., 2006; Scherzer, 2011). Figure 1.2 shows an original image (standard Shepp-Logan phantom), that we blurred thanks to a convolution with a small Gaussian kernel. Then, deblurring this blurred image is

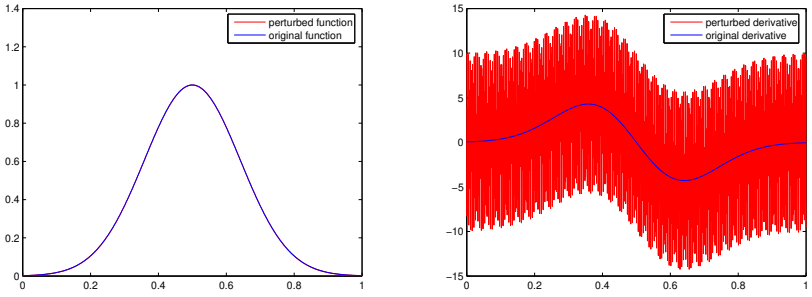

Figure 1.1: f and $f_{\delta,n}$ (left); f' and $f'_{\delta,n}$ (right); for $\delta = 0.001$ and $n = 10$.

quite efficient (as we know the kernel that was used for blurring). But then, adding some noise to the blurred image drastically changes the problem, and deblurring leads to a totally different image, where the original signal is almost totally lost.

X-ray tomography consists in reconstructing the image inside a body from a set of projections called sinograms, which are X-ray line integrals along some directions. Tomography is based on the Radon transform:

$$R(f)(\theta,r) = \int_{\Omega} f(x,y)\,\delta(r - x\cos(\theta) - y\sin(\theta))\,dx\,dy,$$

where f is the body image, Ω is the domain (typically open bounded and convex subset of $\mathbb{R}^2$), θ and r are the polar coordinates of the X-ray direction, and δ is the Dirac distribution (Scherzer, 2011; Natterer, 2001).

Computing the Radon transform $R(f)$ from a source f is the direct problem. And computing the source image f from its Radon transform $R(f)$ is the inverse problem. Even if the Radon transform can be analytically inversed, thanks to the Fourier transform, it is highly unstable and the inverse problem is ill-posed, due to lack of continuity of the solution with respect to the input sinograms (Scherzer, 2011; Natterer, 2001; Bertero et al., 2021; Ólafsson and Quinto, 2006).

As an example, Figure 1.3 shows an example of direct Radon transform (computation of the sinogram from an image) and inverse Radon transform (reconstruction of the image from the sinogram) in the case of a perfect image, and then with some noise added to the sinogram. As one can see, the reconstructed image from the noisy sinogram does not contain any valuable information, while the noisy sinogram looks quite close to the original sinogram.

1.2.5 COMPACTNESS AND ILL-POSEDNESS

The characteristics of the inverse problems are linked to the properties of the model operator M in Equation (1.1). One of these properties is the compactness. A linear operator M is said to be compact if it maps bounded subsets of (X,P) to subsets with compact closure in Y. This means that if we consider bounded sequences of

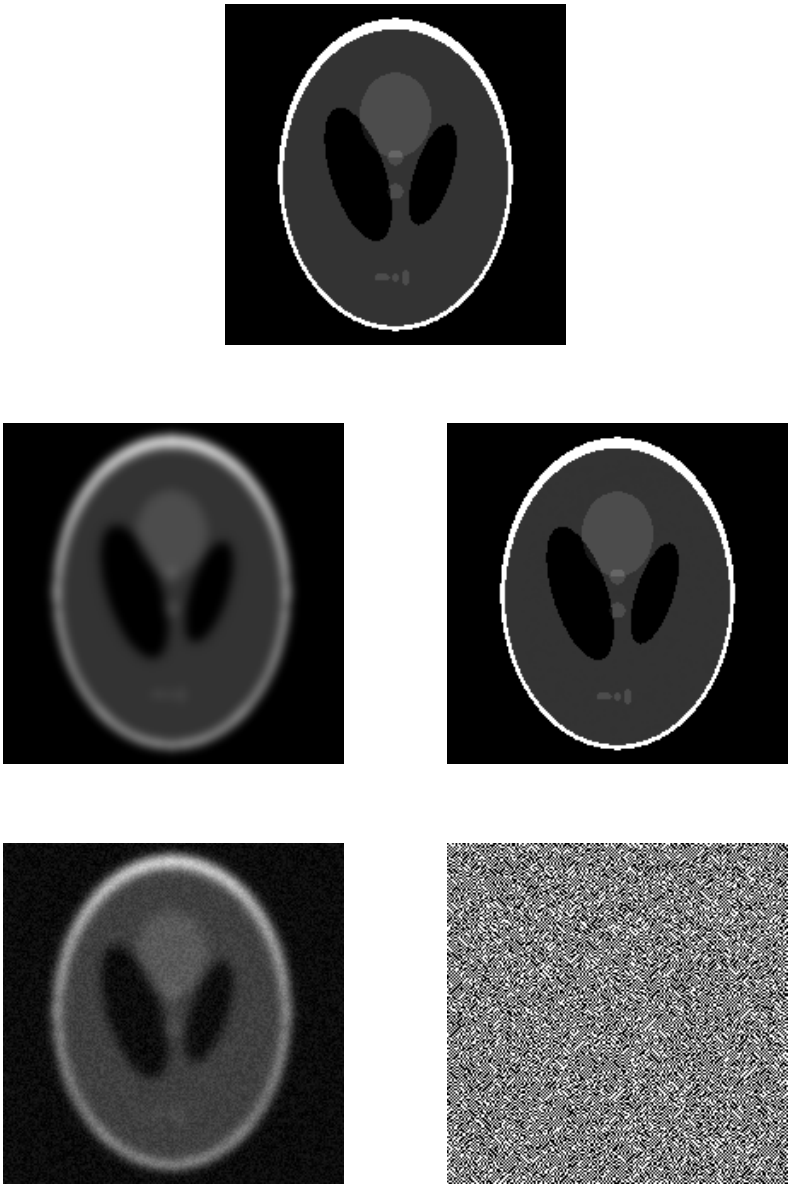

Figure 1.2: Top: Original image; Middle: Blurred image (with a Gaussian kernel convolution), deblurred image; Bottom: Blurred and noisy image (with a SNR of approximately 10 dB), reconstructed image from the noisy blurred image.

