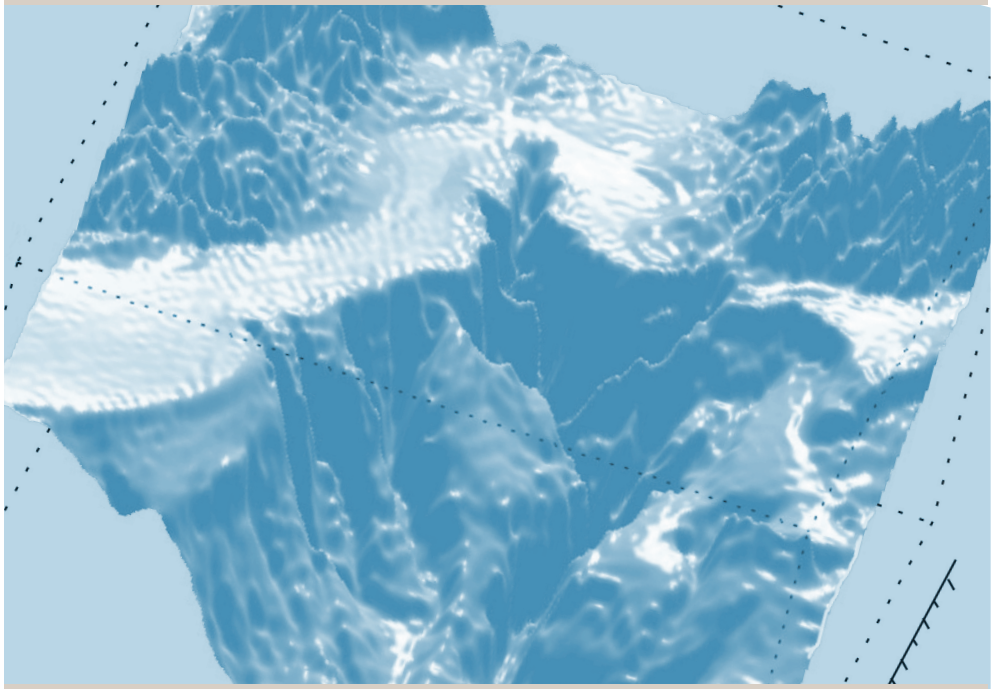
Least Squares Data Fitting with Applications



Per Christian Hansen
Víctor Pereyra
Godela Scherer

Least Squares Data Fitting with Applications

Per Christian Hansen

Department of Informatics and Mathematical Modeling

Technical University of Denmark

Víctor Pereyra

Energy Resources Engineering

Stanford University

Godela Scherer

Department of Mathematics and Statistics

University of Reading

Contents

Foreword

Scientific computing is founded in models that capture the properties of systems under investigation, be they engineering systems; systems in natural sciences; financial, economic, and social systems; or conceptual systems such as those that arise in machine learning or speech processing. For models to be useful, they must be calibrated against "real-world" systems and informed by data. The recent explosion in availability of data opens up unprecedented opportunities to increase the fidelity, resolution, and power of models — but only if we have access to algorithms for incorporating this data into models, effectively and efficiently.

For this reason, least squares — the first and best-known technique for fitting models to data — remains central to scientific computing. This problem class remains a fascinating topic of study from a variety of perspectives. Least-squares formulations can be derived from statistical principles, as maximum-likelihood estimates of models in which the model-data discrepancies are assumed to arise from Gaussian white noise. In scientific computing, they provide the vital link between model and data, the final ingredient in a model that brings the other elements together. In their linear variants, least-squares problems were a foundational problem in numerical linear algebra, as this field grew rapidly in the 1960s and 1970s. From the perspective of optimization, nonlinear least-squares has appealing structure that can be exploited with great effectiveness in algorithm design.

Least squares is foundational in another respect: It can be extended in a variety of ways: to alternative loss functions that are more robust to outliers in the observations, to one-sided "hinge loss" functions, to regularized models that impose structure on the model parameters in addition to fitting the data, and to "total least squares" models in which errors appear in the model coefficients as well as the observations.

This book surveys least-squares problems from all these perspectives. It is both a comprehensive introduction to the subject and a valuable resource to those already well versed in the area. It covers statistical motivations along with thorough treatments of direct and iterative methods for linear least squares and optimization methods for nonlinear least squares. The

later chapters contain compelling case studies of both linear and nonlinear models, with discussions of model validation as well as model construction and interpretation. It conveys both the rich history of the subject and its ongoing importance, and reflects the many contributions that the authors have made to all aspects of the subject.

Stephen Wright
Madison, Wisconsin, USA

Preface

This book surveys basic modern techniques for the numerical solution of linear and nonlinear least squares problems and introduces the treatment of large and ill-conditioned problems. The theory is extensively illustrated with examples from engineering, environmental sciences, geophysics and other application areas.

In addition to the treatment of the numerical aspects of least squares problems, we introduce some important topics from the area of regression analysis in statistics, which can help to motivate, understand and evaluate the computed least squares solutions. The inclusion of these topics is one aspect that distinguishes the present book from other books on the subject.

The presentation of the material is designed to give an overview, with the goal of helping the reader decide which method would be appropriate for a given problem, point toward available algorithms/software and, if necessary, help in modifying the available tools to adapt for a given application. The emphasis is therefore on the properties of the different algorithms and few proofs are presented; the reader is instead referred to the appropriate articles/books. Unfortunately, several important topics had to be left out, among them, direct methods for sparse problems.

The content is geared toward scientists and engineers who must analyze and solve least squares problems in their fields. It can be used as course material for an advanced undergraduate or graduate course in the sciences and engineering, presupposing a working knowledge of linear algebra and basic statistics. It is written mostly in a terse style in order to provide a quick introduction to the subject, while treating some of the not so well-known topics in more depth. This in fact presents the reader with an opportunity to verify the understanding of the material by completing or providing the proofs without checking the references.

The least squares problem is known under different names in different disciplines. One of our aims is to help bridge the communication gap between the statistics and the numerical analysis literature on the subject, often due to the use of different terminology, such as l_2-approximation,

regularization, regression analysis, parameter estimation, filtering, process identification, etc.

Least squares methods have been with us for many years, since Gauss invented and used them in his surveying activities [83]. In 1965, the paper by G. H. Golub [92] on using the QR factorization and later his development of a stable algorithm for the computation of the SVD started a renewed interest in the subject in the, by then, changed work environment of computers.

Thanks also to, among many others, Å. Björck, L. Eldén, C. C. Paige, M. A. Saunders, G. W. Stewart, S. van Huffel and P.-Å. Wedin, the topic is now available in a robust, algorithmic and well-founded form.

There are many books partially or completely dedicated to linear and nonlinear least squares. The first and one of the fundamental references for linear problems is Lawson and Hanson's monograph [150]. Besides summarizing the state of the art at the time of its publication, it highlighted the practical aspects of solving least squares problems. Bates and Watts [9] have an early comprehensive book focused on the nonlinear least squares problem with a strong statistical approach. Björck's book [20] contains a very careful and comprehensive survey of numerical methods for both linear and nonlinear problems, including the treatment of large, sparse problems. Golub and Van Loan's *Matrix Computations* [105] includes several chapters on different aspects of least squares solution and on total least squares. The total least squares problem, known in statistics as latent root regression, is discussed in the book by S. van Huffel and J. Vandewalle [239]. Seber and Wild [223] consider exhaustively all aspects of nonlinear least squares estimation and modeling. Although it is a general treatise on optimization, Nocedal and Wright's book [170] includes a very clear chapter on nonlinear least squares. Additional material can be found in [21, 63, 128, 232, 242, 251].

We would like to acknowledge the help of Michael Saunders (iCME, Stanford University), who read carefully the whole manuscript and made a myriad of observations and corrections that have greatly improved the final product.

Per Christian Hansen would like to thank several colleagues from DTU Informatics who assisted with the statistical aspects.

Godela Scherer gives thanks for all the support at the Department of Mathematics and Statistics, University of Reading, where she was a visiting research fellow while working on this book. In particular, she would like to thank Professor Mike J. Baines and Dr. I Llatas for numerous inspiring discussions.

Victor Pereyra acknowledges Weidlinger Associates Inc. and most especially David Vaughan and Howard Levine, for their unflagging support

and for letting him keep his office and access to computing facilities after retirement. Also Ms. P. Tennant helped immensely by improving many of the figures.

Special thanks are due to the professional handling of the manuscript by the publishers and more specifically to executive editor Vincent J. Burke and production editor Andre Barnett.

Prior to his untimely death on November 2007, Professor Gene Golub had been an integral part of this project team. Although the book has changed significantly since then, it has greatly benefited from his insight and knowledge. He was an inspiring mentor and great friend, and we miss him dearly.

Symbols and Acronyms

Symbol	Represents
A	$m \times n$ matrix
$A^\dagger$, A^-, A^T	pseudoinverse, generalized inverse and transpose of A
$\boldsymbol{b}$	right-hand side, length m
$\mathrm{cond}(\cdot)$	condition number of matrix in l_2-norm
$\mathrm{Cov}(\cdot)$	covariance matrix
$\mathrm{diag}(\cdot)$	diagonal matrix
$\boldsymbol{e}$	vector of noise, length m
e_i	noise component in data
$\widehat{\boldsymbol{e}}_i$	canonical unit vector
ε_M	machine precision
$\mathcal{E}(\cdot)$	expected value
$f_j(t)$	model basis function
$\Gamma(t)$	pure-data function
$M(\boldsymbol{x}, t)$	fitting model
$\mathcal{N}$	normal (or Gaussian) distribution
$\mathrm{null}(A)$	null space of A
p	degree of polynomial
P, P_i, $P_{\boldsymbol{x}}$	probability
$\mathcal{P}_X$	projection onto space X
Π	permutation matrix
Q	$m \times m$ orthogonal matrix, partitioned as $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$
$r = r(A)$	rank of matrix A
$\boldsymbol{r}$, $\boldsymbol{r}^*$	residual vector, least squares residual vector
r_i	residual for ith data
$\mathrm{range}(A)$	range of matrix A
R, R_1	$m \times n$ and $n \times n$ upper triangular matrix
$\mathrm{span}\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_p\}$	subspace generated by the vectors
Σ	diagonal SVD matrix

Symbol	Represents
ς, ς_i	standard deviation
σ_i	singular value
t	independent variable in data fitting problem
t_i	abscissa in data fitting problem
U, V	$m \times m$ and $n \times n$ left and right SVD matrices
$\boldsymbol{u}_i$, $\boldsymbol{v}_i$	left and right singular vectors, length m and n respectively
W	$m \times m$ diagonal weight matrix
w_i	weight in weighted least squares problem
$\boldsymbol{x}$, $\boldsymbol{x}^*$	vector of unknowns, least squares solution, length n
$\widehat{\boldsymbol{x}}^*$	minimum-norm LSQ solution
$\boldsymbol{x}_{\mathrm{B}}$, $\boldsymbol{x}_{\mathrm{TLS}}$	basic LSQ solution and total least squares solution
x_i	coefficient in a linear fitting model
$\boldsymbol{y}$	vector of data in a data fitting problem, length m
y_i	data in data fitting problem
$\|\cdot\|_2$	2-norm, $\|\boldsymbol{x}\|_2 = (x_1^2 + \cdots + x_n^2)^{1/2}$
$\widetilde{\square}$	perturbed version of $\square$

Acronym	**Name**
CG	conjugate gradient
CGLS	conjugate gradient for LSQ
FG	fast Givens
GCV	generalized cross validation
G-N	Gauss-Newton
GS	Gram-Schmidt factorization
GSVD	generalized singular value decomposition
LASVD	SVD for large, sparse matrices
L-M	Levenberg-Marquardt
LP	linear prediction
LSE	equality constrained LSQ
LSI	inequality constrained LSQ
LSQI	quadratically constrained LSQ
LSQ	least squares
LSQR	Paige-Saunders algorithm
LU	LU factorization
MGS	modified Gram-Schmidt
NLLSQ	nonlinear least squares
NMR	nuclear magnetic resonance
NN	neural network
QR	QR decomposition
RMS	root mean square
RRQR	rank revealing QR decomposition
SVD	singular value decomposition
TLS	total least squares problem
TSVD	truncated singular value decomposition
UTV	UTV decomposition
VARPRO	variable projection algorithm, Netlib version
VP	variable projection

Chapter 1

The Linear Data Fitting Problem

This chapter gives an introduction to the linear data fitting problem: how it is defined, its mathematical aspects and how it is analyzed. We also give important statistical background that provides insight into the data fitting problem. Anyone with more interest in the subject is encouraged to consult the pedagogical expositions by Bevington [13], Rust [213], Strutz [233] and van den Bos [242].

We start with a couple of simple examples that introduce the basic concepts of data fitting. Then we move on to a more formal definition, and we discuss some statistical aspects. Throughout the first chapters of this book we will return to these data fitting problems in order to illustrate the ensemble of numerical methods and techniques available to solve them.

1.1 Parameter estimation, data approximation

Example 1. ***Parameter estimation.*** *In food-quality analysis, the amount and mobility of water in meat has been shown to affect quality attributes like appearance, texture and storage stability. The water contents can be measured by means of nuclear magnetic resonance (NMR) techniques, in which the measured signal reflects the amount and properties of different types of water environments in the meat. Here we consider a simplified example involving frozen cod, where the ideal time signal $\phi(t)$ from NMR is a sum of two damped exponentials plus a constant background,*

$$\phi(t) = x_1\, e^{-\lambda_1 t} + x_2\, e^{-\lambda_2 t} + x_3, \qquad \lambda_1, \lambda_2 > 0.$$

In this example we assume that we know the parameters λ_1 and λ_2 that control the decay of the two exponential components. In practice we do not

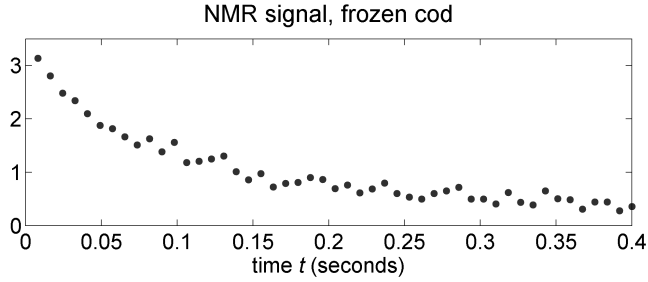


Figure 1.1.1: Noisy measurements of the time signal $\phi(t)$ from NMR, for the example with frozen cod meat.

measure this pure signal, but rather a noisy realization of it as shown in Figure 1.1.1.

The parameters $\lambda_1 = 27\,\mathrm{s}^{-1}$ *and* $\lambda_2 = 8\,\mathrm{s}^{-1}$ *characterize two different types of proton environments, responsible for two different water mobilities. The amplitudes* x_1 *and* x_2 *are proportional to the amount of water contained in the two kinds of proton environments. The constant* x_3 *accounts for an undesired background (bias) in the measurements. Thus, there are three unknown parameters in this model, namely,* x_1*,* x_2 *and* x_3*. The goal of data fitting in relation to this problem is to use the measured data to estimate the three unknown parameters and then compute the different kinds of water contents in the meat sample. The actual fit is presented in Figure 1.2.1.*

In this example we used the technique of data fitting for the purpose of estimating unknown parameters in a mathematical model from measured data. The model was dictated by the physical or other laws that describe the data.

Example 2. ***Data approximation.*** *We are given measurements of air pollution, in the form of the concentration of NO, over a period of 24 hours, on a busy street in a major city. Since the NO concentration is mainly due to the cars, it has maximum values in the morning and in the afternoon, when the traffic is most intense. The data is shown in Table 1.1 and the plot in Figure 1.2.2.*

For further analysis of the air pollution we need to fit a smooth curve to the measurements, so that we can compute the concentration at an arbitrary time between 0 and 24 hours. For example, we can use a low-degree polynomial to model the data, i.e., we assume that the NO concentration can be approximated by

$$f(t) = x_1\, t^p + x_2\, t^{p-1} + \cdots + x_p\, t + x_{p+1},$$

t_i	y_i	t_i	y_i	t_i	y_i	t_i	y_i	t_i	y_i
0	110.49	5	29.37	10	294.75	15	245.04	20	216.73
1	73.72	6	74.74	11	253.78	16	286.74	21	185.78
2	23.39	7	117.02	12	250.48	17	304.78	22	171.19
3	17.11	8	298.04	13	239.48	18	288.76	23	171.73
4	20.31	9	348.13	14	236.52	19	247.11	24	164.05

Table 1.1: Measurements of NO concentration y_i as a function of time t_i. The units of y_i and t_i are $\mu g/m^3$ and hours, respectively.

where t is the time, p is the degree of the polynomial and $x_1, x_2, \ldots, x_{p+1}$ are the unknown coefficients in the polynomial. A better model however, since the data repeats every day, *would use periodic functions:*

$$f(t) = x_1 + x_2 \sin(\omega t) + x_3 \cos(\omega t) + x_4 \sin(2\omega t) + x_5 \cos(2\omega t) + \cdots$$

where $\omega = 2\pi/24$ is the period. Again, $x_1, x_2, \ldots$ are the unknown coefficients. The goal of data fitting in relation to this problem is to estimate the coefficients $x_1, x_2, \ldots$, such that we can evaluate the function $f(t)$ for any argument t. At the same time we want to suppress the influence of errors present in the data.

> *In this example we used the technique of data fitting for the purpose of approximating measured discrete data: we fitted a model to given data in order to be able to compute smoothed data for any value of the independent variable in the model. We were free to choose the model as long as it gave an adequate fit to the data.*

Both examples illustrate that we are given data with measurement errors and that we want to fit a model to these data that captures the "overall behavior" of it without being too sensitive to the errors. The difference between the two examples is that in the first case the model arises from a physical theory, while in the second there is an arbitrary continuous approximation to a set of discrete data.

Data fitting is distinctly different from the problem of interpolation, where we seek a model – a function $f(t)$ – that interpolates the given data, i.e., it satisfies $f(t_i) = y_i$ for all the data points. We are not interested in interpolation (which is not suited for noisy data) – rather, we want to approximate the noisy data with a parametric model that is either given or that we can choose, in such a way that the result is not too sensitive to the noise. In this data fitting approach there is redundant data: i.e., more data than unknown parameters, which also helps to decrease the uncertainty in

the parameters of the model. See Example 15 in the next chapter for a justification of this.

1.2 Formulation of the data fitting problem

Let us now give a precise definition of the data fitting problem. We assume that we are given m *data points*

$$(t_1, y_1),\ (t_2, y_2), \ldots,\ (t_m, y_m),$$

which can be described by the relation

$$y_i = \Gamma(t_i) + e_i, \qquad i = 1, 2, \ldots, m. \tag{1.2.1}$$

The function $\Gamma(t)$, which we call the *pure-data function*, describes the noise-free data (it may be unknown, or given by the application), while $e_1, e_2, \ldots, e_m$ are the data errors (they are unknown, but we may have some statistical information about them). The data errors – also referred to as noise – represent measurement errors as well as random variations in the physical process that generates the data. Without loss of generality we can assume that the abscissas t_i appear in non-decreasing order, i.e., $t_1 \leq t_2 \leq \cdots \leq t_m$.

In data fitting we wish to compute an approximation to $\Gamma(t)$ – typically in the interval $[t_1, t_m]$. The approximation is given by the *fitting model* $M(\boldsymbol{x}, t)$, where the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ contains n parameters that characterize the model and are to be determined from the given noisy data. In the linear data fitting problem we always have a model of the form

$$\boxed{\text{Linear fitting model:} \qquad M(\boldsymbol{x}, t) = \sum_{j=1}^{n} x_j\, f_j(t).} \tag{1.2.2}$$

The functions $f_j(t)$ are called the *model basis functions,* and the number n – the *order* of the fit – should preferably be smaller than the number m of data points. A notable modern exception is related to the so-called compressed sensing, which we discuss briefly in Section 7.5.

The form of the function $M(\boldsymbol{x}, t)$ – i.e., the choice of basis functions – depends on the precise goal of the data fitting. These functions may be given by the underlying mathematical model that describes the data – in which case $M(\boldsymbol{x}, t)$ is often equal to, or an approximation to, the pure-data function $\Gamma(t)$ – or the basis functions may be chosen arbitrarily among all functions that give the desired approximation and allow for stable numerical computations.

The method of least squares (LSQ) is a standard technique for determining the unknown parameters in the fitting model. The *least squares fit*

is defined as follows. We introduce the residual r_i associated with the data points as

$$r_i = y_i - M(\boldsymbol{x}, t_i), \qquad i = 1, 2, \ldots, m,$$

and we note that each residual is a function of the parameter vector $\boldsymbol{x}$, i.e., $r_i = r_i(\boldsymbol{x})$. A least squares fit is a choice of the parameter vector $\boldsymbol{x}$ that minimizes the sum-of-squares of the residuals:

$$\boxed{\text{LSQ fit:} \quad \min_{\boldsymbol{x}} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2 = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \big(y_i - M(\boldsymbol{x}, t_i)\big)^2.} \tag{1.2.3}$$

In the next chapter we shall describe in which circumstances the least squares fit is unique, and in the following chapters we shall describe a number of efficient computational methods for obtaining the least squares parameter vector $\boldsymbol{x}$.

We note in passing that there are other related criteria used in data fitting; for example, one could replace the sum-of-squares in (1.2.3) with the sum-of-absolute-values:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{m} |r_i(\boldsymbol{x})| = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \big|y_i - M(\boldsymbol{x}, t_i)\big|. \tag{1.2.4}$$

Below we shall use a statistical perspective to describe when these two choices are appropriate. However, we emphasize that the book focuses on the least squares fit.

In order to obtain a better understanding of the least squares data fitting problem we take a closer look at the residuals, which we can write as

$$\begin{aligned} r_i = y_i - M(\boldsymbol{x}, t_i) &= \big(y_i - \Gamma(t_i)\big) + \big(\Gamma(t_i) - M(\boldsymbol{x}, t_i)\big) \\ &= e_i + \big(\Gamma(t_i) - M(\boldsymbol{x}, t_i)\big), \\ &\quad i = 1, 2, \ldots, m. \end{aligned} \tag{1.2.5}$$

We see that the ith residual consists of two components: the data error e_i comes from the measurements, while the approximation error $\Gamma(t_i) - M(\boldsymbol{x}, t_i)$ is due to the discrepancy between the pure-data function and the computed fitting model. We emphasize that even if $\Gamma(t)$ and $M(\boldsymbol{x}, t)$ have the same form, there is no guarantee that the estimated parameters $\boldsymbol{x}$ used in $M(\boldsymbol{x}, t)$ will be identical to those underlying the pure-data function $\Gamma(t)$. At any rate, we see from this dichotomy that a good fitting model $M(\boldsymbol{x}, t)$ is one for which the approximation errors are of the same size as the data errors.

Underlying the least squares formulation in (1.2.3) are the assumptions that the data and the errors are independent and that the errors are "white noise." The latter means that all data errors are uncorrelated and of the

same size – or in more precise statistical terms, that the errors e_i have mean zero and identical variance: $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma^2$ for $i = 1, 2, \ldots, m$ (where ς is the standard deviation of the errors).

This ideal situation is not always the case in practice! Hence, we also need to consider the more general case where the standard deviation depends on the index i, i.e.,

$$\mathcal{E}(e_i) = 0, \qquad \mathcal{E}(e_i^2) = \varsigma_i^2, \qquad i = 1, 2, \ldots, m,$$

where ς_i is the standard deviation of e_i. In this case, the maximum likelihood principle in statistics (see Section 1.3), tells us that we should minimize the weighted residuals, with weights equal to the reciprocals of the standard deviations:

$$\boxed{\min_x \sum_{i=1}^m \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i}\right)^2 = \min_x \sum_{i=1}^m \left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2.} \tag{1.2.6}$$

Now consider the expected value of the weighted sum-of-squares:

$$\begin{aligned}
\mathcal{E}\left(\sum_{i=1}^m \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i}\right)^2\right) &= \sum_{i=1}^m \mathcal{E}\left(\frac{r_i(\boldsymbol{x})^2}{\varsigma_i^2}\right) \\
&= \sum_{i=1}^m \mathcal{E}\left(\frac{e_i^2}{\varsigma_i^2}\right) + \sum_{i=1}^m \mathcal{E}\left(\frac{(\Gamma(t_i) - M(\boldsymbol{x}, t_i))^2}{\varsigma_i^2}\right) \\
&= m + \sum_{i=1}^m \frac{\mathcal{E}\big((\Gamma(t_i) - M(\boldsymbol{x}, t_i))^2\big)}{\varsigma_i^2},
\end{aligned}$$

where we have used that $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma_i^2$. The consequence of this relation is the intuitive result that we can allow the expected value of the approximation errors to be larger for those data (t_i, y_i) that have larger standard deviations (i.e., larger errors). Example 4 illustrates the usefulness of this approach. See Chapter 3 in [233] for a thorough discussion on how to estimate weights for a given data set.

We are now ready to state the least squares data fitting problem in terms of matrix-vector notation. We define the matrix $A \in \mathbb{R}^{m \times n}$ and the vectors $\boldsymbol{y}, \boldsymbol{r} \in \mathbb{R}^m$ as follows,

$$A = \begin{pmatrix} f_1(t_1) & f_2(t_1) & \cdots & f_n(t_1) \\ f_1(t_2) & f_2(t_2) & \cdots & f_n(t_2) \\ \vdots & \vdots & & \vdots \\ f_1(t_m) & f_2(t_m) & \cdots & f_n(t_m) \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix},$$

i.e., $\boldsymbol{y}$ is the vector of observations, $\boldsymbol{r}$ is the vector of residuals and the matrix A is constructed such that the jth column is the jth model basis

function sampled at the abscissas $t_1, t_2, \ldots, t_m$. Then it is easy to see that for the un-weighted data fitting problem we have the relations

$$\boldsymbol{r} = \boldsymbol{y} - A\,\boldsymbol{x} \qquad \text{and} \qquad \rho(\boldsymbol{x}) = \sum_{i=1}^{m} r_i(\boldsymbol{x})^2 = \|\boldsymbol{r}\|_2^2 = \|\boldsymbol{y} - A\,\boldsymbol{x}\|_2^2.$$

Similarly, for the weighted problem we have

$$\rho_W(\boldsymbol{x}) = \sum_{i=1}^{m} \left(\frac{r_i(\boldsymbol{x})}{\varsigma_i} \right)^2 = \|W\,(\boldsymbol{y} - A\,\boldsymbol{x})\|_2^2,$$

with the weighting matrix and weights

$$W = \mathrm{diag}(w_1, \ldots, w_m), \qquad w_i = \varsigma_i^{-1}, \quad i = 1, 2, \ldots, m.$$

In both cases, the computation of the coefficients in the least squares fit is identical to the solution of a linear least squares problem for $\boldsymbol{x}$. Throughout the book we will study these least squares problems in detail and give efficient computational algorithms to solve them.

Example 3. *We return to the NMR data fitting problem from Example 1. For this problem there are 50 measured data points and the model basis functions are*

$$f_1(t) = e^{-\lambda_1 t}, \qquad f_2(t) = e^{-\lambda_2 t}, \qquad f_3(t) = 1,$$

and hence we have $m = 50$ *and* $n = 3$*. In this example the errors in all data points have the same standard deviation* $\varsigma = 0.1$*, so we can use the un-weighted approach. The solution to the* 50×3 *least squares problem is*

$$x_1 = 1.303, \qquad x_2 = 1.973, \qquad x_3 = 0.305.$$

The exact parameters used to generate the data are 1.27*,* 2.04 *and* 0.3*, respectively. These data were then perturbed with random errors. Figure 1.2.1 shows the data together with the least squares fit* $M(\boldsymbol{x}, t)$*; note how the residuals are distributed on both sides of the fit.*

For the data fitting problem in Example 2, we try both the polynomial fit and the trigonometric fit. In the first case the basis functions are the monomials $f_j(t) = t^{n-j}$*, for* $j = 1, \ldots, n = p + 1$*, where* p *is the degree of the polynomial. In the second case the basis functions are the trigonometric functions:*

$$f_1(t) = 1, \quad f_2(t) = \sin(\omega\, t), \quad f_3(t) = \cos(\omega\, t),$$

$$f_4(t) = \sin(2\omega\, t), \quad f_5(t) = \cos(2\omega\, t), \quad \ldots$$

Figure 1.2.2 shows the two fits using a polynomial of degree $p = 8$ *(giving a fit of order* $n = 9$*) and a trigonometric fit with* $n = 9$*. The trigonometric fit looks better. We shall later introduce computational tools that let us investigate this aspect in more rigorous ways.*

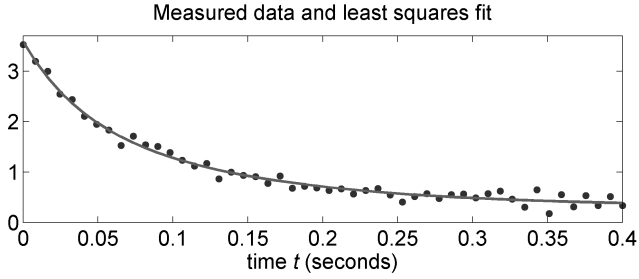


Figure 1.2.1: The least squares fit (solid line) to the measured NMR data (dots) from Figure 1.1.1 in Example 1.

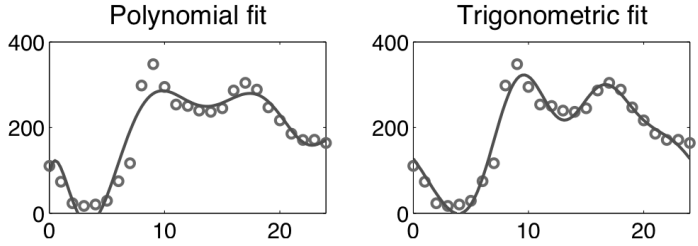


Figure 1.2.2: Two least squares fits (both of order $n = 9$) to the measured NO data from Example 2, using a polynomial (left) and trigonometric functions (right).

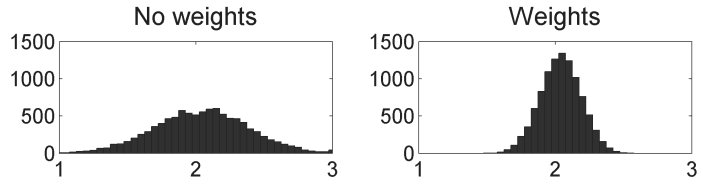


Figure 1.2.3: Histograms of the computed values of x_2 for the modified NMR data in which the first 10 data points have larger errors. The left plot shows results from solving the un-weighted LSQ problem, while the right plot shows the results when weights $w_i = \varsigma_i^{-1}$ are included in the LSQ problem.