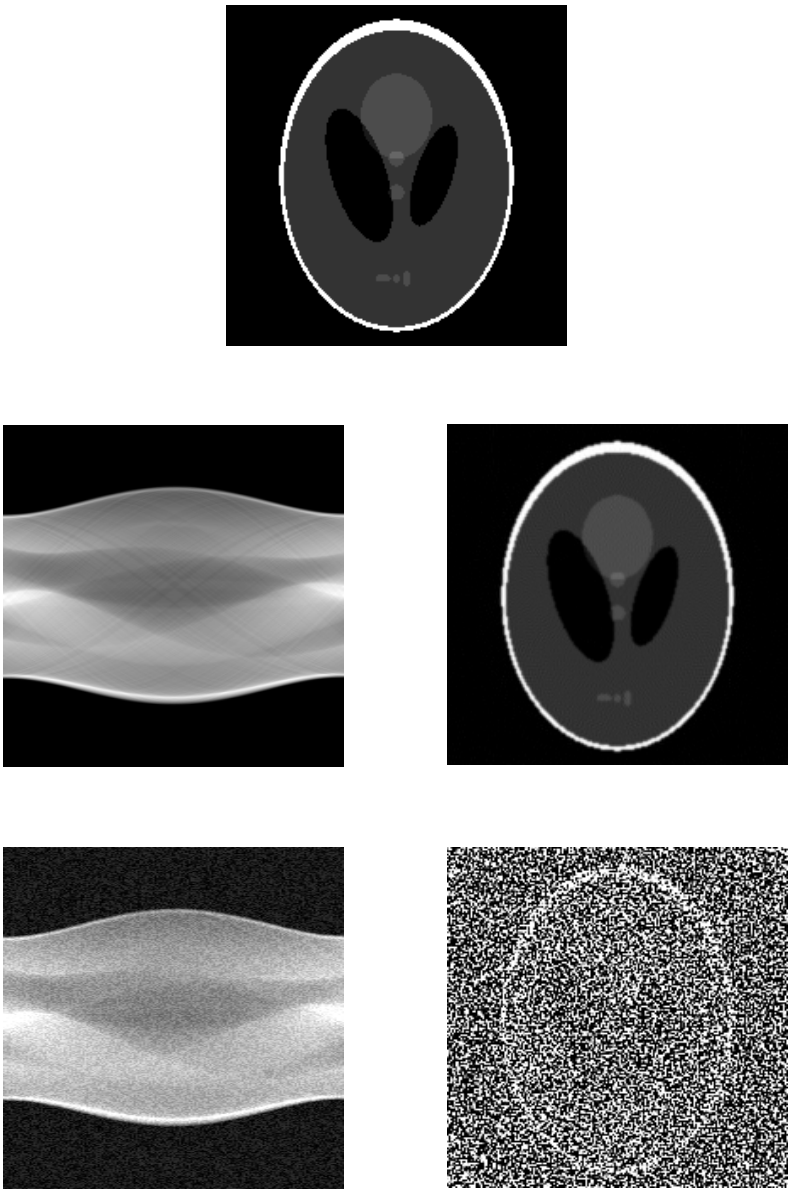

Figure 1.3: Top: Original image; Middle: Corresponding sinogram (using a Radon transform over 180 degrees), reconstructed image from the sinogram (using the inverse Radon transform); Bottom: Noisy sinogram (with a white Gaussian noise, SNR of approximately 14 dB), reconstructed image from the noisy sinogram.

input data, the output sequence contains a converging subsequence (Tarantola, 2005; Hofmann and Kindermann, 2010).

From Bolzano-Weierstrass theorem, in finite dimension, or more generally if the linear model operator has a finite dimensional range, then it is compact. This means that any discretized operator, in order to be numerically solved, is compact. As a consequence, the inverse operator (if it exists) may amplify errors.

In infinite dimension, if the operator is compact and injective, then its inverse (or generalized inverse) is not continuous. So that the inverse problem is ill-posed as nothing can guarantee that a small perturbation on the data will lead to a small perturbation on the reconstructed solution (Tarantola, 2005; Schwartz, 1981; Tikhonov and Arsenin, 1977).

In order to numerically solve the problem, one usually needs to discretize the model operator, leading to a finite dimension operator, which is then compact. As the limit of compact operators is compact in the norm topology, the infinite dimension operator is then compact, leading to an ill-posed inverse problem: any arbitrarily small error in the data can drastically corrupt the reconstruction. Of course, the discretized operator, in finite dimension, will not be ill-posed, but as we solve the inverse of a compact operator, it might amplify the small errors in the data. And the closer to the infinite operator we are, the higher the amplification will be (and at the limit, the reconstruction will be ill-posed) (Tikhonov and Arsenin, 1977; Tarantola, 2005).

Note that function integration, image blurring, the Radon transform, ..., are linear and compact transformations, so that the corresponding inverse problems are indeed ill-posed.

1.3 REGULARIZATION

In order to be able to solve ill-posed inverse problems, regularization is often required. There are many ways to regularize the problem in order to make it well-posed.

When the inverse of the model is not continuous, the reconstruction from noisy data might not be close to the solution (i.e. the reconstruction from perfect data), even for an arbitrarily small noise. The common idea of regularization methods is the following: approximate the inverse model M^{-1} by a series of approximations R_ε, that converge to the exact inverse model when the regularization parameter ε goes to 0, and such that the inversion of the approximation is well-posed, at least for well chosen regularization parameters.

1.3.1 TIKHONOV REGULARIZATION

For sake of simplicity, we consider here a linear model in finite dimension:

$$y = Mx, \tag{1.8}$$

where $x \in \mathbb{R}^n$ represents the input, $y \in \mathbb{R}^p$ the output, and $M : X \to Y$ the linear model operator (i.e. a matrix). The inverse problem consists in computing x from y, solution of Equation (1.8).

If Equation (1.8) is ill-posed, the idea is first to compute the normal equation by multiplying (1.8) by the adjoint operator M^*:

$$M^*y = M^*Mx. \tag{1.9}$$

Tikhonov regularization consists in slightly perturbing the operator M^*M in the following sense:

$$M^*y = (M^*M + \varepsilon I)x, \tag{1.10}$$

where I is the identity matrix, leading to the regularized inverse model

$$R_\varepsilon = (M^*M + \varepsilon I)^{-1}M^*.$$

The solution of the well-posed regularized problem is then $x_\varepsilon = R_\varepsilon y$ (Tikhonov and Arsenin, 1977; Tarantola, 2005; Hensel, 1991).

Note that finding the solution of Equation (1.9) is equivalent to minimizing the following cost function:

$$J(x) = \|y - Mx\|^2,$$

that measures the residuals in square norm, using the standard associated $\mathscr{L}^2$ norm. Tikhonov regularized model (1.10) is then equivalent to minimizing the following regularized cost function:

$$J_\varepsilon(x) = \|y - Mx\|^2 + \varepsilon\|x\|^2. \tag{1.11}$$

By forcing the solution x to remain bounded (close to 0 here), the regularization term ensures positive definiteness of the quadratic form, leading to existence, uniqueness and continuity of the solution (i.e. well-posed problem). Note that the choice of the norm can be adapted, particularly for the regularization term: $\mathscr{L}^1$ norm, total variation (for image analysis), ... Note also that the regularization term can be used to force the solution to be close to some a priori estimation of the solution: $\varepsilon\|x - x_{background}\|^2$ (Kirsch, 1996; Chalmond, 2003; Bertero et al., 2021; Aubert and Kornprobst, 2006; Scherzer, 2011; Engl, 1987).

Let first see that we now have a well-posed problem for any $\varepsilon > 0$. If we take the inner product of Equation (1.10) with x, we obtain:

$$\|Mx\|^2 + \varepsilon\|x\|^2 = \langle M^*y, x\rangle = \langle y, Mx\rangle \leq \|y\|\,\|Mx\|.$$

We deduce that $\|Mx\|^2 \leq \|y\|\,\|Mx\|$ and thus $\|Mx\| \leq \|y\|$. Moreover, $\varepsilon\|x\|^2 \leq \|y\|\,\|Mx\| \leq \|y\|^2$. As $x = R_\varepsilon y$, we deduce that $\varepsilon\|R_\varepsilon y\|^2 \leq \|y\|^2$, and thus

$$\|R_\varepsilon\| \leq \frac{1}{\sqrt{\varepsilon}}.$$

We easily deduce that R_ε is continuous, so that $x = R_\varepsilon y$ continuously depends on y.

We can moreover prove that the solution corresponding to noisy data converges to the solution corresponding to perfect data when noise goes to 0: let y_δ be an approximation of y such that $\|y_\delta - y\| \leq \delta$. As

$$\|R_\varepsilon y_\delta - M^{-1}y\| \leq \|R_\varepsilon(y_\delta - y)\| + \|R_\varepsilon y - M^{-1}y\| \leq \frac{\delta}{\sqrt{\varepsilon}} + \|R_\varepsilon y - M^{-1}y\|,$$

and as by definition of the regularization, $\|R_\varepsilon y - M^{-1}y\| \to 0$ when $\varepsilon \to 0$, it is easy to choose ε, e.g. $\varepsilon = \delta$, such that $\|R_\varepsilon y_\delta - M^{-1}y\| \to 0$ when $\delta \to 0$. Using an appropriate regularization coefficient, this proves that the reconstructed regularized solution tends to the true solution when noise goes to 0, while we know that the reconstructed unregularized solution may not converge.

As an example, we consider a simple case where M is a Hilbert matrix in dimension 10, known to be ill-conditioned (Bakushinsky and Goncharsky, 1994; Burden et al., 2015). We define a test vector x, we compute $y = Mx$, and we then forget x, the goal being to recover x from y (which is our typical inverse problem). When trying to invert the linear system without regularization, the solution obtained has a relative error (in $\mathscr{L}^2$ norm) of 17%, due to ill-conditioning: some of the vector components are well identified, some are totally wrong. Note that the solver returns a warning about the bad conditioning of the matrix (approximately 10^{-19}) and possibly inaccurate results.

We now add a Tikhonov regularization, and solve the regularized system for various values of ε. The relative norm of the error, between the identified solution and the true solution (the vector x we chose at the beginning) is plotted versus ε, in logarithmic scale, on Figure 1.4. For large values of ε, the problem is too far away from the original inverse problem, so that the reconstructed solution is not very good. When ε gets smaller, the regularized inverse problem gets closer to the original inverse problem, so that the identified solution becomes more accurate. Note that there are no warnings about bad conditioning, and no issues in computing the solution. Finally, for very small values of ε, approximately less than 10^{-15}, the conditioning of the regularized system becomes poor again, leading to inaccuracies and the error increases again. For ε smaller than 10^{-18}, the identified solution of the regularized system is exactly the same as the unregularized one.

This illustrates the main advantages of Tikhonov regularization: instead of solving an ill-conditioned problem, for which the solution might be partially or totally wrong, it is usually better to solve a slightly different problem, the regularized one, which is well-posed and for which the solution can be easily (and accurately) computed. This of course leads to a slightly approximated solution (as the problem is not exactly the original one), but still much better than what can be expected from the ill-posed original problem.

1.3.2 CHOICE OF THE REGULARIZATION PARAMETER

As shown on Figure 1.4, the standard question that naturally arises with regularization is: how to choose the regularization parameter? A small parameter will of course help to solve a problem which is close to the original system, but it might still be ill-posed. A large parameter will ensure the well-posedness, but the regularized model might be far away from the original system, so that the regularized solution can be useless (too different from the desired solution of the original unregularized system).

As this parameter acts as a weight between the two terms - data fitting and regularization - of Equation (1.11), a standard method to tune it is called the L-curve. It consists in plotting in log-log scale the residual norm $\|y - Mx\|^2$ versus the rough-

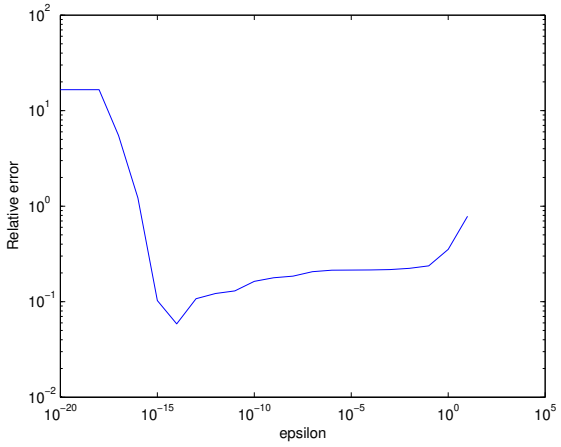

Figure 1.4: Relative norm of the error on the identified solution of (1.10) for various values of the regularization parameter ε.

ness $\|x\|^2$ at the optimum (identified minimum of the cost function), for a wide range of ε values. Such curve will usually have a characteristic L-shape, and the idea is to choose the regularization parameter corresponding to the corner of the L (Hansen and O'Leary, 1993; Hansen et al., 2006).

For large values of the regularization parameter (high filtering), the function to be minimized will be driven by the roughness term, so that the data misfit term will vary a lot while the roughness will not change that much (the norm of the solution being already small). On the contrary, for small values of the regularization parameter (low filtering), the cost function is driven by the data misfit term, the problem becomes ill-posed, and the solution may have its norm increase a lot in order to try to make the data misfit decrease.

Figure 1.5 shows the square norm of the residual error versus the square norm of the solution for various values of the regularization parameter, for the same example as in the previous section. The optimal parameter would be around 10^{-15} (red diamond), for which the compromise seems the best. As seen in the previous section, the problem starts to become again ill-posed for ε smaller than 10^{-15}, so that choosing ε close to 10^{-15} is in this case the best choice for keeping the problem well-posed, while being the closest possible to the original unregularized problem.

Choosing the parameter near the corner implies that the trade-off between filtering and data fitting is optimal: less filtering will degrade sharply the solution norm, while more filtering will degrade the data fitting term.

1.3.3 NUMERICAL DISCRETIZATION

Numerical discretization is another way to regularize and make the inverse problem well-posed. As explained in Section 1.2.5, discretizing a continuous compact oper-

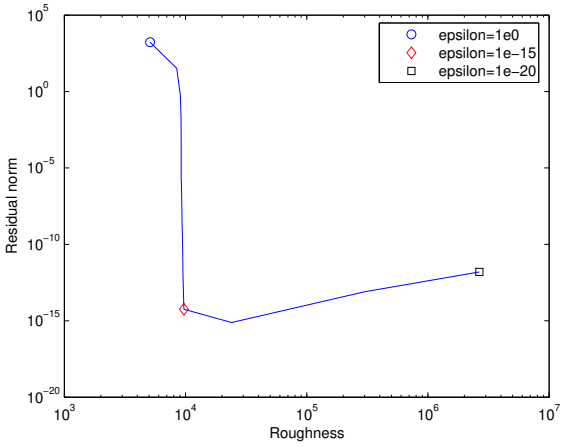

Figure 1.5: Square norm of the residual error $\|y - Mx\|^2$ versus square norm of the solution $\|x\|^2$ in log scale, for various values of the regularization parameter ε.

ator leads to a compact operator, but it is not ill-posed in finite dimension. So that discretization is a form of regularization. As previously said, the finer the discretization, the more ill-posed the discretized problem.

As a simple example, let us consider again differentiation of functions on the interval $(0,1)$. As we have seen in Section 1.2.3, differentiation does not depend continuously on the input data. Let now consider discrete, or numerical, differentiation:

$$f'^{\varepsilon}(x) := \frac{f(x+\varepsilon) - f(x)}{\varepsilon},$$

where ε is chosen small and such that $x+\varepsilon$ still belongs to $(0,1)$. The discrete derivative f'^{ε} is simply defined as the rate of change of the function between x and $x+\varepsilon$.

If f is differentiable, by definition of the derivative, the discrete derivative tends to the actual derivative when the discretization parameter ε tends to 0. Let now see what happens if we slightly perturb the input function: let f_δ such that $\|f_\delta - f\|_\infty \leq \delta$. As an example, note that $f_{\delta,n}$ defined by Equation (1.7) satisfies this inequality.

We have:

$$|f_\delta'^{\varepsilon}(x) - f'(x)| = \left| \frac{f_\delta(x+\varepsilon) - f_\delta(x)}{\varepsilon} - f'(x) \right|$$

$$\leq \left| \frac{f(x+\varepsilon) - f(x)}{\varepsilon} - f'(x) \right| + \left| \frac{(f_\delta - f)(x+\varepsilon) - (f_\delta - f)(x)}{\varepsilon} \right|.$$

Under the hypothesis that f' is continuously differentiable, i.e. $f \in \mathscr{C}^2$, then the first term $|\frac{f(x+\varepsilon)-f(x)}{\varepsilon} - f'(x)|$ is bounded by $C\varepsilon$, for some C (namely half of the infinity norm of f'', from Taylor series expansion). The second term is bounded by

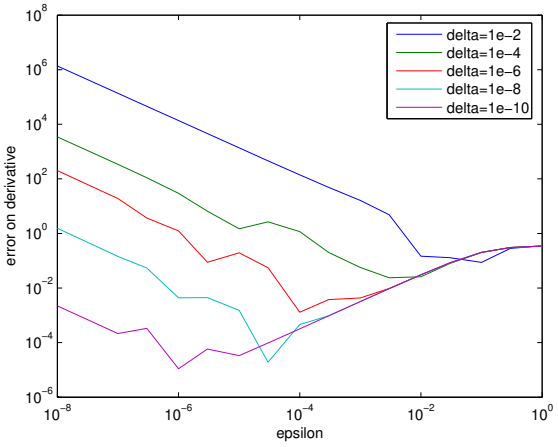

Figure 1.6: Difference between $f'(x)$ and the numerical derivative of f_δ as a function of ε (parameter of the numerical differentiation) in logarithmic scale, for various values of δ.