Example 4. *This example illustrates the importance of using weights when computing the fit. We use again the NMR data from Example 1, except this time we add larger Gaussian noise to the first 10 data points, with standard deviation* 0.5*. Thus we have* $\varsigma_i = 0.5$ *for* $i = 1, 2, \ldots, 10$ *(the first 10 data with larger errors) and* $\varsigma_i = 0.1$ *for* $i = 11, 12, \ldots, 50$ *(the remaining data with smaller errors). The corresponding weights* $w_i = \varsigma_i^{-1}$ *are therefore* $2, 2, \ldots, 2, 10, 10, \ldots, 10$*. We solve the data fitting problem with and without weights for* $10,000$ *instances of the noise. To evaluate the results, we consider how well we estimate the second parameter* x_2 *whose exact value is* 2.04*. The results are shown in Figure 1.2.3 in the form of histograms of the computed values of* x_2*. Clearly, the weighted fit gives more robust results because it is less influenced by the data with large errors.*

1.3 Maximum likelihood estimation

At first glance the problems of interpolation and data fitting seem to resemble each other. In both cases, we use approximation theory to select a good model (through the model basis functions) for the given data that results in small approximation errors – in the case of data fitting, given by the term $\Gamma(t) - M(\boldsymbol{x}, t)$ for $t \in [t_1, t_m]$. The main difference between the two problems comes from the presence of data errors and the way we deal with these errors.

In data fitting we deliberately avoid interpolating the data and instead settle for less degrees of freedom in the model, in order to reduce the model's sensitivity to errors. Also, it is clear that the data noise plays an important role in data fitting problems, and we should use concepts and tools from statistics to deal with it.

The classical statistical motivation for the least squares fit is based on the maximum likelihood principle. Our presentation follows [13]. We assume that the data are given by (1.2.1), that the errors e_i are unbiased

and uncorrelated, and that each error e_i has a Gaussian distribution with standard deviation ς_i, i.e.,

$$y_i = \Gamma(t_i) + e_i, \qquad e_i \sim \mathcal{N}(0, \varsigma_i^2).$$

Here, $\mathcal{N}(0, \varsigma_i^2)$ denotes the normal (Gaussian) distribution with zero mean and standard deviation σ_i. Gaussian errors arise, e.g., from the measurement process or the measuring devices, and they are also good models of composite errors that arise when several sources contribute to the noise. Then the probability P_i for making the observation y_i is given by

$$P_i = \frac{1}{\varsigma_i\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\varsigma_i}\right)^2} = \frac{1}{\varsigma_i\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2}.$$

The probability P for making the observed set of measurements $y_1, y_2, \ldots, y_m$ is the product of these probabilities:

$$P = P_1\, P_2\, \cdots\, P_m = \prod_{i=1}^{m} \frac{1}{\varsigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2} = K \prod_{i=1}^{m} e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma_i}\right)^2},$$

where the factor $K = \prod_{i=1}^{m} \left(\varsigma_i\sqrt{2\pi}\right)^{-1}$ is independent of the pure-data function.

Now assume that the pure-data function $\Gamma(t)$ is identical to the fitting model $M(\boldsymbol{x}, t)$, for a specific but unknown parameter vector $\boldsymbol{x}^*$. The probability for the given data $y_1, y_2, \ldots, y_m$ to be produced by the model $M(\boldsymbol{x}, t)$ for an arbitrary parameter vector $\boldsymbol{x}$ is given by

$$P_{\boldsymbol{x}} = K \prod_{i=1}^{m} e^{-\frac{1}{2}\left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2} = K\, e^{-\frac{1}{2}\sum_{i=1}^{m}\left(\frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i}\right)^2}. \tag{1.3.1}$$

The method of maximum likelihood applied to data fitting consists of making the assumption that the given data are more likely to have come from our model $M(\boldsymbol{x}, t)$ – with specific but unknown parameters $\boldsymbol{x}^*$ – than any other model of this form (i.e., with any other parameter vector $\boldsymbol{x}$). The best estimate of the model is therefore the one that maximizes the probability $P_{\boldsymbol{x}}$ given above, as a function of $\boldsymbol{x}$. Clearly, $P_{\boldsymbol{x}}$ is maximized by minimizing the exponent, i.e., the weighted sum-of-squared residuals. We have thus – under the assumption of Gaussian errors – derived the weighted least squares data fitting problem (1.2.6). In practice, we use the same principle when there is a discrepancy between the pure-data function and the fitting model.

The same approach can be used to derive the maximum likelihood fit for other types of errors. As an important example, we consider errors that follow a Laplace distribution, for which the probability of making the

observation y_i is given by

$$P_i = \frac{1}{2\varsigma_i}\, e^{-\frac{|y_i - \Gamma(t_i)|}{\varsigma_i}} = \frac{1}{2\varsigma_i}\, e^{-\frac{|e_i|}{\varsigma_i}},$$

where again ς_i is the standard deviation. The Laplace density function decays slower than the Gaussian density function, and thus the Laplace distribution describes measurement situations where we are more likely to have large errors than in the case of the Gaussian distribution.

Following once again the maximum likelihood approach we arrive at the problem of maximizing the function

$$\begin{aligned} P_{\boldsymbol{x}} &= \widehat{K} \prod_{i=1}^{m} e^{-\frac{|y_i - M(\boldsymbol{x}, t_i)|}{\varsigma_i}} \\ &= \widehat{K}\, e^{-\sum_{i=1}^{m} \left| \frac{y_i - M(\boldsymbol{x}, t_i)}{\varsigma_i} \right|} \end{aligned}$$

with $\widehat{K} = \prod_{i=1}^{m} \big(2\varsigma_i\big)^{-1}$. Hence, for these errors we should minimize the sum-of-absolute-values of the weighted residuals. This is the linear 1-norm minimization problem that we mentioned in (1.2.4).

While the principle of maximum likelihood is universally applicable, it can lead to complicated or intractable computational problems. As an example, consider the case of Poisson data, where y_i comes from a Poisson distribution, with expected value $\Gamma(t_i)$ and standard deviation $\Gamma(t_i)^{1/2}$. Poisson data typically show up in counting measurements, such as the photon counts underlying optical detectors. Then the probability for making the observation y_i is

$$P_i = \frac{\Gamma(t_i)^{y_i}}{y_i!}\, e^{-\Gamma(t_i)},$$

and hence, we should maximize the probability

$$\begin{aligned} P_{\boldsymbol{x}} &= \prod_{i=1}^{m} \frac{M(\boldsymbol{x}, t_i)^{y_i}}{y_i!}\, e^{-M(\boldsymbol{x}, t_i)} \\ &= \prod_{i=1}^{m} \frac{1}{y_i!} \prod_{i=1}^{m} M(\boldsymbol{x}, t_i)^{y_i}\, e^{-\sum_{i=1}^{m} M(\boldsymbol{x}, t_i)}. \end{aligned}$$

Unfortunately, it is computationally demanding to maximize this quantity with respect to $\boldsymbol{x}$, and instead one usually makes the assumption that the Poisson errors for each data y_i are nearly Gaussian, with standard deviation $\sigma_i = \Gamma(t_i)^{1/2} \simeq y_i^{1/2}$ (see, e.g., pp. 342–343 in [146] for a justification of this assumption). Hence, the above weighted least squares approach derived for Gaussian noise, with weights $w_i = y_i^{-1/2}$, will give a good approximation to the maximum likelihood fit for Poisson data.

The Gauss and Laplace errors discussed above are used to model additive errors in the data. We finish this section with a brief look at relative errors, which arise when the size of the error e_i is – perhaps to a good approximation – proportional to the magnitude of the pure data $\Gamma(t_i)$. A straightforward way to model such errors, which fits into the above framework, is to assume that the data y_i can be described by a normal distribution with mean $\Gamma(t_i)$ and standard deviation $\varsigma_i = |\Gamma(t_i)|\,\varsigma$. This "relative Gaussian errors" model can also be written as

$$y_i = \Gamma(t_i)\,(1 + e_i), \qquad e_i \sim \mathcal{N}(0, \varsigma^2). \tag{1.3.2}$$

Then the probability for making the observation y_i is

$$P_i = \frac{1}{|\Gamma(t_i)|\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\Gamma(t_i)\,\varsigma}\right)^2}.$$

Using the maximum likelihood principle again and substituting the measured data y_i for the unknown pure data $\Gamma(t_i)$, we arrive at the following weighted least squares problem:

$$\min_{\boldsymbol{x}} \sum_{i=1}^{m} \left(\frac{y_i - M(\boldsymbol{x}, t_i)}{y_i}\right)^2 = \min_{\boldsymbol{x}} \|W\,(\boldsymbol{y} - A\,\boldsymbol{x})\|_2^2, \tag{1.3.3}$$

with weights $w_i = y_i^{-1}$.

An alternative formulation, which is suited for problems with positive data $\Gamma(t_i) > 0$ and $y_i > 0$, is to assume that y_i can be described by a log-normal distribution, for which $\log y_i$ has a normal distribution, with mean $\log \Gamma(t_i)$ and standard deviation ς:

$$y_i = \Gamma(t_i)\, e^{\epsilon_i} \quad \Leftrightarrow \quad \log y_i = \log \Gamma(t_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, \varsigma^2).$$

In this case we again arrive at a sum-of-squares minimization problem, but now involving the difference of the logarithms of the data y_i and the model $M(\boldsymbol{x}, t_i)$. Even when $M(\boldsymbol{x}, t)$ is a linear model, this is not a linear problem in $\boldsymbol{x}$.

In the above log-normal model with standard deviation ς, the probability P_i for making the observation y_i is given by

$$P_i = \frac{1}{y_i\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{\log y_i - \log \Gamma(t_i)}{\varsigma}\right)^2} = \frac{1}{y_i\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{\log \hat{y}_i}{\varsigma}\right)^2},$$

with $\hat{y}_i = y_i/\Gamma(t_i)$. Now let us assume that ς is small compared to $\Gamma(t_i)$, such that $y_i \simeq \Gamma(t_i)$ and $\hat{y}_i \simeq 1$. Then, we can write $y_i = \Gamma(t_i)\,(1 + e_i) \Leftrightarrow \hat{y}_i = 1 + e_i$, with $e_i \ll 1$ and $\log \hat{y}_i = e_i + O(e_i^2)$. Hence, the exponential

factor in P_i becomes

$$\begin{aligned} e^{-\frac{1}{2}\left(\frac{\log \hat{y}_i}{\varsigma}\right)^2} &= e^{-\frac{1}{2}\left(\frac{\hat{e}_i + O(e_i^2)}{\varsigma}\right)^2} = e^{-\frac{1}{2}\left(\frac{e_i^2}{\varsigma^2} + \frac{O(e_i^3)}{\varsigma^2}\right)} \\ &= e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} e^{-\frac{O(e_i^3)}{\varsigma^2}} = e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} O(1), \end{aligned}$$

while the other factor in P_i becomes

$$\frac{1}{y_i\,\varsigma\sqrt{2\pi}} = \frac{1}{\Gamma(t_i)\,(1+e_i)\,\varsigma\sqrt{2\pi}} = \frac{1+O(e_i)}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}.$$

Hence, as long as $\varsigma \ll \Gamma(t_i)$ we have the approximation

$$P_i \simeq \frac{1}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{e_i}{\varsigma}\right)^2} = \frac{1}{\Gamma(t_i)\,\varsigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{y_i - \Gamma(t_i)}{\Gamma(t_i)\,\varsigma}\right)^2},$$

which is the probability introduced above for the case of "relative Gaussian errors." Hence, for small noise levels $\varsigma \ll |\Gamma(t_i)|$, the two different models for introducing relative errors in the data are practically identical, leading to the same weighted LSQ problem (1.3.3).

1.4 The residuals and their properties

This section focuses on the residuals $r_i = y_i - M(\boldsymbol{x}, t_i)$ for a given fit and how they can be used to analyze the "quality" of the fit $M(\boldsymbol{x}, t)$ that we have computed. Throughout the section we assume that the residuals behave like a time series, i.e., they have a natural ordering $r_1, r_2, \ldots, r_m$ associated with the ordering $t_1 < t_2 < \cdots < t_m$ of the samples of the independent variable t.

As we already saw in Equation (1.2.5), each residual r_i consists of two components – the data error e_i and the approximation error $\Gamma(t_i) - M(\boldsymbol{x}, t_i)$. For a good fitting model, the approximation error should be of the same size as the data errors (or smaller). At the same time, we do not want the residuals to be too small, since then the model $M(\boldsymbol{x}, t)$ may overfit the data: i.e., not only will it capture the behavior of the pure-data function $\Gamma(t)$, but it will also adapt to the errors, which is undesirable.

In order to choose a good fitting model $M(\boldsymbol{x}, t)$ we must be able to analyze the residuals r_i and determine whether the model captures the pure-data function "well enough." We can say that this is achieved when the approximation errors are smaller than the data errors, so that the residuals are practically dominated by the data errors. In that case, some of the statistical properties of the errors will carry over to the residuals. For example, if the noise is white (cf. Section 1.1), then we will expect that the residuals associated with a satisfactory fit show properties similar to white noise.

If, on the other hand, the fitting model does not capture the main behavior of the pure-data function, then we can expect that the residuals are dominated by the approximation errors. When this is the case, the residuals will not have the characteristics of noise, but instead they will tend to behave as a sampled signal, i.e., the residuals will show strong local correlations. We will use the term *trend* to characterize a long-term movement in the residuals when considered as a time series.

Below we will discuss some statistical tests that can be used to determine whether the residuals behave like noise or include trends. These and many other tests are often used in time series analysis and signal processing. Throughout this section we make the following assumptions about the data errors e_i:

- They are random variables with mean zero and identical variance, i.e., $\mathcal{E}(e_i) = 0$ and $\mathcal{E}(e_i^2) = \varsigma^2$ for $i = 1, 2, \ldots, m$.
- They belong to a normal distribution, $e_i \sim \mathcal{N}(0, \varsigma^2)$.

We will describe three tests with three different properties:

- Randomness test: check for randomness of the signs of the residuals.
- Autocorrelation test: check whether the residuals are uncorrelated.
- White noise test: check for randomness of the residuals.

The use of the tools introduced here is illustrated below and in Chapter 11 on applications.

Test for random signs

Perhaps the simplest analysis of the residuals is based on the statistical question: can we consider the signs of the residuals to be random? (Which will often be the case when e_i is white noise with zero mean.) We can answer this question by means of a *run test* from time series analysis; see, e.g., Section 10.4 in [134].

Given a sequence of two symbols – in our case, "+" and "−" for positive and negative residuals r_i – a run is defined as a succession of identical symbols surrounded by different symbols. For example, the sequence "+ + + − − − − − + + − − − − − − + + +" has $m = 17$ elements, $n_+ = 8$ pluses, $n_- = 9$ minuses and $u = 5$ runs: + + +, − − −−, ++, − − − − − and +++. The distribution of runs u (not the residuals!) can be approximated by a normal distribution with mean μ_u and standard deviation ς_u given by

$$\mu_u = \frac{2\, n_+\, n_-}{m} + 1, \qquad \varsigma_u^2 = \frac{(\mu_u - 1)\,(\mu_u - 2)}{m - 1}. \tag{1.4.1}$$

With a 5% significance level we will accept the sign sequence as random if

$$z_{\pm} = \frac{|u - \mu_u|}{\varsigma_u} < 1.96 \tag{1.4.2}$$

(other values of the threshold, for other significance levels, can be found in any book on statistics). If the signs of the residuals are not random, then it is likely that trends are present in the residuals. In the above example with 5 runs we have $z_{\pm} = 2.25$, and according to (1.4.2) the sequence of signs is not random.

Test for correlation

Another question we can ask is whether short sequences of residuals are correlated, which is a clear indication of trends. The autocorrelation of the residuals is a statistical tool for analyzing this. We define the *autocorrelation* ϱ of the residuals, as well as the trend threshold T_ϱ, as the quantities

$$\varrho = \sum_{i=1}^{m-1} r_i\, r_{i+1}, \qquad T_\varrho = \frac{1}{\sqrt{m-1}} \sum_{i=1}^{m} r_i^2. \tag{1.4.3}$$

Since ϱ is the sum of products of neighboring residuals, it is in fact the unit-lag autocorrelation. Autocorrelations with larger lags, or distances in the index, can also be considered. Then, we say that trends are likely to be present in the residuals if the absolute value of the autocorrelation exceeds the trend threshold, i.e., if $|\varrho| > T_\varrho$. Similar techniques, based on shorter sequences of residuals, are used for placing knots in connection with spline fitting; see Chapter 6 in [125].

We note that in some presentations, the mean of the residuals is subtracted before computing ϱ and T_ϱ. In our applications this should not be necessary, as we assume that the errors have zero mean.

Test for white noise

Yet another question we can ask is whether the sequence of residuals behaves like white noise, which can be answered by means of the normalized cumulative periodogram. The underlying idea is that white noise has a flat spectrum, i.e., all frequency components in the discrete Fourier spectrum have the same probability; hence, we must determine whether this is the case. Let the complex numbers $\hat{r}_k$ denote the components of the discrete Fourier transform of the residuals, i.e.,

$$\hat{r}_k = \sum_{i=1}^{m} r_i\, e^{(2\pi\,\hat{\imath}\,(i-1)(k-1)/m)}, \qquad k = 1, \ldots, m,$$

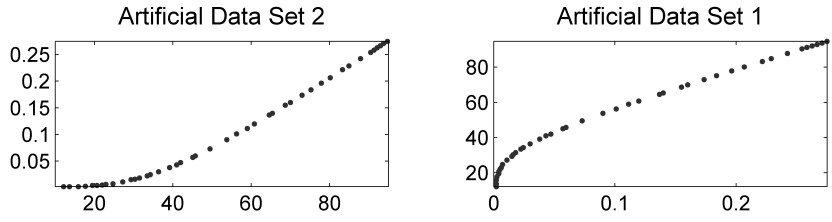


Figure 1.4.1: Two artificially created data sets used in Example 5. The second data set is inspired by the data on p. 60 in [125].

where $\hat{\imath}$ denotes the imaginary unit. Our indices are in the range $1, \ldots, m$ and thus shifted by 1 relative to the range $0, \ldots, m-1$ that is common in signal processing. Note that $\hat{r}_1$ is the sum of the residuals (called the DC component in signal processing), while $\hat{r}_{q+1}$ with $q = \lfloor m/2 \rfloor$ is the component of the highest frequency. The squared absolute values $|\hat{r}_1|^2, |\hat{r}_2|^2, \ldots, |\hat{r}_{q+1}|^2$ are known as the periodogram (in statistics) or the power spectrum (in signal processing). Then the normalized cumulative periodogram consists of the q numbers

$$c_i = \frac{|\hat{r}_2|^2 + |\hat{r}_3|^2 + \cdots + |\hat{r}_{i+1}|^2}{|\hat{r}_2|^2 + |\hat{r}_3|^2 + \cdots + |\hat{r}_{q+1}|^2}, \qquad i = 1, \ldots, q, \qquad q = \lfloor m/2 \rfloor,$$

which form an increasing sequence from 0 to 1. Note that the sums exclude the first term in the periodogram.

If the residuals are white noise, then the expected values of the normalized cumulative periodogram lie on a straight line from $(0, 0)$ to $(q, 1)$. Any realization of white noise residuals should produce a normalized cumulative periodogram close to a straight line. For example, with the common 5% significance level from statistics, the numbers c_i should lie within the Kolmogorov-Smirnoff limit $\pm 1.35/q$ of the straight line. If the maximum deviation $\max_i\{|c_i - i/q|\}$ is smaller than this limit, then we recognize the residual as white noise.

Example 5. ***Residual analysis.*** *We finish this section with an example that illustrates the above analysis techniques. We use the two different data sets shown in Figure 1.4.1; both sets are artificially generated (in fact, the second set is the first set with the t_i and y_i values interchanged). In both examples we have $m = 43$ data points, and in the test for white noise we have $q = 21$ and $1.35/q = 0.0643$. The fitting model $M(\boldsymbol{x}, t)$ is the polynomial of degree $p = n - 1$.*

For fitting orders $n = 2, 3, \ldots, 9$, Figure 1.4.2 shows the residuals and the normalized cumulative periodograms, together with $z_\pm$ (1.4.2), the ratios

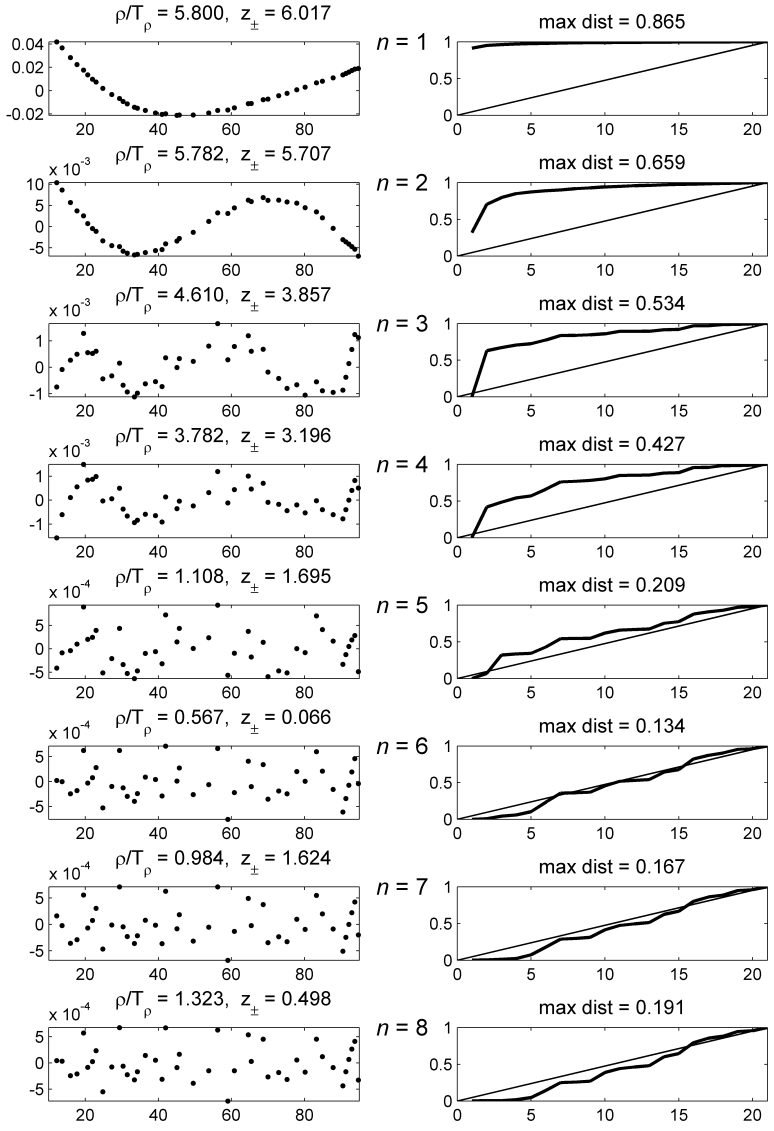


Figure 1.4.2: Residual analysis for polynomial fits to artificial data set 1.

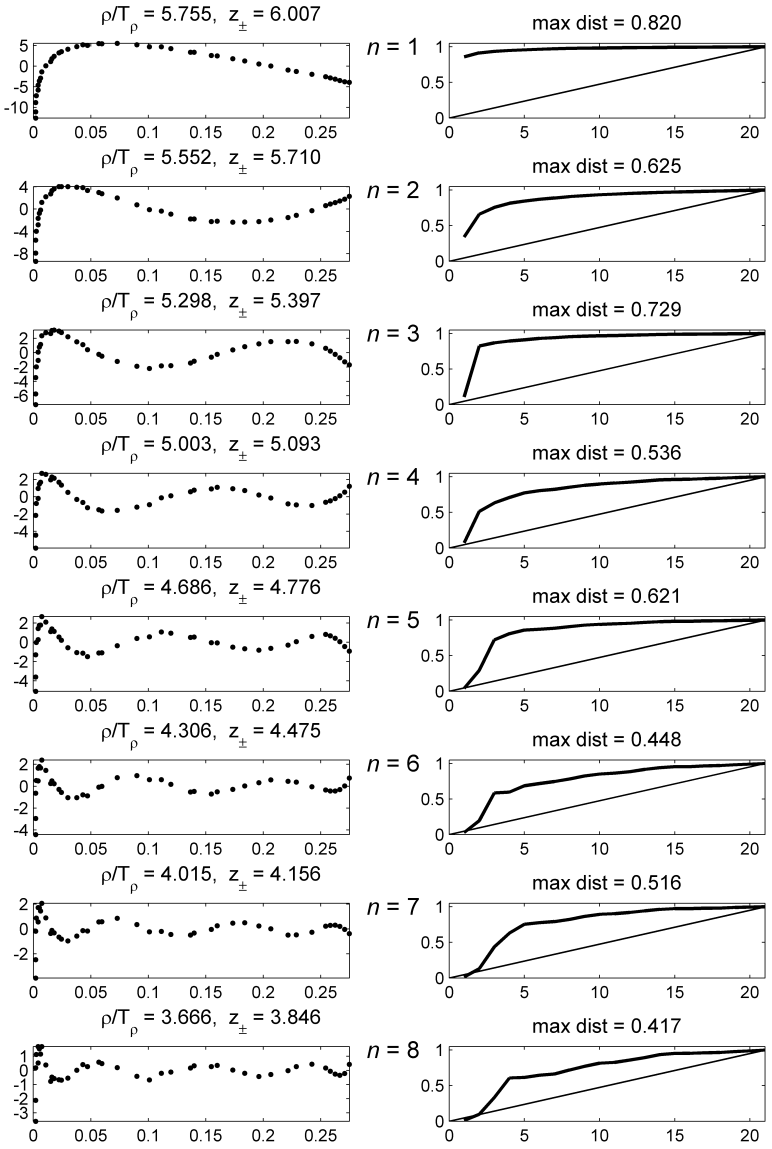


Figure 1.4.3: Residual analysis for polynomial fits to artificial data set 2.

ϱ/T_ϱ from (1.4.3) and the maximum distance of the normalized cumulative periodogram to the straight line. A visual inspection of the residuals in the left part of the figure indicates that for small values of n the polynomial model does not capture all the information in the data, as there are obvious trends in the residuals, while for $n \geq 5$ the residuals appear to be more random. The test for random signs confirms this: for $n \geq 5$ the numbers $z_\pm$ are less than 1.96, indicating that the signs of the residuals could be considered random. The autocorrelation analysis leads to approximately the same conclusion: for $n = 6$ and 7 the absolute value of the autocorrelation ϱ is smaller than the threshold T_ϱ.

The normalized cumulative periodograms are shown in the right part of Figure 1.4.2. For small values, the curves rise fast toward a flat part, showing that the residuals are dominated by low-frequency components. The closest we get to a straight line is for $n = 6$, but the maximum distance 0.134 to the straight line is still too large to clearly signify that the residuals are white noise. The conclusion from these three tests is nevertheless that $n = 6$ is a good choice of the order of the fit.

Figure 1.4.3 presents the residual analysis for the second data set. A visual inspection of the residuals clearly shows that the polynomial model is not well suited for this data set – the residuals have a slowly varying trend for all values of n. This is confirmed by the normalized cumulative periodograms that show that the residuals are dominated by low-frequency components. The random-sign test and the autocorrelation analysis also give a clear indication of trends in the residuals.

1.5 Robust regression

The least squares fit introduced in this chapter is convenient and useful in a large number of practical applications – but it is not always the right choice for a data fitting problem. In fact, we have already seen in Section 1.3 that the least squares fit is closely connected to the assumption about Gaussian errors in the data. There we also saw that other types of noise, in the framework of maximum likelihood estimation, lead to other criteria for a best fit – such as the sum-of-absolute-values of the residuals (the 1-norm) associated with the Laplace distribution for the noise. The more dominating the "tails" of the probablility density function for the noise, the more important it is to use another criterion than the least squares fit.

Another situation where the least squares fit is not appropriate is when the data contain *outliers*, i.e., observations with exceptionally large errors and residuals. We can say that an outlier is a data point (t_i, y_i) whose value y_i is unusual compared to its predicted value (based on all the reliable data points). Such outliers may come from different sources:

- The data errors may come from more than one statistical distribution. This could arise, e.g., in an astronomical CCD camera, where we have Poisson noise (or "photon noise") from the incoming light, Gaussian noise from the electronic circuits (amplifier and A/D-converter), and occasional large errors from cosmic radiation (so-called cosmic ray events).

- The outliers may be due to data recording errors arising, e.g., when the measurement device has a malfunction or the person recording the data makes a blunder and enters a wrong number.

A manual inspection can sometimes be used to delete blunders from the data set, but it may not always be obvious which data are blunders or outliers. Therefore we prefer to have a mathematical formulation of the data fitting problem that handles outliers in such a way that all data are used, and yet the outliers do not have a deteriorating influence on the fit. This is the goal of *robust regression*. Quoting from [82] we say that "an estimator or statistical procedure is robust if it provides useful information even if some of the assumptions used to justify the estimation method are not applicable."

Example 6. ***Mean and median.*** *Assume we are given $n-1$ samples $z_1, \ldots, z_{n-1}$ from the same distribution and a single sample z_n that is an outlier. Clearly, the arithmetic mean $\frac{1}{n}(z_1 + z_2 + \cdots + z_n)$ is not a good estimate of the expected value because the outlier constributes with the same weight as all the other data points. On the other hand, the median gives a robust estimate of the expected value since it is insensitive to a few outliers; we recall that if the data are sorted then the median is $z_{(n+1)/2}$ if n is odd, and $\frac{1}{2}(z_{n/2} + z_{n/2+1})$ if n is even.*

The most common method for robust data fitting – or robust regression, as statisticians call it – is based on the principle of M-estimation introduced by Huber [130], which can be considered as a generalization of maximum likelihood estimation. Here we consider un-weighted problems only (the extension to weighted problems is straightforward). The underlying idea is to replace the sum of squared residuals in (1.2.3) with the sum of some function of the residuals:

$$\boxed{\text{Robust fit:} \quad \min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\big(r_i(\boldsymbol{x})\big) = \min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\big(y_i - M(\boldsymbol{x}, t_i)\big),} \tag{1.5.1}$$

where the function ρ defines the contribution of each residual to the function to be minimized. In particular, we obtain the least squares fit when $\rho(r) = \frac{1}{2}r^2$. The function ρ must satisfy the following criteria:

1. Non-negativity: $\rho(r) \geq 0 \; \forall r$.

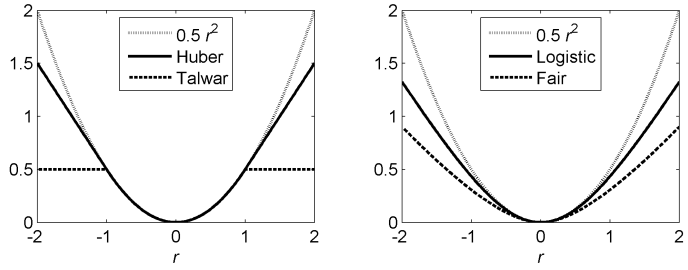


Figure 1.5.1: Four functions $\rho(r)$ used in the robust data fitting problem. All of them increase slower than $\frac{1}{2}r^2$ that defines the LSQ problem and thus they lead to robust data fitting problems that are less sensitive to outliers.

2. Zero only when the argument is zero: $\rho(r) = 0 \Leftrightarrow r = 0$.
3. Symmetry: $\rho(-r) = \rho(r)$.
4. Monotonicity: $\rho(r') \geq \rho(r)$ for $r' \geq r$.