$2\frac{\delta}{\varepsilon}$ as $\|f_\delta - f\|_\infty \leq \delta$. Hence,

$$|f'^{\varepsilon}_{\delta}(x) - f'(x)| \leq C\varepsilon + 2\frac{\delta}{\varepsilon}. \tag{1.12}$$

As we can see, when ε goes to 0, the right hand side goes to infinity: we are back to the continuous differentiation, which can be ill-posed. But the idea is to choose an appropriate parameter such that we are close to the original problem and well-posed. In our case, ε should be chosen such that the right hand side bound still tends to 0 when $\delta \to 0$. Typically, we can choose ε of the order of $\sqrt{\delta}$, so that

$$|f'^{\varepsilon}_{\delta}(x) - f'(x)| \leq C\sqrt{\delta} + 2\sqrt{\delta}, \tag{1.13}$$

meaning that the discrete differentiation is now well-posed: it depends continuously on the input data.

Figure 1.6 shows $|f'^{\varepsilon}_{\delta}(x) - f'(x)|$ versus ε for various values of δ. In all cases, the shape of the curve is the same: when ε decreases, the numerical derivative (given by the rate of change of the function) gets closer to the true derivative, but as the theoretical bound (see Equation (1.12)) has a term in $\frac{1}{\varepsilon}$, when ε keeps decreasing, the numerical derivative becomes poorer. As seen in Equations (1.12) and (1.13), choosing ε small enough, but such that $\frac{\delta}{\varepsilon}$ is also small, is necessary. As shown on Figure 1.6, the optimal value for ε is close to the square root of δ, ensuring a good error bound as in Equation (1.13).

Note that discretization makes operators bounded, and thus continuous (and their inverse also). But as the discretization parameter goes to 0, the discretized operator

converges to the continuous operator, and its spectrum tends to be unbounded, leading to ill-posedness.

This latter remark is valid for any regularization technique: the closer to the original problem, the more ill-posed. The choice of the regularization or discretization parameter is thus crucial in this process: not too large, so that the approximated/regularized problem is not too far away from the original problem (poor approximation); not too small, so that the problem does not become ill-posed again (noise amplification).

1.4 OPTIMIZATION

Solving the inverse problem is usually done by minimizing a cost function that measures the misfit between the solution and the data (residual term), usually incremented by a regularization term (see previous section). Using similar notations as in Section 1.3.1, the goal is to minimize

$$J_\varepsilon(x) = \|y - Mx\|^2 + \varepsilon\|x\|^2, \tag{1.14}$$

where y is the input data, M represents the model, and ε is the regularization parameter. For sake of simplicity, we will assume the model M to be linear, and also that the involved norms are the standard $\mathscr{L}^2$ norms, but it is not mandatory.

1.4.1 MINIMIZATION OF THE COST FUNCTION

We first assume that there are no constraints on x, and also that the cost function is differentiable. This latter point is usually satisfied, for instance in the machine learning framework, and more generally guaranteed by the use of squares of standard $\mathscr{L}^2$ (or equivalent) norms, at least for the residual. We will assume the differentiability of the regularization term, which is less obvious for instance in image processing where total variation norms might be used (Scherzer, 2011; Aubert and Kornprobst, 2006).

Standard minimization approaches are iterative and gradient-based, in the sense that they build a sequence of approximations of the minimum, and at each iteration, the gradient of the cost function is used to update the approximation. As examples of such minimization algorithms, we can cite the optimal-step gradient algorithm, the conjugate gradient, quasi-Newton algorithms, Levenberg-Marquardt algorithm, ... (Pytlak, 2010; Allaire, 2007).

In the simplest gradient algorithm methods, the sequence of approximations is built in the following iterative way:

$$x_{k+1} = x_k - \rho_k \nabla J_\varepsilon(x_k),$$

starting from an initial guess x_0, and where ρ_k is a step, usually found by minimizing the cost function along the gradient direction: it is a one dimensional minimization subproblem, called line-search (Allaire, 2007; Babaeizadeh and Ahmad, 2012; Bierlaire, 2015).

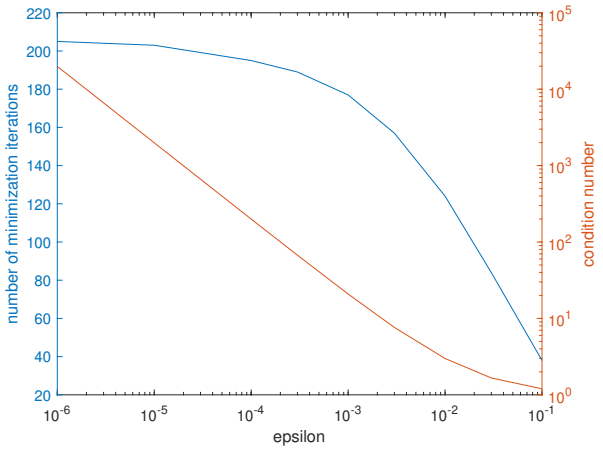

Figure 1.7: Number of optimization iterations needed to achieve convergence towards the minimum (left) and condition number of the system (right) versus ε (regularization parameter).

Using all previous gradients and not only the last one ($\nabla J_\varepsilon(x_k)$) for updating the solution leads to more complex algorithms such as the conjugate gradient, or quasi-Newton approaches in which an approximation of the Hessian matrix is built from the family of previously computed gradient vectors (Allaire, 2007; chong and Zak, 2013).

At the optimum, as there are no constraints, the gradient of the cost is equal to 0, so the gradient algorithms usually include a test on small values of the gradient used as a stopping criterion.

Figure 1.7 shows on the left y-axis the number of iterations required by the optimization algorithm in order to achieve convergence towards the minimum of J_ε, as a function of the regularization parameter ε. As one can see, the smaller ε, the larger the number of iterations. This is due to the condition number of the system, which is shown on the right y-axis, in logarithmic scale. The condition number increases more or less like $\frac{1}{\varepsilon}$ (line of slope -1 in log-log scale), so that it becomes more and more difficult to solve the system when ε gets smaller, the condition number becoming larger (Hilbert, 1894; Choi, 1983). Regularizing the cost function to be minimized is then often necessary in order to have a well-conditioned system, and thus a well-posed problem.

Note that finding the zero of the gradient (which will correspond to a minimum of the cost function) is equivalent to solving the normal equation, see Equation (1.10). And in a purely linear situation, the minimum could then be found by solving this linear system:

$$(M^*M + \varepsilon I)x = M^*y.$$

The point is that if the regularization parameter is too small (what should be

the case as much as possible in order to solve the original problem), the condition number of the system matrix $(M^*M + \varepsilon I)$ will be close to the square of the condition number of the original inverse problem M, which is already large (because ill-posed). So that a direct solver might fail inverting this system. It is then highly recommended to use either a QR factorization for preconditioning, or a singular-value decomposition (SVD) for truncation, in order to drastically reduce the condition number of the matrix (Allaire and Kaber, 2008; Burden et al., 2015).

In more realistic (and then complex) situations, either the high dimension or non-linearities (or both) can prevent direct solvers from being efficient. And actually minimizing the cost function by evaluating its gradient becomes necessary (Allaire, 2007; chong and Zak, 2013).

1.4.2 EXAMPLE: LINEAR REGRESSION

We consider here as an example one of the simplest and most well known algorithms in machine learning: linear regression. The aim of linear regression is to model the relationship between explanatory (input) variables and a dependent (output) variable as a linear equation (Montgomery et al., 2012; Olive, 2017).

Let X_1, X_2, $\dots X_p$ be the p explanatory (input) variables, and Y the dependent (output) variable. From a stochastic point of view, these variables are random variables, that are observed multiple times, leading us to consider these variables as vectors of $\mathbb{R}^n$, n being the number of observations of each random variable. The linear regression model assumes the following linear dependence between variables:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots \beta_p X_p + \varepsilon,$$

where β_0 is the intercept, $\beta_1 \dots \beta_p$ are the slope coefficients for each explanatory variable, and ε represents the model error (or residuals). The β coefficients are all scalars. The inverse problem consists in finding the optimal coefficients β from the knowledge of $Y, X_1 \dots X_p$, optimal in the sense that the model error ε is the smallest possible.