Some well-known examples of the function ρ are (cf. [82, 171]):

$$\text{Huber}: \quad \rho(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \beta \\ \beta|r| - \frac{1}{2}\beta^2, & |r| > \beta \end{cases} \tag{1.5.2}$$

$$\text{Talwar}: \quad \rho(r) = \begin{cases} \frac{1}{2}r^2, & |r| \leq \beta \\ \frac{1}{2}\beta^2, & |r| > \beta \end{cases} \tag{1.5.3}$$

$$\text{Bisquare}: \quad \rho(r) = \beta^2 \log\left(\cosh\left(\frac{z}{\beta}\right)\right) \tag{1.5.4}$$

$$\text{Logistic}: \quad \rho(r) = \beta^2 \left(\frac{|r|}{\beta} - \log\left(1 + \frac{|r|}{\beta}\right)\right) \tag{1.5.5}$$

Note that all four functions include a problem-dependent positive parameter β that is used to control the behavior of the function for "large" values of r, corresponding to the outliers. Figure 1.5.1 shows these functions for the case $\beta = 1$, and we see that all of them increase slower than the function $\frac{1}{2}r^2$, which underlies the LSQ problem. This is precisely why they lead to a robust data fitting problem whose solution is less sensitive to outliers than the LSQ solution. The parameter β should be chosen from our knowledge of the standard deviation ς of the noise; if ς is not known, then it can be estimated from the fit as we will discuss in Section 2.2.

It appears that the choice of function ρ for a given problem relies mainly on experience with the specific data for that problem. Still, the Huber

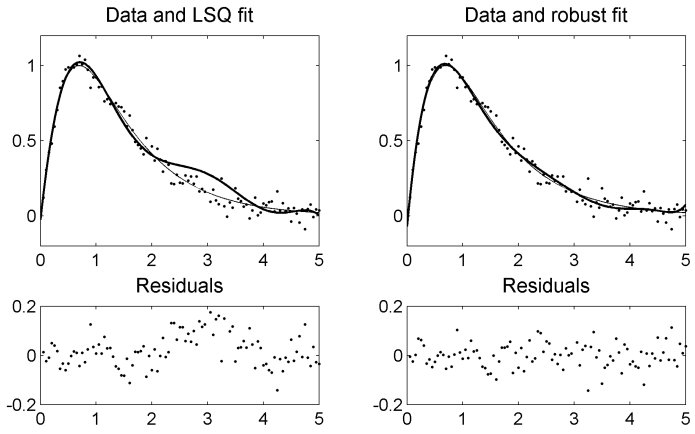


Figure 1.5.2: The pure-data function $\Gamma(t)$ (thin line) and the data with Gaussian noise (dots); the outlier $(t_{60}, y_{60}) = (3, 2.5)$ is outside the plot. The fitting model $M(\boldsymbol{x}, t)$ is a polynomial with $n = 9$. Left: the LSQ fit and the corresponding residuals; this fit is dramatically influenced by the outlier. Right: the robust Huber fit, using $\beta = 0.025$, together with the residuals; this is a much better fit to the given data because it approximates the pure-data function well.

function has attained widespread use, perhaps due to the natural way it distinguishes between large and small residuals:

- Small residual satisfying $|r_i| \leq \beta$ are treated in the same way as in the LSQ fitting problem; if there are no outliers then we obtain the LSQ solution.

- Large residuals satisfying $|r_i| > \beta$ are essentially treated as $|r_i|$ and the robust fit is therefore not so sensitive to the corresponding data points.

Thus, robust regression is a compromise between excluding the outliers entirely from the analysis and including all the data points and treating all of them equally in the LSQ regression. The idea of robust regression is to weight the observations differently based on how well behaved these observations are. For an early use in seismic data processing see [48, 219].

Example 7. ***Robust data fitting with the Huber function.*** *This example illustrates that the Huber function gives a more robust fit than the LSQ fit. The pure-data funtion* $\Gamma(t)$ *is given by*

$$\Gamma(t) = \sin\left(\pi e^{-t}\right), \qquad 0 \leq t \leq 5.$$

We use $m = 100$ data points with $t_i = 0.05i$, and we add Gaussian noise with standard deviation $\varsigma = 0.05$. Then we change the 60th data point to an outlier with $(t_{60}, y_{60}) = (3, 2.5)$; Figure 1.5.2 shows the function $\Gamma(t)$ and the noisy data. We note that the outlier is located outside the plot.

As fitting model $M(\boldsymbol{x}, t)$ we use a polynomial with $n = 9$ and the left part of Figure 1.5.2 shows the least squares fit with this model, together with the corresponding residuals. Clearly, this fit is dramatically influenced by the outlier, which is evident from the plot of the fit and as well by the behavior of the residuals, which exhibit a strong positive "trend" in the range $2 \leq t \leq 4$. This illustrates the inability of the LSQ fit to handle outliers in a satisfactory way.

The right part of Figure 1.5.2 shows the robust Huber fit, with parameter $\beta = 0.025$ (this parameter is chosen to reflect the noise level in the data). The resulting fit is not influenced by the outlier, and the residuals do not seem to exhibit any strong "trend." This is a good illustration of robust regression.

In Section 9.5 we describe numerical algorithms for computing the solutions to robust data fitting problems.

Chapter 2

The Linear Least Squares Problem

This chapter covers some of the basic mathematical facts of the linear least squares problem (LSQ), as well as some important additional statistical results for data fitting. We introduce the two formulations of the least squares problem: the linear system of normal equations and the optimization problem form.

The computation of the LSQ solution via an optimization problem has two aspects: simplification of the problem structure and actual minimization. In this and in the next chapter we present a number of matrix factorizations, for both full-rank and rank-deficient problems, which transform the original problem to one easier to solve. The QR factorization is emphasized, for both the analysis and the solution of the LSQ problem, while in the last section we look into the more expensive complete factorizations.

Some very interesting historical paper on Gaussian elimination that includes also least squares problems can be found in Grcar [109, 110].

2.1 Linear least squares problem formulation

As we saw in the previous chapter, underlying the linear (and possibly weighted) least squares data fitting problem is the linear least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \qquad \text{or} \qquad \min_{\boldsymbol{x}} \|W(\boldsymbol{b} - A\,\boldsymbol{x})\|_2,$$

where $A \in \mathbb{R}^{m\times n}$ is the matrix with samples of the model basis functions, x is the vector of parameters to be determined, the right-hand side $\boldsymbol{b}$ is the vector of observations and W is a diagonal weight matrix (possibly the identity matrix). Since the weights can always be absorbed into A and b

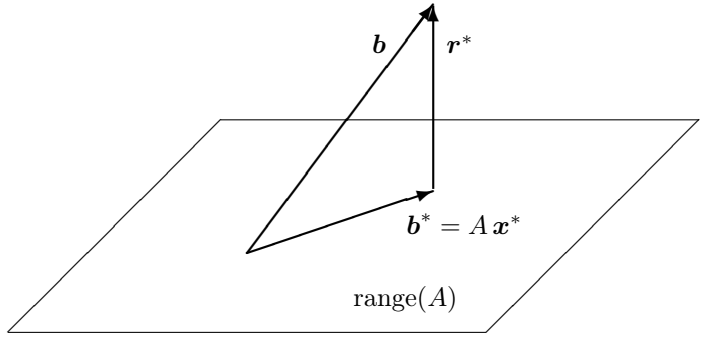


Figure 2.1.1: The geometric interpretation of the linear least squares solution $\boldsymbol{x}^*$. The plane represents the range of A, and if the vector $\boldsymbol{b}$ has a component outside this subspace, then we have an inconsistent system. Moreover, $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$ is the orthogonal projection of $\boldsymbol{b}$ on range(A), and $\boldsymbol{r}^*$ is the LSQ residual vector.

in the mathematical formulation we can, without loss of generality, restrict our discussion to the un-weighted case. Also, from this point on, when discussing the generic linear least squares problem, we will use the notation $\boldsymbol{b}$ for the right-hand side (instead of $\boldsymbol{y}$ that we used in Chapter 1), which is more common in the LSQ literature.

Although most of the material in this chapter is also applicable to the underdetermined case ($m < n$), for notational simplicity we will always consider the overdetermined case $m \geq n$. We denote by r the rank of A, and we consider both full-rank and rank-deficient problems. Thus we always have $r \leq n \leq m$ in this chapter.

It is appropriate to remember at this point that an $m \times n$ matrix A is always a representation of a linear transformation $\boldsymbol{x} \rightarrow A\,\boldsymbol{x}$ with $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and therefore there are two important subspaces associated with it: The *range* or *column space*,

$$\text{range}(A) = \{\boldsymbol{z} \in \mathbb{R}^m \mid \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{z} = A\,\boldsymbol{x}\}\,,$$

and its orthogonal complement, the *null space* of A^T:

$$\text{null}(A^T) = \left\{\boldsymbol{y} \in \mathbb{R}^m \mid A^T\boldsymbol{y} = \boldsymbol{0}\right\}.$$

When A is square and has full rank, then the LSQ problem $\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$ reduces to the linear system of equations $A\,\boldsymbol{x} = \boldsymbol{b}$. In all other cases, due to the data errors, it is highly probable that the problem is *inconsistent*, i.e., $\boldsymbol{b} \notin \text{range}(A)$, and as a consequence there is no exact solution, i.e., no coefficients x_j exist that express $\boldsymbol{b}$ as a linear combination of columns of A.

Instead, we can find the coefficients x_j for a vector $\boldsymbol{b}^*$ in the range of A and "closest" to $\boldsymbol{b}$. As we have seen, for data fitting problems it is natural to use the Euclidean norm as our measure of closeness, resulting in the least squares problem

$$\boxed{\text{Problem LSQ}: \quad \min_x \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2, \quad \text{A}\in \mathbb{R}^{m\times n}, \quad r \le n \le m} \tag{2.1.1}$$

with the corresponding residual vector $\boldsymbol{r}$ given by

$$\boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}. \tag{2.1.2}$$

See Figure 2.1.1 for a geometric interpretation. The minimizer, i.e., the least squares solution – which may not be unique, as it will be seen later – is denoted by $\boldsymbol{x}^*$. We note that the vector $\boldsymbol{b}^* \in \text{range}(A)$ mentioned above is given by $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$.

The LSQ problem can also be looked at from the following point of view. When our data are contaminated by errors, then the data are not in the span of the model basis functions $f_j(t)$ underlying the data fitting problem (cf. Chapter 1). In that case the data vector $\boldsymbol{b}$ cannot – and should not – be precisely "predicted" by the model, i.e., the columns of A. Hence, it must be perturbed by a minimum amount $\boldsymbol{r}$, so that it can then be "represented" by A, in the form of $\boldsymbol{b}^* = A\,\boldsymbol{x}^*$. This approach will establish a viewpoint used in Section 7.3 to introduce the total least squares problem.

As already mentioned, there are good statistical reasons to use the Euclidean norm. The underlying statistical assumption that motivates this norm is that the vector $\boldsymbol{r}$ has random error elements, uncorrelated, with zero mean and a common variance. This is justified by the following theorem.

Theorem 8. *(Gauss-Markov) Consider the problem of fitting a model $M(\boldsymbol{x}, t)$ with the n-parameter vector $\boldsymbol{x}$ to a set of data $b_i = \Gamma(t_i) + e_i$ for $i = 1, \ldots, m$ (see Chapter 1 for details).*

In the case of a linear model $\boldsymbol{b} = A\,\boldsymbol{x}$, if the errors are uncorrelated with mean zero and constant variance ς^2 (not necessarily normally distributed) and assuming that the $m \times n$ matrix A obtained by evaluating the model at the data abscissas $\{t_i\}_{i=1,\ldots,m}$ has full rank n, then the best linear unbiased estimator is the least squares estimator $\boldsymbol{x}^*$, obtained by solving the problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$.

For more details see [20] Theorem 1.1.1. Recall also the discussion on maximum likelihood estimation in Chapter 1. Similarly, for nonlinear models, if the errors e_i for $i = 1, \ldots, m$ have a normal distribution, the unknown parameter vector $\boldsymbol{x}$ estimated from the data using a least squares criterion is the maximum likelihood estimator.

There are also clear mathematical and computational advantages associated with the Euclidean norm: the objective function in (2.1.1) is differentiable, and the resulting gradient system of equations has convenient properties. Since the Euclidean norm is preserved under orthogonal transformations, this gives rise to a range of stable numerical algorithms for the LSQ problem.

Theorem 9. *A necessary and sufficient condition for $\boldsymbol{x}^*$ to be a minimizer of $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ is that it satisfies*

$$A^T(\boldsymbol{b} - A\,\boldsymbol{x}) = \boldsymbol{0}. \tag{2.1.3}$$

Proof. The minimizer of $\rho(\boldsymbol{x}) = \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ must satisfy $\nabla\rho(\boldsymbol{x}) = \boldsymbol{0}$, i.e., $\partial\rho(\boldsymbol{x})/\partial x_k = 0$ for $k = 1, \ldots, n$. The kth partial derivative has the form

$$\begin{aligned} \frac{\partial\rho(\boldsymbol{x})}{\partial x_k} &= \sum_{i=1}^{m} 2\Big(b_i - \sum_{j=1}^{n} x_j a_{ij}\Big)(-a_{ik}) = -2\sum_{i=1}^{m} r_i\, a_{ik} \\ &= -2\,\boldsymbol{r}^T A(:,k) = -2\,A(:,k)^T\boldsymbol{r}, \end{aligned}$$

where $A(:,k)$ denotes the kth column of A. Hence, the gradient can be written as

$$\nabla\rho(\boldsymbol{x}) = -2\,A^T\boldsymbol{r} = -2\,A^T(\boldsymbol{b} - A\,\boldsymbol{x})$$

and the requirement that $\nabla\rho(\boldsymbol{x}) = \boldsymbol{0}$ immediately leads to (2.1.3). □

Definition 10. *The two conditions* (2.1.2) *and* (2.1.3) *can be written as a symmetric $(m+n)\times(m+n)$ system in $\boldsymbol{x}$ and $\boldsymbol{r}$, the so-called augmented system:*

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{r} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix}. \tag{2.1.4}$$

This formulation preserves any special structure that A might have, such as sparsity. Also, it is the formulation used in an iterative refinement procedure for the LSQ solution (discussed in Section 4.5), because of the relevance it gives to the residual.

Theorem 9 leads to the *normal equations* for the solution x^* of the least squares problem:

$$\boxed{\text{Normal equations: } A^T A\,\boldsymbol{x} = A^T\boldsymbol{b}.} \tag{2.1.5}$$

The normal equation matrix A^TA, which is sometimes called the Grammian, is square, symmetric and additionally:

- If $r = n$ (A has full rank), then A^TA is positive definite and the LSQ problem has a unique solution. (Since the Hessian for the least squares problem is equal to $2A^TA$, this establishes the uniqueness of $\boldsymbol{x}^*$.)

- If $r < n$ (A is rank deficient), then $A^T A$ is non-negative definite. In this case, the set of solutions forms a linear manifold of dimension $n - r$ that is a translation of the subspace null(A).

Theorem 9 also states that the residual vector of the LSQ solution lies in null(A^T). Hence, the right-hand side $\boldsymbol{b}$ can be decomposed into two orthogonal components

$$\boldsymbol{b} = A\,\boldsymbol{x} + \boldsymbol{r},$$

with $A\,\boldsymbol{x} \in \text{range}(A)$ and $\boldsymbol{r} \in \text{null}(A^T)$, i.e., $A\,\boldsymbol{x}$ is the orthogonal projection of $\boldsymbol{b}$ onto range(A) (the subspace spanned by the columns of A) and $\boldsymbol{r}$ is orthogonal to range(A).

Example 11. *The normal equations for the NMR problem in Example 1 take the form*

$$\begin{pmatrix} 2.805 & 4.024 & 5.055 \\ 4.024 & 8.156 & 1.521 \\ 5.055 & 1.521 & 50 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13.14 \\ 25.98 \\ 51.87 \end{pmatrix},$$

giving the least squares solution $\boldsymbol{x}^* = (\,1.303\,,\,1.973\,,\,0.305\,)^T$.

Example 12. ***Simplified NMR problem.*** *In the NMR problem, let us assume that we know that the constant background is 0.3, corresponding to fixing* $x_3 = 0.3$. *The resulting* 2×2 *normal equations for* x_1 *and* x_2 *take the form*

$$\begin{pmatrix} 2.805 & 4.024 \\ 4.024 & 8.156 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 10.74 \\ 20.35 \end{pmatrix}$$

and the LSQ solution to this simplified problem is $x_1 = 1.287$ *and* $x_2 = 1.991$. *Figure 2.1.2 illustrates the geometry of the minimization associated with the simplified LSQ problem for the two unknowns* x_1 *and* x_2. *The left plot shows the residual norm surface as a function of* x_1 *and* x_2, *and the right plot shows the elliptic contour curves for this surface; the unique minimum – the LSQ solution – is marked with a dot.*

In the rank-deficient case the LSQ solution is not unique, but one can reduce the solution set by imposing additional constraints. For example, the linear least squares problem often arises from a linearization of a nonlinear least squares problem, and it may be of interest then to impose the additional constraint that the solution has minimal l_2-norm, $\widehat{\boldsymbol{x}} = \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2$, so that the solution stays in the region where the linearization is valid. Because the set of all minimizers is convex there is a unique solution. Another reason for imposing minimal length is stability, as we will see in the section on regularization.

For data approximation problems, where we are free to choose the model basis functions $f_j(t)$, cf. (1.2.2), one should do it in a way that

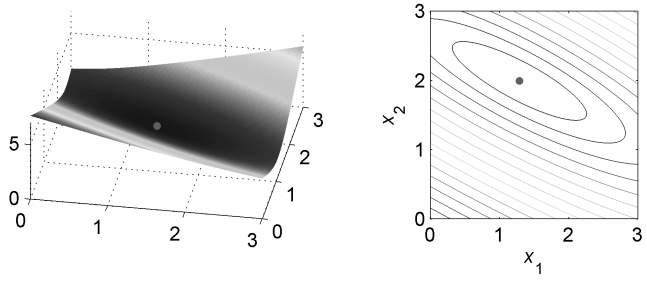


Figure 2.1.2: Illustration of the LSQ problem for the simplified NMR problem. Left: the residual norm as a function of the two unknowns x_1 and x_2. Right: the corresponding contour lines for the residual norm.

gives A full rank. A necessary condition is that the (*continuous*) functions $f_1(t), \ldots, f_n(t)$ are linearly independent, but furthermore, they have to define linearly independent vectors when evaluated on the *specific discrete set of abscissas*. More formally:

A necessary and sufficient condition for the matrix A to have full rank is that the model basis functions be linearly independent over the abscissas $t_1, \ldots, t_m$:

$$\sum_{j=1}^{n} \alpha_j f_j(t_i) = 0 \text{ for } i = 1, \ldots, m \quad \Leftrightarrow \quad \alpha_j = 0 \text{ for } j = 1, ..., n.$$

Example 13. *Consider the linearly independent functions* $f_1(t) = \sin(t)$, $f_2(t) = \sin(2t)$ *and* $f_3(t) = \sin(3t)$*; if we choose the data abscissas* $t_i = \pi/4 + i\pi/2$, $i = 1, \ldots, m$, *the matrix* A *has rank* $r = 2$, *whereas the same functions generate a full-rank matrix* A *when evaluated on the abscissas* $t_i = \pi(i/m)$, $i = 1 \ldots, m - 1$.

An even stronger requirement is that the model basis functions $f_j(t)$ be such that the columns of A are orthogonal.

Example 14. *In general, for data fitting problems, where the model basis functions* $f_j(t)$ *arise from the underlying model, the properties of the matrix* A *are dictated by these functions. In the case of polynomial data fitting, it is possible to choose the functions* $f_j(t)$ *so that the columns of* A *are orthogonal, as described by Forsythe [81], which simplifies the computation of the LSQ solution. The key is to choose a clever representation of the fitting polynomials, different from the standard one with the monomials:*

$f_j(t) = t^{j-1},\ j = 1, ..., n$, such that the sampled polynomials satisfy

$$\sum_{i=1}^{m} f_j(t_i)\, f_k(t_i) = 0 \quad \text{for } j \neq k. \tag{2.1.6}$$

When this is the case we say that the functions are orthogonal over the given abscissas. This is satisfied by the family of orthogonal polynomials defined by the recursion:

$$\begin{aligned} f_1(t) &= 1 \\ f_2(t) &= t - \alpha_1 \\ f_{j+1}(t) &= (t - \alpha_j)\, f_j(t) - \beta_j f_{j-1}(t), \qquad j = 2, \ldots n-1, \end{aligned}$$

where the constants are given by

$$\begin{aligned} \alpha_j &= \frac{1}{s_j^2} \sum_{i=1}^{m} t_i\, f_j(t_i)^2, \qquad j = 0, 1, \ldots, n-1 \\ \beta_j &= \frac{s_j^2}{s_{j-1}^2}, \qquad j = 0, 1, \ldots, n-1 \\ s_j^2 &= \sum_{i=1}^{m} f_j(t_i)^2, \qquad j = 2, \ldots n, \end{aligned}$$

i.e., s_j is the l_2-norm of the jth column of A. These polynomials satisfy (2.1.6), hence, the normal equations matrix A^TA is diagonal, and it follows that the LSQ coefficients are given by

$$x_j^* = \frac{1}{s_j^2} \sum_{i=1}^{m} y_i\, f_j(t_i), \qquad j = 1, \ldots, n.$$

When A has full rank, it follows from the normal equations (2.1.5) that we can write the least squares solution as

$$\boldsymbol{x}^* = (A^TA)^{-1}A^T\boldsymbol{b},$$

which allows us to analyze the solution and the residual vector in statistical terms. Consider the case where the data errors e_i are independent, uncorrelated and have identical standard deviations ς, meaning that the covariance for $\boldsymbol{b}$ is given by

$$\mathrm{Cov}(\boldsymbol{b}) = \varsigma^2\, I_m,$$

since the errors e_i are independent of the exact $\Gamma(t_i)$. Then a standard result in statistics says that the covariance matrix for the LSQ solution is

$$\mathrm{Cov}(\boldsymbol{x}^*) = (A^TA)^{-1}A^T\mathrm{Cov}(\boldsymbol{b})\, A\, (A^TA)^{-1} = \varsigma^2\, (A^TA)^{-1}.$$

We see that the unknown coefficients in the fit – the elements of $\boldsymbol{x}^*$ – are uncorrelated if and only if A^TA is a diagonal matrix, i.e., when the columns of A are orthogonal. This is the case when the model basis functions are orthogonal over the abscissas $t_1, \ldots, t_m$; cf. (2.1.6).

Example 15. ***More data gives better accuracy.*** *Intuitively we expect that if we increase the number of data points then we can compute a more accurate LSQ solution, and the present example confirms this. Specifically we give an asymptotic analysis of how the solution's variance depends on the number m of data points, in the case of linear data fitting. There is no assumption about the distribution of the abscissas t_i except that they belong to the interval $[a, b]$ and appear in increasing order. Now let $h_i = t_i - t_{i-1}$ for $i = 2, \ldots, m$ and let $h = (b-a)/m$ denote the average spacing between the abscissas. Then for $j, k = 1, \ldots, n$ the elements of the normal equation matrix can be approximated as*

$$\begin{aligned}(A^TA)_{jk} &= \sum_{i=1}^{m} h_i^{-1} f_j(t_i) f_k(t_i) h_i \simeq \frac{1}{h} \sum_{i=1}^{m} f_j(t_i) f_k(t_i) h_i \\ &\simeq \frac{m}{b-a} \int_a^b f_j(t) f_k(t)\, dt,\end{aligned}$$

and the accuracy of these approximations increases as m increases. Hence, if F denotes the matrix whose elements are the scaled inner products of the model basis functions,

$$F_{jk} = \frac{1}{b-a} \int_a^b f_j(t) f_k(t) dt, \qquad i, j = 1, \ldots, n,$$

then for large m the normal equation matrix approximately satisfies

$$A^TA \simeq m\, F \qquad \Leftrightarrow \qquad (A^TA)^{-1} \simeq \frac{1}{m}\, F^{-1},$$

where the matrix F is independent of m. Hence, the asymptotic result (as m increases) is that no matter the choice of abscissas and basis functions, as long as A^TA is invertible we have the approximation for the white-noise case:

$$\mathrm{Cov}(\boldsymbol{x}^*) = \varsigma^2 (A^TA)^{-1} \simeq \frac{\varsigma^2}{m} F^{-1}.$$

We see that the solution's variance is (to a good approximation) inversely proportional to the number m of data points.

To illustrate the above result we consider again the frozen cod meat example, this time with two sets of abscissas t_i uniformly distributed in $[0, 0.4]$ for $m = 50$ and $m = 200$, leading to the two matrices $(A^TA)^{-1}$

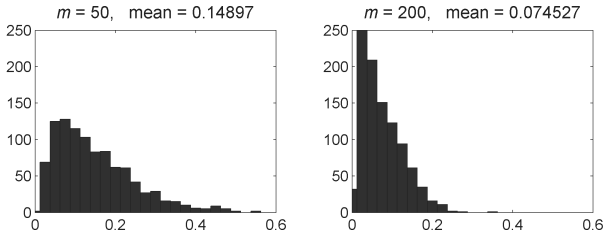


Figure 2.1.3: Histograms of the error norms $\|\boldsymbol{x}^{\text{exact}} - \boldsymbol{x}^*\|_2$ for the two test problems with additive white noise; the errors are clearly reduced by a factor of 2 when we increase m from 50 to 200.

given by

$$\begin{pmatrix} 1.846 & -1.300 & 0.209 \\ -1.300 & 1.200 & -0.234 \\ 0.209 & -0.234 & 0.070 \end{pmatrix} \quad \textit{and} \quad \begin{pmatrix} 0.535 & -0.359 & 0.057 \\ -0.359 & 0.315 & -0.061 \\ 0.057 & -0.061 & 0.018 \end{pmatrix},$$

respectively. The average ratio between the elements in the two matrices is 3.71*, i.e., fairly close to the factor* 4 *we expect from the above analysis when increasing* m *by a factor* 4*.*

We also solved the two LSQ problems for 1000 realizations of additive white noise, and Figure 2.1.3 shows histograms of the error norms $\|\boldsymbol{x}^{\text{exact}} - \boldsymbol{x}^*\|_2$*, where* $\boldsymbol{x}^{\text{exact}} = (1.27,\ 2.04,\ 0.3)^T$ *is the vector of exact parameters for the problem. These results confirm that the errors are reduced by a factor of* 2 *corresponding to the expected reduction of the standard deviation by the same factor.*

2.2 The QR factorization and its role

In this and the next section we discuss the QR factorization and its role in the analysis and solution of the LSQ problem. We start with the simpler case of full-rank matrices in this section and then move on to rank-deficient matrices in the next section.

The first step in the computation of a solution to the least squares problem is the reduction of the problem to an equivalent one with a more convenient matrix structure. This can be done through an explicit factorization, usually based on orthogonal transformations, where instead of solving the original LSQ problem (2.1.1) one solves an equivalent problem with a triangular matrix. The basis of this procedure is the QR factorization, the less expensive decomposition that takes advantage of the isometric properties of orthogonal transformations (proofs for all the theorems in this section can be found in [20], [105] and many other references).

Theorem 16. ***QR factorization.*** *Any real $m \times n$ matrix A can be factored as*

$$\boxed{A = QR \quad with \quad Q \in \mathbb{R}^{m\times m}, \quad R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \in \mathbb{R}^{m\times n},} \tag{2.2.1}$$

where Q is orthogonal (i.e., $Q^TQ = I_m$) and $R_1 \in \mathbb{R}^{n\times n}$ is upper triangular. If A has full rank, then so has R and therefore all its diagonal elements are nonzero.

Theorem 17. ***Economical QR factorization.*** *Let $A \in \mathbb{R}^{m\times n}$ have full column rank $r = n$. The economical (or thin) QR factorization of A is*

$$\boxed{A = Q_1R_1 \quad with \quad Q_1 \in \mathbb{R}^{m\times n}, \quad R_1 \in \mathbb{R}^{n\times n},} \tag{2.2.2}$$

where Q_1 has orthonormal columns (i.e., $Q_1^TQ_1 = I_n$) and the upper triangular matrix R_1 has nonzero diagonal entries. Moreover, Q_1 can be chosen such that the diagonal elements of R_1 are positive, in which case R_1 is the Cholesky factor of A^TA.

Similar theorems hold if the matrix A is complex, with the factor Q now a unitary matrix.

Remark 18. *If we partition the $m \times m$ matrix Q in the full QR factorization* (2.2.1) *as*

$$Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix},$$

then the sub-matrix Q_1 is the one that appears in the economical QR factorization (2.2.2). *The $m \times (m - n)$ matrix Q_2 satisfies $Q_2^TQ_1 = 0$ and $Q_1Q_1^T + Q_2Q_2^T = I_m$.*

Geometrically, the QR factorization corresponds to an orthogonalization of the linearly independent columns of A. The columns of matrix Q_1 are an orthonormal basis for range(A) and those of Q_2 are an orthonormal basis for null(A^T).

The following theorem expresses the least squares solution of the full-rank problem in terms of the economical QR factorization.

Theorem 19. *Let $A \in \mathbb{R}^{m\times n}$ have full column rank $r = n$, with the economical QR factorization $A = Q_1R_1$ from Theorem 17. Considering that*

$$\begin{aligned} \|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 &= \|Q^T(\boldsymbol{b} - A\boldsymbol{x})\|_2^2 = \| \begin{pmatrix} Q_1^T\boldsymbol{b} \\ Q_2^T\boldsymbol{b} \end{pmatrix} - \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \boldsymbol{x}\|_2^2 \\ &= \|Q_1^T\boldsymbol{b} - R_1\boldsymbol{x}\|_2^2 + \|Q_2^T\boldsymbol{b}\|_2^2, \end{aligned}$$

then, the unique solution of the LSQ problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2$ *can be computed from the simpler, equivalent problem*

$$\min_{\boldsymbol{x}} \|Q_1^T\boldsymbol{b} - R_1\boldsymbol{x}\|_2^2,$$

whose solution is

$$\boldsymbol{x}^* = R_1^{-1}Q_1^T\boldsymbol{b} \tag{2.2.3}$$

and the corresponding least squares residual is given by

$$\boldsymbol{r}^* = \boldsymbol{b} - A\,\boldsymbol{x}^* = (I_m - Q_1Q_1^T)\boldsymbol{b} = Q_2Q_2^T\boldsymbol{b}, \tag{2.2.4}$$

with the matrix Q_2 *that was introduced in Remark 18.*

Of course, (2.2.3) is short-hand for solving $R\,x^* = Q_1^T\boldsymbol{b}$, and one point of this reduction is that it is much simpler to solve a triangular system of equations than a full one. Further on we will also see that this approach has better numerical properties, as compared to solving the normal equations introduced in the previous section.