As the model error is unknown, the standard way to identify the slope coefficients and intercept consists in minimizing the residuals:

$$J(\beta_0, \beta_1 \dots \beta_p) = \| Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots \beta_p X_p) \|^2.$$

We artificially introduce a dummy vector $X_0 \in \mathbb{R}^n$ full of ones, so that we can replace β_0 by $\beta_0 X_0$ in the previous equations. Let X be the $n \times (p+1)$ matrix with $X_0 \dots X_p$ as the $p+1$ columns. Let $\beta \in \mathbb{R}^{p+1}$ be the vector of unknowns, with $\beta_0 \dots \beta_p$ as the components. Then, we can rewrite the cost function in the following way:

$$J(\beta) = \| Y - X\beta \|^2.$$

This cost function being quadratic (as the model is linear with respect to the coefficients β), the conjugate gradient algorithm is a good choice for minimizing J.

Figure 1.8 shows the evolution of the parameters β during the optimization process (here a conjugate gradient). For this example, $p = 10$, and we used synthetic

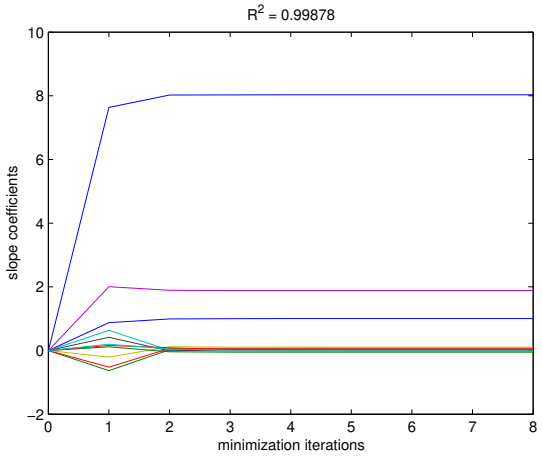

Figure 1.8: Evolution of the identified coefficients β during the optimization process (iterations).

data where $X_1 \dots X_{10}$ are generated from random white Gaussian processes, and Y is defined as an actual linear combination of the X variables, with some additional (white Gaussian) noise. The true parameters used for this linear combination are: all $\beta = 0$ except the intercept $\beta_0 = 1$, $\beta_4 = 2$ and $\beta_7 = 8$. As the cost function is quadratic, the conjugate gradient converges in at most 11 iterations (the number of unknowns). Here, after 8 iterations, the residual error is smaller than the standard threshold, and the parameters converged to almost the exact values.

The R^2 coefficient is almost equal to 99.9%, which means that nearly all the information contained in Y has been explained by the explanatory variables. This coefficient, known as the coefficient of determination, is the square of the correlation coefficient, and is defined as

$$R^2 = 1 - \frac{\text{sum of the squares of the residuals}}{\text{total sum of squares}}.$$

It quantifies the efficiency of the regression: if the residuals are small (compared to the original variance of Y), then R^2 is close to 1 (or 100%). On the contrary, when the residuals remain large (relatively to the original variance), R^2 decreases to a lower value. A score smaller than 50% is usually considered as weak, while a score larger than 75% is usually considered as good (Montgomery et al., 2012; Olive, 2017).

In this synthetic simulation, the coefficient of determination is almost 1, and of course clearly validates the choice of a linear regression model.

1.4.3 CONSTRAINED MINIMIZATION

For different reasons, there can be constraints on the input x. The simplest case is bound constraints, imposing for instance that x is positive, or smaller than some

given value.

In simple cases (e.g. bounds), a gradient algorithm with projection can handle the constraints: after each iteration, the new approximation of the minimum might not satisfy the bounds, so it is projected onto the desired interval of admissible values. But a more efficient and global way to handle constraints, in particular when these are not only bound constraints, is to define a Lagrangian, based on the cost function, that incorporates all the constraints (Allaire, 2007; chong and Zak, 2013).

As an example, if we consider the case where x represents weights, that must be positive and of total sum 1, then the Lagrangian will be defined as:

$$\mathscr{L}(x;p,q) = J_\varepsilon(x) - \langle p,x\rangle + q\left(\sum_{i=1}^{n} x_i - 1\right),$$

where the x_i ($1 \leq i \leq n$) represent the scalar components of x, $\langle\,,\rangle$ denotes the usual inner product, $p \in \mathbb{R}_+^n$ is a positive Lagrange multiplier associated to the positivity constraints of all components of x, and $q \in \mathbb{R}$ is a scalar Lagrange multiplier associated to the constraint that the sum of all components of x must be 1.

Under some standard assumptions, minimizing J_ε under the given constraints is equivalent to finding a saddle-point (i.e. a min-max) of the Lagrangian. Standard algorithms are based on alternatively minimizing the Lagrangian with respect to x and maximizing it with respect to the Lagrange multipliers, see e.g. Uzawa algorithm (Uzawa, 1958; Allaire, 2007; chong and Zak, 2013).

Figure 1.9 shows, as in the unconstrained case, the evolution of the parameters during the optimization process. Here, the constraints are the following:

- Top: all $\beta_i \geq 0$;
- Bottom left: all $\beta_i \geq 0$ and $\sum_{i=0}^{p} \beta_i = 10$;
- Bottom right: $\sum_{i=0}^{p} \beta_i = 1$ (without any sign constraints).

On the top figure, there is almost no difference with Figure 1.8, as the coefficients were almost positive in the unconstrained case. The final identified values are very close, but as one can see, the algorithm does not consider negative values during the iterations (contrary to Figure 1.8 in the first iterations). The final solution being almost the same, the R^2 coefficient is still very high (and close to the unconstrained case).

On the bottom left figure, we add a total sum constraint on the coefficients (here 10). As the total sum of the parameters in the unconstrained case is close to 11 (see previous section for the target values), the algorithm has to adapt the values, so that the final identified non-zero coefficients have been slightly reduced in order to satisfy the total sum constraint, and are now $\beta_0 = 0.97$, $\beta_4 = 1.41$ and $\beta_7 = 7.63$. The R^2 coefficient is still very high, as the β coefficients remain close to the unconstrained case.

Finally, on the bottom right figure, we relax the positivity constraint, but we now impose a total sum of parameters equal to 1. In this case, the identified β_0 is back to almost 1 (0.99), $\beta_4 = 1.12$ and $\beta_7 = 7.14$, but all other β are now significantly different from 0 and negative, so as to compensate the three other positive parameters.

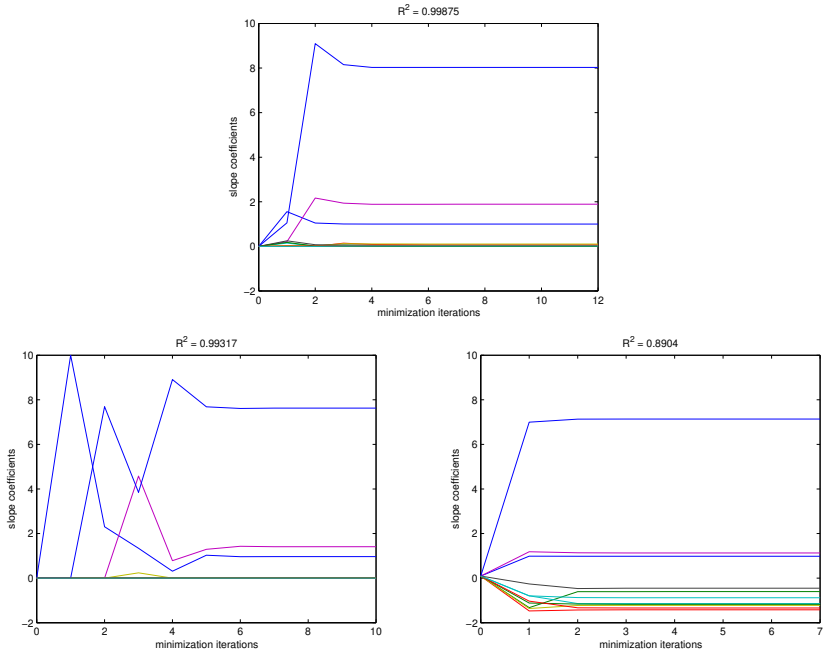

Figure 1.9: Evolution of the identified coefficients β during the optimization process (iterations). Top: positivity constraints on all parameters; Bottom: positivity constraints on all parameters and additional constraint that the total sum of the parameters is equal to 10 (left), and only one constraint that the total sum of the parameters is equal to 1 (right).