Example 20. *In Example 11 we saw the normal equations for the NMR problem from Example 1; here we take a look at the economical QR factorization for the same problem:*

$$A = \begin{pmatrix} 1.00 & 1.00 & 1 \\ 0.80 & 0.94 & 1 \\ 0.64 & 0.88 & 1 \\ \vdots & \vdots & \vdots \\ 3.2\cdot 10^{-5} & 4.6\cdot 10^{-2} & 1 \\ 2.5\cdot 10^{-5} & 4.4\cdot 10^{-2} & 1 \\ 2.0\cdot 10^{-5} & 4.1\cdot 10^{-2} & 1 \end{pmatrix},$$

$$Q_1 = \begin{pmatrix} 0.597 & -0.281 & 0.172 \\ 0.479 & -0.139 & 0.071 \\ 0.384 & -0.029 & -0.002 \\ \vdots & \vdots & \vdots \\ 1.89\cdot 10^{-5} & 0.030 & 0.224 \\ 1.52\cdot 10^{-5} & 0.028 & 0.226 \\ 1.22\cdot 10^{-5} & 0.026 & 0.229 \end{pmatrix},$$

$$R_1 = \begin{pmatrix} 1.67 & 2.40 & 3.02 \\ 0 & 1.54 & 5.16 \\ 0 & 0 & 3.78 \end{pmatrix}, \quad Q\boldsymbol{b} = \begin{pmatrix} 7.81 \\ 4.32 \\ 1.19 \end{pmatrix}.$$

We note that the upper triangular matrix R_1 *is also the Cholesky factor of the normal equation matrix, i.e.,* $A^TA = R_1^TR_1$.

The QR factorization allows us to study the residual vector in more detail. Consider first the case where we augment A with an additional column, corresponding to adding an additional model basis function in the data fitting problem.

Theorem 21. *Let the augmented matrix* $\overline{A} = (\, A \,,\, \boldsymbol{a}_{n+1} \,)$ *have the QR factorization*

$$\overline{A} = \begin{pmatrix} \overline{Q}_1 & \overline{Q}_2 \end{pmatrix} \begin{pmatrix} \overline{R}_1 \\ 0 \end{pmatrix},$$

with $\overline{Q}_1 = \begin{pmatrix} Q_1 & \bar{\boldsymbol{q}} \end{pmatrix}$, $Q_1^T \bar{\boldsymbol{q}} = \boldsymbol{0}$ *and* $\overline{Q}_2^T \bar{\boldsymbol{q}} = \boldsymbol{0}$. *Then the norms of the least squares residual vectors* $\boldsymbol{r}^* = (I_m - Q_1 Q_1^T)\boldsymbol{b}$ *and* $\bar{\boldsymbol{r}}^* = (I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b}$ *are related by*

$$\|\boldsymbol{r}^*\|_2^2 = \|\bar{\boldsymbol{r}}^*\|_2^2 + (\bar{\boldsymbol{q}}^T \boldsymbol{b})^2.$$

Proof. From the relation $\overline{Q}_1 \overline{Q}_1^T = Q_1 Q_1^T + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T$ it follows that $I_m - Q_1 Q_1^T = I_m - \overline{Q}_1 \overline{Q}_1^T + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T$, and hence,

$$\begin{aligned} \|\boldsymbol{r}^*\|_2^2 &= \|(I_m - Q_1 Q_1^T)\boldsymbol{b}\|_2^2 = \|(I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b} + \bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2^2 \\ &= \|(I_m - \overline{Q}_1 \overline{Q}_1^T)\boldsymbol{b}\|_2^2 + \|\bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2^2 = \|\bar{\boldsymbol{r}}^*\|_2^2 + (\bar{\boldsymbol{q}}^T \boldsymbol{b})^2, \end{aligned}$$

where we used that the two components of $\boldsymbol{r}^*$ are orthogonal and that $\|\bar{\boldsymbol{q}} \bar{\boldsymbol{q}}^T \boldsymbol{b}\|_2 = |\bar{\boldsymbol{q}}^T \boldsymbol{b}| \, \|\bar{\boldsymbol{q}}\|_2 = |\bar{\boldsymbol{q}}^T \boldsymbol{b}|$. □

This theorem shows that, when we increase the number of model basis functions for the fit in such a way that the matrix retains full rank, then the least squares residual norm decreases (or stays fixed if $\boldsymbol{b}$ is orthogonal to $\bar{\boldsymbol{q}}$).

To obtain more insight into the least squares residual we study the influence of the approximation and data errors. According to (1.2.1) we can write the right-hand side as

$$\boldsymbol{b} = \boldsymbol{\Gamma} + \boldsymbol{e},$$

where the two vectors

$$\boldsymbol{\Gamma} = \big(\Gamma(t_1), \ldots, \Gamma(t_m)\big)^T \qquad \text{and} \qquad \boldsymbol{e} = (e_1, \ldots, e_m)^T$$

contain the pure data (the sampled pure-data function) and the data errors, respectively. Hence, the least squares residual vector is

$$\boldsymbol{r}^* = \boldsymbol{\Gamma} - A\,\boldsymbol{x}^* + \boldsymbol{e}, \tag{2.2.5}$$

where the vector $\boldsymbol{\Gamma} - A\,\boldsymbol{x}^*$ is the approximation error. From (2.2.5) it follows that the least squares residual vector can be written as

$$\boldsymbol{r}^* = (I_m - Q_1 Q_1^T)\,\boldsymbol{\Gamma} + (I_m - Q_1 Q_1^T)\,\boldsymbol{e} = Q_2 Q_2^T \boldsymbol{\Gamma} + Q_2 Q_2^T \boldsymbol{e}.$$

We see that the residual vector consists of two terms. The first term $Q_2Q_2^T\mathbf{\Gamma}$ is an "approximation residual," due to the discrepancy between the n model basis functions (represented by the columns of A) and the pure-data function. The second term is the "projected error", i.e., the component of the data errors that lies in the subspace null(A^T). We can summarize the statistical properties of the least squares residual vector as follows.

Theorem 22. *The least squares residual vector* $\boldsymbol{r}^* = \boldsymbol{b} - A\,\boldsymbol{x}^*$ *has the following properties:*

$$\mathcal{E}(\boldsymbol{r}^*) = Q_2Q_2^T\mathbf{\Gamma}, \quad \mathrm{Cov}(\boldsymbol{r}^*) = Q_2Q_2^T\mathrm{Cov}(\boldsymbol{e})Q_2Q_2^T,$$

$$\mathcal{E}(\|\boldsymbol{r}^*\|_2^2) = \|Q_2^T\mathbf{\Gamma}\|_2^2 + \mathcal{E}(\|Q_2^T\boldsymbol{e}\|_2^2).$$

If $\boldsymbol{e}$ *is white noise, i.e.,* $\mathrm{Cov}(\boldsymbol{e}) = \varsigma^2 I_m$*, then*

$$\mathrm{Cov}(\boldsymbol{r}^*) = \varsigma^2 Q_2\,Q_2^T, \quad E(\|\boldsymbol{r}^*\|_2^2) = \|Q_2^T\mathbf{\Gamma}\|_2^2 + (m-n)\,\varsigma^2.$$

Proof. It follows immediately that

$$\mathcal{E}(Q_2Q_2^T\boldsymbol{e}) = Q_2Q_2^T\mathcal{E}(\boldsymbol{e}) = 0 \;\text{ and }\; E(\mathbf{\Gamma}^TQ_2Q_2^T\boldsymbol{e}) = 0,$$

as well as

$$\mathrm{Cov}(\boldsymbol{r}^*) = Q_2Q_2^T\mathrm{Cov}(\mathbf{\Gamma}+\boldsymbol{e})Q_2Q_2^T \;\text{ and }\; \mathrm{Cov}(\mathbf{\Gamma}+\boldsymbol{e}) = \mathrm{Cov}(\boldsymbol{e}).$$

Moreover,

$$\mathcal{E}(\|\boldsymbol{r}^*\|_2^2) = \mathcal{E}(\|Q_2Q_2^T\mathbf{\Gamma}\|_2^2) + \mathcal{E}(\|Q_2Q_2^T\boldsymbol{e}\|_2^2) + \mathcal{E}(2\mathbf{\Gamma}^TQ_2Q_2^T\boldsymbol{e}).$$

It follows that

$$\mathrm{Cov}(Q_2^T\boldsymbol{e}) = \varsigma^2 I_{m-n} \text{ and } \mathcal{E}(\|Q_2^T\boldsymbol{e}\|_2^2) = \mathrm{trace}(\mathrm{Cov}(Q_2^T\boldsymbol{e})) = (m-n)\varsigma^2.$$

□

From the above theorem we see that if the approximation error $\mathbf{\Gamma} - A\,\boldsymbol{x}^*$ is somewhat smaller than the data error $\boldsymbol{e}$ then, in the case of white noise, the scaled residual norm s^* (sometimes referred to as the standard error), defined by

$$s^* = \frac{\|\boldsymbol{r}^*\|_2}{\sqrt{m-n}}, \tag{2.2.6}$$

provides an estimate for the standard deviation ς of the errors in the data. Moreover, provided that the approximation error decreases sufficiently fast when the fitting order n increases, then we should expect that for large

enough n the least squares residual norm becomes dominated by the projected error term, i.e.,

$$\boldsymbol{r}^* \simeq Q_2 Q_2^T \boldsymbol{e} \qquad \text{for } n \text{ sufficiently large.}$$

Hence, if we monitor the scaled residual norm $s^* = s^*(n)$ as a function of n, then we expect to see that $s^*(n)$ initially decreases – when it is dominated by the approximation error – while at a later stage it levels off, when the projected data error dominates. The transition between the two stages of the behavior of $s^*(n)$ indicates a good choice for the fitting order n.

Example 23. *We return to the air pollution example from Example 2. We compute the polynomial fit for* $n = 1, 2, \ldots, 19$ *and the trigonometric fit for* $n = 1, 3, 5, \ldots, 19$ *(only odd values of* n *are used, because we always need a* sin-cos *pair). Figure 2.2.1 shows the residual norm* $\|\boldsymbol{r}^*\|_2$ *and the scaled residual norm* s^* *as functions of* n.

The residual norm decreases monotonically with n*, while the scaled residual norm shows the expected behavior mentioned above, i.e., a decaying phase (when the approximation error dominates), followed by a more flat or slightly increasing phase when the data errors dominate.*

The standard error s^* introduced in (2.2.6) above, defined as the residual norm adjusted by the degrees of freedom in the residual, is just one example of a quantity from statistics that plays a central role in the analysis of LSQ problems. Another quantity arising from statistics is the *coefficient of determination* R^2, which is used in the context of linear regression analysis (statistical modeling) as a measure of how well a linear model fits the data. Given a model $M(\boldsymbol{x}, t)$ that predicts the observations $b_1, b_2, \ldots, b_m$ and the residual vector $\boldsymbol{r} = (b_1 - M(\boldsymbol{x}, t_1), \ldots, b_m - M(\boldsymbol{x}, t_m))^T$, the coefficient of determination is defined by

$$R^2 = 1 - \frac{\|\boldsymbol{r}\|_2^2}{\sum_{i=1}^m (b_i - \bar{b})^2}, \tag{2.2.7}$$

where $\bar{b}$ is the mean of the observations. In general, it is an approximation of the unexplained variance, since the second term compares the variance in the model's errors with the total variance of the data. Yet another useful quantity for analysis is the *adjusted coefficient of determination*, $\operatorname{adj} R^2$, defined in the same way as the coefficient of determination R^2, but adjusted using the residual degrees of freedom,

$$\operatorname{adj} R^2 = 1 - \frac{(s^*)^2}{\sum_{i=1}^m (b_i - \bar{b})^2/(m-1)}, \tag{2.2.8}$$

making it similar in spirit to the squared standard error $(s^*)^2$. In Chapter 11 we demonstrate the use of these statistical tools.

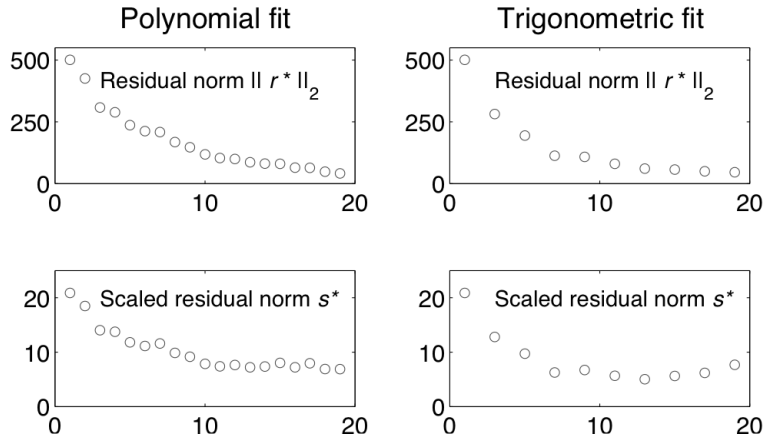


Figure 2.2.1: The residual norm and the scaled residual norm, as functions of the fitting order n, for the polynomial and trigonometric fits to the air pollution data.

2.3 Permuted QR factorization

The previous section covered in detail full-rank problems, and we saw that the QR factorization was well suited for solving such problems. However, for parameter estimation problems – where the model is given – there is no guarantee that A always has full rank, and therefore we must also consider the rank-deficient case. We give first an overview of some matrix factorizations that are useful for detecting and treating rank-deficient problems, although they are of course also applicable in the full-rank case. The minimum-norm solution from Definition 27 below plays a central role in this discussion.

When A is rank deficient we cannot always compute a QR factorization (2.2.1) that has a convenient economical version, where the range of A is spanned by the first columns of Q. The following example illustrates that a column permutation is needed to achieve such a form.

Example 24. *Consider the factorization*

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix}, \qquad \text{for any } c^2 + s^2 = 1.$$

This QR factorization has the required form, i.e., the first factor is orthogonal and the second is upper triangular – but range(A) *is not spanned by the first column of the orthogonal factor. However, a permutation* Π *of the*

columns of A gives a QR factorization of the desired form,

$$A\,\Pi = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = Q\,R = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

with a triangular R and such that the range of A is spanned by the first column of Q.

In general, we need a permutation of columns that selects the linearly independent columns of A and places them first. The following theorem formalizes this idea.

Theorem 25. ***QR factorization with column permutation.*** *If A is real, $m \times n$ with $\operatorname{rank}(A) = r < n \leq m$, then there exists a permutation Π, not necessarily unique, and an orthogonal matrix Q such that*

$$A\,\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{matrix} r \\ m-r \end{matrix}, \tag{2.3.1}$$

where R_{11} is $r \times r$ upper triangular with positive diagonal elements. The range of A is spanned by the first r columns of Q.

Similar results hold for complex matrices where Q now is unitary.

The first r columns of the matrix $A\,\Pi$ are guaranteed to be linearly independent. For a model with basis functions that are not linearly dependent over the abscissas, this provides a method for choosing r linearly independent functions. The rank-deficient least squares problem can now be solved as follows.

Theorem 26. *Let A be a rank-deficient $m \times n$ matrix with the pivoted QR factorization in Theorem 25. Then the LSQ problem* (2.1.1) *takes the form*

$$\begin{aligned} \min_x \|Q^T A\,\Pi\,\Pi^T \boldsymbol{x} - Q^T b\|_2^2 \qquad &= \\ \min_y \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} - \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} \right\|_2^2 \qquad &= \\ \min_y \left(\|R_{11}\boldsymbol{y}_1 - R_{12}\boldsymbol{y}_2 - \boldsymbol{d}_1\|_2^2 + \|\boldsymbol{d}_2\|_2^2 \right), \end{aligned}$$

where we have introduced

$$Q^T \boldsymbol{b} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{y} = \Pi^T \boldsymbol{x} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}.$$

The general solution is

$$\boldsymbol{x}^* = \Pi \begin{pmatrix} R_{11}^{-1}\,(\boldsymbol{d}_1 - R_{12}\,\boldsymbol{y}_2) \\ \boldsymbol{y}_2 \end{pmatrix}, \qquad \boldsymbol{y}_2 = \text{arbitrary} \tag{2.3.2}$$

and any choice of $\boldsymbol{y}_2$ leads to a least squares solution with residual norm $\|\boldsymbol{r}^*\|_2 = \|\boldsymbol{d}_2\|_2$.