Finally, the coefficient R^2 is equal to 89%, which is still reasonable in this synthetic experiment, as the main explanatory variables are still identified, and somehow the other negative parameters compensate and do not degrade too much the quality of the model.

1.4.4 GRADIENT EVALUATION

As previously seen, minimizing a cost function J_ε often requires at least the evaluation of its gradient ∇J_ε, in order to update the minimizing sequence. Sometimes, the gradient has an explicit expression, and is then straightforward to implement. For instance, the gradient of the cost function given by Equation (1.14) is:

$$\nabla J_\varepsilon(x) = 2M^T(Mx-y) + 2\varepsilon x.$$

In finite dimension, when the matrix M is already implemented, multiplying $Mx-y$ by the transpose of M is usually not an issue. But if the linear operator M is the result of a black-box code, or if it involves the resolution of differential equations in infinite dimension, it might become much more difficult to compute the gradient, as the operator M^T is not implemented.

As an example, let consider the following linear operator M: $x \in \mathscr{L}^2(\Omega) \mapsto u \in \mathscr{H}_0^1(\Omega)$, where u is the solution of

$$\begin{cases} -\Delta u = x & \text{in } \Omega, \\ u = 0 & \text{in } \partial\Omega, \end{cases} \tag{1.15}$$

Ω being an open subset of $\mathbb{R}^n$. Equation (1.15) is known as the Poisson equation (Evans, 1998). The inverse problem consists then in finding x such that Mx, i.e. the solution u of (1.15), is the closest possible to the data y. As y is not necessarily regular, computing explicitely $-\Delta y$ in order to find x is not possible.

The gradient of the cost function can be obtained from its definition:

$$\lim_{h\to 0} \frac{J_\varepsilon(x+hz) - J_\varepsilon(x)}{h} = \langle \nabla J_\varepsilon(x), z\rangle = \int_\Omega \nabla J_\varepsilon(x) z, \tag{1.16}$$

where z is a direction of perturbation. By linearity of the model (1.15), $M(x+hz) = Mx + hMz$, where Mz is the function $\tilde{u} \in \mathscr{H}_0^1(\Omega)$ such that $-\Delta\tilde{u} = z$ in Ω.

The cost function J_ε being quadratic, it is easy to see that Equation (1.16) leads to

$$\langle \nabla J_\varepsilon(x), z\rangle = \int_\Omega \nabla J_\varepsilon(x) z = \int_\Omega \left(2(u-y)\tilde{u} + 2\varepsilon x z\right). \tag{1.17}$$

Evaluating Mx, i.e. computing u solution of (1.15), is usually not an issue. It is also easy to compute $\tilde{u} = Mz$ for any direction z. But as it can be seen in (1.17), the expression of ∇J_ε is not explicit. Only the evaluation of the gradient in some direction z can be explicitely done, thanks to the computation of $\tilde{u}$ (Laporte and Le Tallec, 2003; Plessix, 2006; Cea, 1986; Steiner and Reichl, 2012).

In such cases, there are several ways to evaluate, at least approximately/numerically, the gradient.

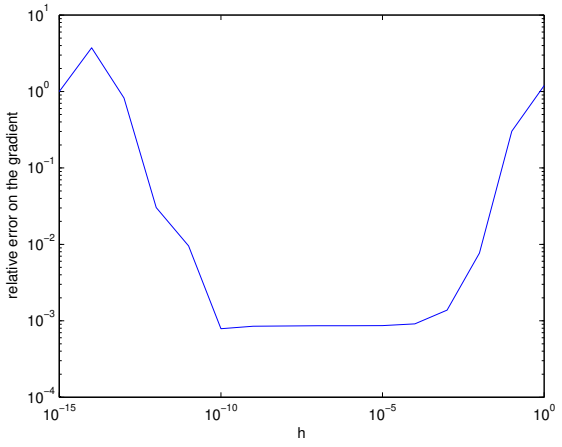

Figure 1.10: Relative error on the gradient computation (difference between the exact gradient and its approximation by finite differences) versus h, in log-log scale.

1.4.4.1 Finite differences

In finite dimension, the simplest method, on the paper, is to use finite differences. Indeed, using the definition of the derivative, for any $z \in \mathbb{R}^n$,

$$\langle \nabla J_\varepsilon(x), z \rangle = \lim_{h \to 0} \frac{J_\varepsilon(x+hz) - J_\varepsilon(x)}{h},$$

so that for a small enough scalar h, the left hand side can be approximated by the finite difference (right hand side without the limit). The point is that one evaluation of the right hand side gives just one projection of the gradient, onto the direction of perturbation z. Typically, using z equal to one of the Euclidean basis vectors, i.e. a vector with only one non-zero component, which is equal to 1, the left hand side is then equal to the corresponding component of the gradient:

$$\frac{\partial J_\varepsilon(x)}{\partial x_i} = \langle \nabla J_\varepsilon(x), e_i \rangle \simeq \frac{J_\varepsilon(x+he_i) - J_\varepsilon(x)}{h}, \tag{1.18}$$

with $e_i = (0; \ldots; 0; 1; 0; \ldots; 0)$, and h small enough.

Evaluating the gradient with finite differences is quite easy, as there is always a quite large range of values of h such that the finite differences give almost the same value. The main drawback is that the full evaluation of the gradient (its n components) requires then $n+1$ evaluations of the cost function, 1 for the reference value $J_\varepsilon(x)$, and n for the perturbations in the n basis directions.

Figure 1.10 shows the relative difference between the exact gradient and its approximation computed by finite differences (see Equation (1.18)) for various values of the parameter h. As shown on this figure, if h is too large, the approximation is no more valid. This is easily explained by a Taylor series expansion, telling us that the

error will be of the order of h. But when h is too small, the error starts to reincrease, leading to dramatic errors for very small values of h: this comes from numerical errors, when dividing a very small number by another one. Even if there is a quite wide range of parameters h for which the error is almost stable, it is not that small with around 0.1% of error at the minimum. This shows that even with a good parameter h, the computed gradient by finite differences is just an approximation of the true value, that might lead to poor optimization performances.

1.4.4.2 Adjoint method

In large (or infinite) dimension, or when M involves the resolution of differential equations (see e.g. Equation (1.15)), another much more efficient way to compute the gradient is called the adjoint method (Steiner and Reichl, 2012; Laporte and Le Tallec, 2003).

As the operator $M : x \mapsto u$ solution of (1.15) is linear, we can consider its adjoint: $M^T : u \mapsto p$ solution of

$$\begin{cases} -\Delta p = u - y & \text{in } \Omega, \\ p = 0 & \text{in } \partial\Omega. \end{cases} \tag{1.19}$$

This allows us to rewrite the directional derivative of the cost function:

$$\langle \nabla J_\varepsilon(x), z \rangle = \int_\Omega (2(u-y)\tilde{u} + 2\varepsilon xz) = \int_\Omega (-2\Delta p\ \tilde{u} + 2\varepsilon xz)\,.$$

The first term in the integral can be integrated by parts, twice:

$$\int_\Omega -2\Delta p\ \tilde{u} = \int_\Omega 2\nabla p \cdot \nabla \tilde{u} = \int_\Omega 2p(-\Delta \tilde{u}) = \int_\Omega 2pz,$$

so that

$$\langle \nabla J_\varepsilon(x), z \rangle = \int_\Omega (2p + 2\varepsilon x)\, z.$$

We now have an explicit expression of the gradient thanks to the adjoint state:

$$\nabla J_\varepsilon(x) = 2p + 2\varepsilon x. \tag{1.20}$$

This explicit expression requires only 2 resolutions: one of the direct model (1.15) for computing u, and one of the adjoint model (1.19) for computing p. Not only Equation (1.20) gives an exact expression of the gradient, but it requires only 2 model resolutions (i.e. one evaluation of M and one of M^T), which is much more efficient and accurate than finite differences.

The main drawback appears when the operator in M, the Laplacian in our example, is not symmetric, so that the adjoint operator is not the same as the direct operator, leading to additional work in the numerical code. This is also true when the discretized operator, used for numerical resolutions of the equation, is not symmetric. The adjoint resolution indeed requires the use of the adjoint of the discretized operator corresponding to M, not the discretized version of M^T (Lauß et al., 2016).

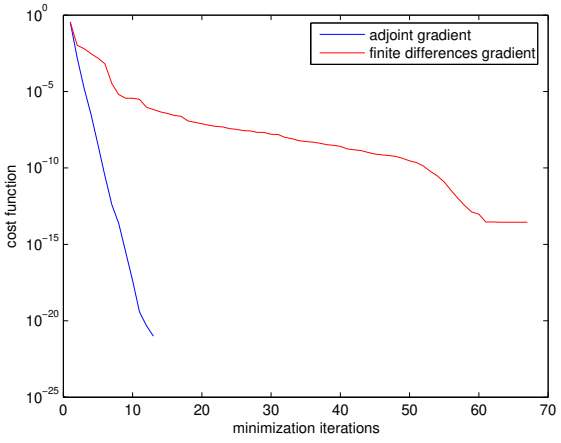

Figure 1.11: Evolution of the cost function J_ε during the minimization, using the exact gradient provided by the adjoint (blue) or the approximated gradient computed by finite differences, versus the minimization iterations.

In the framework of Equation (1.15), in dimension 1, discretized using $n = 100$ grid points, Figure 1.11 shows the evolution of the cost function J_ε during the minimization, using either the exact gradient provided by the adjoint method (in blue), or the approximated gradient computed by finite differences (in red) (see previous section for the finite differences computation). As previously explained, the gradient computed by finite differences is just an approximation of the exact gradient, so that the minimization process cannot be as efficient as when providing the exact gradient. The algorithm needs 67 iterations to achieve convergence (note that it did not converge, the algorithm stopped due to a too small step size, the provided approximate gradient being no more a descent direction) with the approximated gradient, but only 13 iterations with the exact gradient (and it stopped because of a too small update between two iterations).

Note also that one iteration using the adjoint costs one resolution of the direct model (1.15) and one resolution of the adjoint model (1.19), while one iteration using the approximated gradient by finite differences costs $n+1 = 101$ resolutions of the direct equation (1 for the cost, and 1 for each of the 100 components of the gradient). The computing time required for the minimization with the approximated gradient is then 260 times larger than the time required for the minimization using the adjoint, with 5 times more iterations, plus each iteration costing 50 times more model resolutions. Note also that it not only requires a much larger computing time, but it also leads to a poorer identification of the solution.

1.5 PROBABILITY AND INVERSE PROBLEMS

Inverse problems can also be solved using stochastic approaches. In this approach, both data y and unknown x are considered as random variables, determined by their probability density functions (PDF) $f(x)$ and $f(y)$ that measure the likelihood to be close to a given value.

The direct problem gives information about $f(y|x)$, the conditional probability of the output y of the model, given the input x. The inverse problem consists then in computing $f(x|y)$, the inverse probability that represents the knowledge of x given the measures y.

From Bayes' theorem (Bayes, 1763; Hartigan, 1983; Joyce, 2003; Swinburne, 2002; Jeffreys, 1973), the density functions satisfy:

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)}. \tag{1.21}$$

In this equation, $f(x)$ represents the prior probability, which quantifies the knowledge about x before observing the output y of the model. This prior probability is then multiplied by $f(y|x)$ (which is often referred to as the likelihood function). It is also divided by $f(y)$, which can be seen as a renormalization step. Finally, Equation (1.21) gives the value of $f(x|y)$, the posterior probability: it measures the information that we know about x after observing the output y of the model.

As an example, we consider here a simple transport equation in 1D:

$$\begin{cases} \dfrac{\partial u}{\partial t} = a\dfrac{\partial u}{\partial s}, & t \in [0;2],\ s \in [0;10], \\ u(t=0,s) = u_0(s), & s \in [0;10], \end{cases} \tag{1.22}$$

where $u(t,s)$ is the solution, t and s represent the time and space variables respectively. We assume that the initial condition u_0 is known:

$$u_0(s) = e^{-\frac{(s-5)^2}{2}}$$

is a Gaussian function centered at 5 and of standard deviation 1. We assume periodic boundary conditions.

Figure 1.12 represents the solution of Equation (1.22) with the true velocity $a = 5$, and the corresponding noisy observations at final time (in blue):

$$y = u(T,s) + \eta,$$

where η is a Gaussian white noise (with a 0.1 standard deviation).

The inverse problem is the following: from (noisy) observations y of the final solution $u(T,s)$ (with $T = 2$), we want to recover the transport velocity a.

From Bayes' theorem, the posterior probability of a knowing y satisfies:

$$p(a|y) \propto p(y|a)p(a),$$

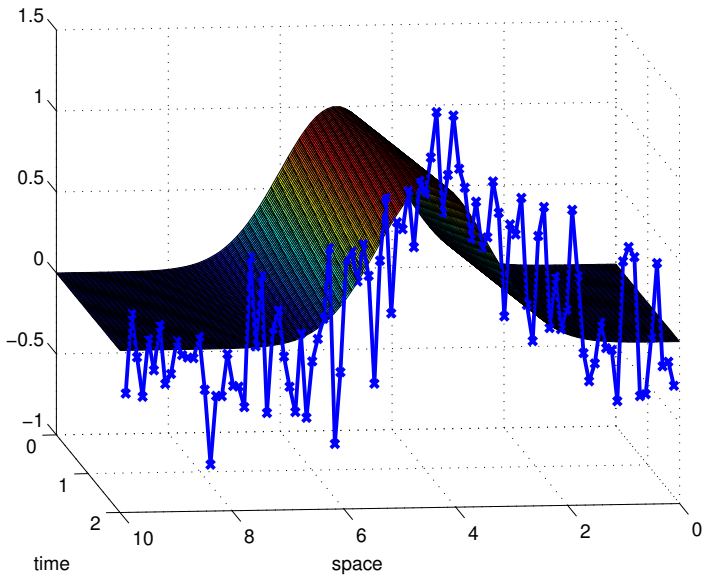

Figure 1.12: Solution of Equation (1.22) and noisy observations of the solution at final time (in blue)

forgetting the renormalization term. As we do not assume anything on a here, the prior distribution of a is uniform. Note that any prior information about the parameter can be used here. Then $p(a|y) \propto p(y|a)$, the likelihood function.

Given a velocity a, as we know the initial condition $u_0(s)$, it is then possible to solve Equation (1.22) and get $u_a(T,s)$ for this value of a. Note that we add a as a subscript of u, the solution depending on this parameter value. The probability that we observe the data y is then

$$p(y|a) = p(\eta = y - u_a(T,s)) \propto e^{-\frac{(y-u_a(T,s))^2}{2*0.1^2}},$$

from the noise distribution of η.

We then have the following posterior distribution of a:

$$p(a|y) \propto e^{-\frac{(y-u_a(T,s))^2}{2*0.1^2}}.$$

We can then estimate a using Bayesian inference, e.g. with Markov chain Monte-Carlo (MCMC) algorithms, that build sequences of samples of a drawn from the posterior probability (Robert and Casella, 2004; Gilks et al., 1995; McElreath, 2016; Murphy, 2012).

Figure 1.13 shows the evoluation of the samples from the posterior probability in blue, starting from the initial guess $a_0 = 2$, and the true value of the velocity

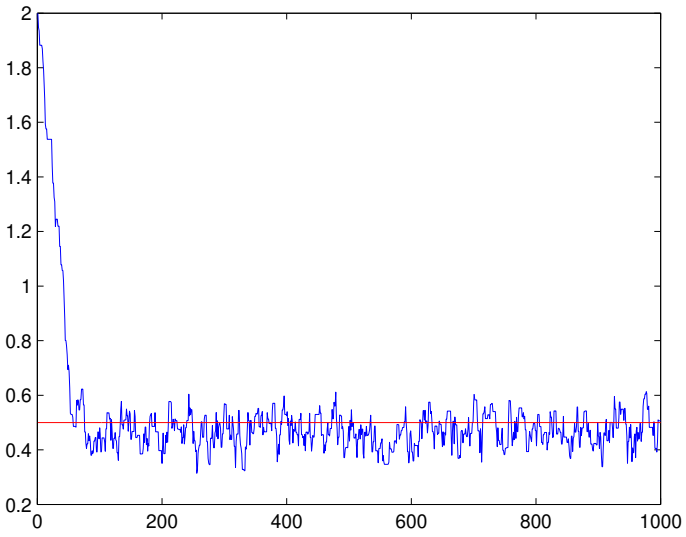

Figure 1.13: Evolution of the samples from the posterior probability versus MCMC updates (blue), and true value of the parameter a (red)

$a = 0.5$ in red. As one can see, the samples quickly converge to a neighborhood of the true parameter. From a deterministic point of view, the mean of the MCMC updates (after a large enough number of iterations) can be seen as the solution of the inverse problem, along with standard statistics (standard deviation, histogram, ...).

1.6 REFERENCES

Bibliography

Allaire, G. (2007). *Numerical analysis and optimization: an introduction to mathematical modelling and numerical simulation*. Oxford Univ. Press.

Allaire, G. and S. M. Kaber (2008). *Numerical linear algebra*. Springer.

Aster, R. C., B. Borchers, and C. H. Thurber (2008). *Parameter Estimation and Inverse Problems* (3rd ed.). Elsevier.

Aubert, G. and P. Kornprobst (2006). *Mathematical problems in image processing* (2nd ed.). Springer.

Auroux, D. and M. Masmoudi (2009). Image processing by topological asymptotic expansion. *J. Math. Imaging Vision 33*(2), 122–134.

Babaeizadeh, S. and R. Ahmad (2012). *Line search methods in conjugate gradient algorithms*. Lambert Academic Publishing.

Bakushinsky, A. and A. Goncharsky (1994). *Ill-posed problems: theory and applications*. Springer.

Bayes, T. (1763). An essay toward solving a problem in the doctrine of chances. *Philos. Trans. R. Soc. 53*, 370–418.

Beck, J. V. and K. J. Arnold (1977). *Parameter estimation in engineering and science*. John Wiley and Sons.

Beck, J. V., B. Blackwell, and C. R. S. Clair Jr. (1985). *Inverse Heat Conduction*. Wiley, Hoboken.

Bertero, M., P. Boccacci, and C. De Mol (2021). *Introduction to inverse problems in imaging* (2nd ed.). CRC Press.

Bierlaire, M. (2015). *Optimization principles and algorithms*. CRC Press.

Bôcher, M. (1909). *An introduction to the study of integral equations*. Cambridge Univ. Press.

Burden, R. L., D. J. Faires, and A. M. Burden (2015). *Numerical analysis* (10th ed.). Cengage Learning.

Cea, J. (1986). Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût. *ESAIM: Math. Model. Numer. Anal. 20*(3), 371–402.

Chalmond, B. (2003). *Modeling and Inverse Problems in Imaging Analysis*. Springer New York.

Choi, M.-D. (1983). Tricks or treats with the hilbert matrix. *Am. Math. Mon. 90*(5), 301–312.

chong, E. K. P. and S. H. Zak (2013). *An introduction to optimization* (4th ed.). Wiley.

Engl, H. W. (1987). Discrepancy principles for tikhonov regularization of ill-posed problems leading to optimal convergence rates. *J. Optim. Th. Appl. 52*, 209–215.

Engl, H. W. and C. W. Groetsch (Eds.) (1987). *Inverse and ill-posed problems*. Academic Press.

Evans, L. C. (1998). *Partial Differential Equations*. Am. Math. Soc.

Friedman, A. and M. Vogelius (1989). Determining cracks by boundary measurements. *Indiana Univ. Math. J. 38*(2), 497–525.

Gilks, W. R., N. G. Best, and K. K. C. Tan (1995). Adaptive rejection metropolis sampling within gibbs sampling. *J. R. Stat. Soc. Ser. C Appl. Stat. 44*(4), 455–472.

Groetsch, C. W. (1984). *The theory of Tikhonov regularization for Fredholm equations*. Boston Pitman Publication.

Groetsch, C. W. (2007). Integral equations of the first kind, inverse problems and regularizatoin: a crash course. *J. Phys: Conf. Ser. 73*.

Hadamard, J. (1902). Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton Univ. Bulletin 14*(4), 49–52.

Hansen, P. C., J. G. Nagy, and D. P. O'Leary (2006). *Deblurring images*. Fundamentals of algorithms. SIAM.

Hansen, P. C. and D. P. O'Leary (1993). The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput. 14*(6), 1487–1503.

Hartigan, J. A. (1983). *Bayes Theory*. Springer-Verlag.

Hensel, E. (1991). *Inverse theory and applications for engineers*. Prentice Hall.

Hilbert, D. (1894). Ein beitrag zur theorie des legendre'schen polynoms. *Acta Math. 18*, 155–159.

Hofmann, B. and S. Kindermann (2010). On the degree of ill-posedness for linear problems with non-compact operators. *Methods Appl. Anal. 17*(4), 445–462.

Jeffreys, H. (1973). *Scientific Inference*. Cambridge Univ. Press.

Joyce, J. (2003). Bayes' theorem. In E. N. Zalta (Ed.), *Stanford Encyclopedia of Philosophy* (Spring 2019 ed.). Metaphysics Research Lab, Stanford University.

Kirsch, A. (1996). *An introduction to the mathematical theory of inverse problems*. Springer.

Kohn, R. and M. Vogelius (1984). Determining conductivity by boundary measurements. *Comm. Pure Appl. Math. 37*, 289–298.

Laporte, E. and P. Le Tallec (2003). Computing gradients by adjoint states. In *Numericla methods in sensitivity analysis and shape optimization*, pp. 87–95. Birkhäuser.

Lauß, T., S. Oberpeilsteiner, W. Steiner, and K. Nachbagauer (2016). The discrete adjoint gradient computation for optimization problems in multibody dynamics. *J. Comput. Nonlinear Dyn. 12*(3), 031016.

Lavrent'ev, M. M. and L. Y. Savel'ev (2006). *Operator theory and ill-posed problems*, Volume 50 of *Inverse and Ill-Posed Problems Series*. De Gruyter.

McElreath, R. (2016). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC Press.

Montgomery, D. C., E. A. Peck, and G. G. Vining (2012). *Introduction to linear regression analysis* (5th ed.). Wiley.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Natterer, F. (2001). *The mathematics of computerized tomography*. SIAM Philadelphia.

Ólafsson, G. and E. T. Quinto (Eds.) (2006). *The Radon Transform, Inverse Problems, and Tomography*, Volume 63 of *Proceedings of Symposia in Applied Mathematics*. Amer. Math. Soc., Providence.

Olive, D. J. (2017). *Linear regression*. Springer.

Plessix, R.-E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophys. J. Int. 167*(2), 495–503.

Pytlak, R. (2010). *Conjugate gradient algorithms in nonconvex optimization*. Springer.

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods*. Springer.

Scherzer, O. (Ed.) (2011). *Handbook of Mathematical Methods in Imaging*. Springer New York.

Schwartz, L. (1981). *Cours d'analyse*. Hermann.

Steiner, W. and S. Reichl (2012). The optimal control approach to dynamical inverse problems. *ASME J. Dyn. Syst., Meas., Control 134*(2), 021010.

Swinburne, R. (2002). *Bayes' Theorem*. Oxford Univ. Press.

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM.

Tikhonov, A. N. and V. Y. Arsenin (1977). *Solutions of ill-posed problems*. Winston-Wiley.

Uzawa, H. (1958). Iterative methods for concave programming. In K. J. Arrow, L. Hurwicz, and H. Uzawa (Eds.), *Studies in linear and nonlinear programming*. Stanford Univ. Press.

Vogel, C. R. (2002). *Computational methods for inverse problems*. SIAM.