Definition 27. *Given the LSQ problem with a rank deficient matrix A and the general solution given by (2.3.2), we define $\widehat{\boldsymbol{x}}^*$ as the solution of minimal l_2-norm that satisfies*

$$\widehat{\boldsymbol{x}}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 \qquad \textit{subject to} \qquad \|\boldsymbol{b} - A\boldsymbol{x}\|_2 = \min .$$

The choice $\boldsymbol{y}_2 = \boldsymbol{0}$ in (2.3.2) is an important special case that leads to the so-called *basic solution*,

$$\boldsymbol{x}_{\mathrm{B}} = \Pi \begin{pmatrix} R_{11}^{-1} Q_1^T \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix},$$

with at least $n-r$ zero components. This corresponds to using only the first r columns of $A\,\Pi$ in the solution, while setting the remaining elements to zero. As already mentioned, this is an important choice in data fitting – as well as other applications – because it implies that $\boldsymbol{b}$ is represented by the smallest subset of r columns of A, i.e., it is fitted with as few variables as possible. It is also related to the new field of compressed sensing [4, 35, 235].

Example 28. ***Linear prediction.*** *We consider a digital signal, i.e., a vector $\boldsymbol{s} \in \mathbb{R}^N$, and we seek a relation between neighboring elements of the form*

$$s_i = \sum_{j=1}^{\ell} x_i \, s_{i-j}, \qquad i = p+1, \ldots, N, \tag{2.3.3}$$

for some (small) value of ℓ. The technique of estimating the ith element from a number of previous elements is called linear prediction (LP), and the LP coefficients x_i can be used to characterize various underlying properties of the signal. Throughout this book we will use a test problem where the elements of the noise-free signal are given by

$$s_i = \alpha_1 \sin(\omega_1 t_i) + \alpha_2 \sin(\omega_2 t_i) + \cdots + \alpha_p \sin(\omega_p t_i), \quad i = 1, 2, \ldots, N.$$

In this particular example, we use $N = 32$, $p = 2$, $\alpha_1 = 2$, $\alpha_2 = -1$ and no noise.

There are many ways to estimate the LP coefficients in (2.3.3)*. One of the popular methods amounts to forming a Toeplitz matrix A (i.e., a matrix with constant diagonals) and a right-hand side $\boldsymbol{b}$ from the signal, with elements given by*

$$a_{ij} = s_{n+i-j}, \quad b_i = s_{n+i}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n,$$

where the matrix dimensions m and n satisfy $m+n = N$ and $\min(m, n) \geq \ell$. We choose $N = 32$ and $n = 7$ giving $m = 25$, and the first 7 rows of A and

the first 7 elements of $\boldsymbol{b}$ *are*

$$\begin{pmatrix} 1.011 & -1.151 & -0.918 & -2.099 & -0.029 & 2.770 & 0.875 \\ 2.928 & 1.011 & -1.151 & -0.918 & -2.099 & -0.029 & 2.770 \\ 1.056 & 2.928 & 1.011 & -1.151 & -0.918 & -2.099 & -0.029 \\ -1.079 & 1.056 & 2.928 & 1.011 & -1.151 & -0.918 & -2.099 \\ -1.197 & -1.079 & 1.056 & 2.928 & 1.011 & -1.151 & -0.918 \\ -2.027 & -1.197 & -1.079 & 1.056 & 2.928 & 1.011 & -1.151 \\ 1.160 & -2.027 & -1.197 & -1.079 & 1.056 & 2.928 & 1.011 \end{pmatrix}, \quad \begin{pmatrix} 2.928 \\ 0.0101 \\ -1.079 \\ -1.197 \\ -2.027 \\ 1.160 \\ 2.559 \end{pmatrix}.$$

The matrix A *is rank deficient and it turns out that for this problem we can safely compute the ordinary QR factorization without pivoting, corresponding to* $\Pi = I$*. The matrix* R_1 *and the vector* $Q_1^T\boldsymbol{b}$ *are*

$$\begin{pmatrix} 7.970 & 2.427 & -3.392 & -4.781 & -5.273 & 1.890 & 7.510 \\ 0 & 7.678 & 3.725 & -1.700 & -3.289 & -6.482 & -0.542 \\ 0 & 0 & 6.041 & 2.360 & -4.530 & -1.136 & -2.765 \\ 0 & 0 & 0 & 5.836 & -0.563 & -4.195 & 0.252 \\ 0 & 0 & 0 & 0 & \epsilon & \epsilon & \epsilon \\ 0 & 0 & 0 & 0 & 0 & \epsilon & \epsilon \\ 0 & 0 & 0 & 0 & 0 & 0 & \epsilon \end{pmatrix}, \quad \begin{pmatrix} 2.573 \\ -4.250 \\ -1.942 \\ -5.836 \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix},$$

where ϵ *denotes an element whose absolute value is of the order* 10^{-14} *or smaller. We see that the numerical rank of* A *is* $r = 4$ *and that* $\boldsymbol{b}$ *is the weighted sum of columns 1 through 4 of the matrix* A*, i.e., four LP coefficients are needed in* (2.3.3)*. A basic solution is obtained by setting the last three elements of the solution to zero:*

$$\boldsymbol{x}_{\mathrm{B}} = (\,-0.096\,,\,-0.728\,,\,-0.096\,,\,-1.000\,,\,0\,,\,0\,,\,0\,)^T.$$

A numerically safer approach for rank-deficient problems is to use the QR factorization with column permutations from Theorem 25, for which we get

$$\Pi = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$R_{11} = \begin{pmatrix} 8.052 & 1.747 & -3.062 & -4.725 \\ 0 & 7.833 & -4.076 & -2.194 \\ 0 & 0 & 5.972 & -3.434 \\ 0 & 0 & 0 & 5.675 \end{pmatrix}, \quad \boldsymbol{d}_1 = \begin{pmatrix} -3.277 \\ 3.990 \\ -5.952 \\ 6.79 \end{pmatrix}.$$

The basic solution corresponding to this factorization is

$$\boldsymbol{x}_{\mathrm{B}} = (\,0\,,\,-0.721\,,\,0\,,\,-0.990\,,\,0.115\,,\,0\,,\,0.027\,)^T.$$

This basic solution expresses $\boldsymbol{b}$ *as a weighted sum of columns 1, 3, 4 and 6 of* A*. The example shows that the basic solution is not unique – both basic solutions given above solve the LSQ problem associated with the linear prediction problem.*

The basic solution introduced above is one way of defining a particular type of solution of the rank-deficient LSQ problem, and it is useful in some applications. However, in other applications we require the minimum-norm solution $\widehat{\boldsymbol{x}}^*$ from Definition 27, whose computation reduces to solving the least squares problem

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_2 = \min_{\boldsymbol{y}_2} \left\| \Pi \begin{pmatrix} R_{11}^{-1}(\boldsymbol{d}_1 - R_{12}\boldsymbol{y}_2) \\ \boldsymbol{y}_2 \end{pmatrix} \right\|_2 .$$

Using the basic solution $\boldsymbol{x}_{\mathrm{B}}$ the problem is reduced to

$$\min_{\boldsymbol{y}_2} \left\| \boldsymbol{x}_{\mathrm{B}} - \Pi \begin{pmatrix} R_{11}^{-1}R_{12} \\ I \end{pmatrix} \boldsymbol{y}_2 \right\|_2 .$$

This is a full-rank least squares problem with matrix $\Pi \begin{pmatrix} R_{11}^{-1}R_{12} \\ I \end{pmatrix}$. The right-hand side $\boldsymbol{x}_{\mathrm{B}}$ and the solution $\boldsymbol{y}_2^*$ can be obtained via a QR factorization. The following results from [101] relates the norms of the basic and minimum-norm solutions:

$$1 \le \frac{\|\boldsymbol{x}_{\mathrm{B}}\|_2}{\|\widehat{\boldsymbol{x}}^*\|_2} \le \sqrt{1 + \|R_{11}^{-1}R_{12}\|_2^2}.$$

Complete orthogonal factorization

As demonstrated above, we cannot immediately compute the minimum-norm least squares solution $\widehat{\boldsymbol{x}}^*$ from the pivoted QR factorization. However, the QR factorization with column permutations can be considered as a first step toward the so-called *complete orthogonal factorization.* This is a decomposition that, through basis changes by means of orthogonal transformations in both $\mathbb{R}^m$ and $\mathbb{R}^n$, concentrates the whole information of A into a leading square nonsingular matrix of size $r \times r$. This gives a more direct way of computing $\widehat{\boldsymbol{x}}^*$. The existence of complete orthogonal factorizations is stated in the following theorem.

Theorem 29. *Let* A *be a real* $m \times n$ *matrix of rank* r*. Then there is an* $m \times m$ *orthogonal matrix* U *and an* $n \times n$ *orthogonal matrix* V *such that*

$$A = URV^T \quad \textit{with} \quad R = \begin{pmatrix} R_{11} & 0 \\ 0 & 0 \end{pmatrix}, \tag{2.3.4}$$

where R_{11} is an $r \times r$ nonsingular triangular matrix.

A similar result holds for complex matrices with U and V unitary. The LSQ solution can now be obtained as stated in the following theorem.

Theorem 30. *Let A have the complete orthogonal decomposition* (2.3.4) *and introduce the auxiliary vectors*

$$U^T\boldsymbol{b} = \boldsymbol{g} = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \end{pmatrix} \begin{matrix} r \\ m-r \end{matrix}, \quad V^T\boldsymbol{x} = \boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} \begin{matrix} r \\ n-r \end{matrix}. \tag{2.3.5}$$

Then the solutions to $\min_x \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ are given by

$$\boldsymbol{x}^* = V \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}, \qquad \boldsymbol{y}_1 = R_{11}^{-1}\boldsymbol{g}_1, \qquad \boldsymbol{y}_2 = \textit{arbitrary}, \tag{2.3.6}$$

and the residual norm is $\|\boldsymbol{r}^*\|_2 = \|\boldsymbol{g}_2\|_2$. In particular, the minimum-norm solution $\widehat{\boldsymbol{x}}^*$ is obtained by setting $\boldsymbol{y}_2 = \boldsymbol{0}$.

Proof. Replacing A by its complete orthogonal decomposition we get

$$\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2 = \|\boldsymbol{b} - URV^T\boldsymbol{x}\|_2^2 = \|U^T\boldsymbol{b} - RV^T\boldsymbol{x}\|_2^2 = \|\boldsymbol{g}_1 - R_{11}\boldsymbol{y}_1\|_2^2 + \|\boldsymbol{g}_2\|_2^2.$$

Since the sub-vector $\boldsymbol{y}_2$ cannot lower this minimum, it can be chosen arbitrarily and the result follows. □

The triangular matrix R_{11} contains all the fundamental information of A. The SVD, which we will introduce shortly, is a special case of a complete orthogonal factorization, which is more computationally demanding and involves an iterative part. The most sparse structure that can be obtained by a finite number of orthogonal transformations, the bidiagonal case, is left to be analyzed exhaustively in the chapter on direct numerical methods.

Example 31. *We return to the linear prediction example from the previous section; this time we compute the complete orthogonal factorization from Theorem 29 and get*

$$R_{11} = \begin{pmatrix} -9.027 & -4.690 & -1.193 & 5.626 \\ 0 & -8.186 & -3.923 & -0.373 \\ 0 & 0 & 9.749 & 5.391 \\ 0 & 0 & 0 & 9.789 \end{pmatrix}, \quad \boldsymbol{g} = \begin{pmatrix} -1.640 \\ 3.792 \\ -6.704 \\ 0.712 \end{pmatrix},$$

and

$$V = \begin{pmatrix} -0.035 & 0.355 & -0.109 & -0.521 & -0.113 & -0.417 & 0.634 \\ 0.582 & -0.005 & 0.501 & -0.103 & 0.310 & -0.498 & -0.237 \\ 0.078 & 0.548 & 0.044 & 0.534 & 0.507 & 0.172 & 0.347 \\ -0.809 & 0.034 & 0.369 & 0.001 & 0.277 & -0.324 & -0.163 \\ 0 & -0.757 & -0.006 & 0.142 & 0.325 & -0.083 & 0.543 \\ 0 & 0 & -0.774 & 0.037 & 0.375 & -0.408 & -0.305 \\ 0 & 0 & 0 & -0.641 & 0.558 & 0.520 & -0.086 \end{pmatrix}.$$

This factorization is not unique and the zeros in V are due to the particular algorithm from [78] used here. The minimum-norm solution is

$$\begin{aligned}\widehat{\boldsymbol{x}}^* &= V\begin{pmatrix} R_{11}^{-1}\boldsymbol{g}_1 \\ \mathbf{0}\end{pmatrix} \\ &= (\,-0.013\,,\,-0.150\,,\,-0.028\,,\,-0.581\,,\,0.104\,,\,0.566\,,\,-0.047\,)^T.\end{aligned}$$

This solution, as well as the basic solutions from the previous example, all solve the rank-deficient least squares problem.

Example 32. *We will show yet another way to compute the linear prediction coefficients that uses the null space information in the complete orthogonal factorization. In particular, we observe that the last three columns of V span the null space of A. If we extract them to form the matrix V_0 and compute a QR factorization of its transpose, then we get*

$$V_0^T = Q_0 R_0, \qquad R_0^T = \begin{pmatrix} 0.767 & 0 & 0 \\ 0.031 & 0.631 & 0 \\ 0.119 & -0.022 & 0.626 \\ 0.001 & 0.452 & 0.060 \\ 0.446 & 0.001 & 0.456 \\ -0.085 & 0.624 & 0.060 \\ -0.436 & -0.083 & 0.626 \end{pmatrix}.$$

Since $A\,R_0^T = 0$, we can normalize the last column (by dividing it by its maximum norm) to obtain the null vector

$$\boldsymbol{v}_0 = (\,0\,,\,0\,,\,1\,,\,0.096\,,\,0.728\,,\,0.096\,,\,1\,)^T,$$

which is another way to describe the linear dependence between five neighboring columns $\boldsymbol{a}_j$ of A. Specifically, this $\boldsymbol{v}_0$ states that $\boldsymbol{a}_7 = -0.096\boldsymbol{a}_6 - 0.728\boldsymbol{a}_5 - 0.96\boldsymbol{a}_4 - \boldsymbol{a}_3$, and it follows that the LP coefficients are $x_1 = -0.728$, $x_2 = -0.096$, $x_3 = -0.096$ and $x_4 = -1$, which is identical to the results in Example 28.

Chapter 7

Additional Topics in Least Squares

In this chapter we collect some more specialized topics, such as problems with constraints, sensitivity analysis, total least squares and compressed sensing.

7.1 Constrained linear least squares problems

The inclusion of constraints in the linear least squares problem is often a convenient way to incorporate a priori knowledge about the problem. Linear equality constraints are the easiest to deal with, and their solution is an important part of solving problems with inequality constraints. Bound constraints and quadratic constraints for linear least squares are also considered in this chapter.

Least squares with linear constraints

Linear equality constraints (LSE)

The general form of a linear least squares problem with linear equality constraints is as follows: find a vector $\boldsymbol{x} \in \mathbb{R}^n$ that minimizes $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ subject to the constraints $C^T\boldsymbol{x} = \boldsymbol{d}$, where C is a given $n \times p$ matrix with $p \leq n$ and $\boldsymbol{d}$ is a given vector of length p. This results in the LSE problem:

$$\boxed{\text{Problem LSE:} \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \quad \text{subject to} \quad C^T\boldsymbol{x} = \boldsymbol{d}.} \tag{7.1.1}$$

A solution exists if $C^T\boldsymbol{x} = \boldsymbol{d}$ is consistent, which is the case if $\text{rank}(C) = p$. For simplicity, we assume that this is satisfied; Björck ([20], pp. 188) ex-

plains ways to proceed when there is no a priori knowledge about consistency. Furthermore, the minimizing solution will be unique if the augmented matrix

$$A_{\text{aug}} = \begin{pmatrix} C^T \\ A \end{pmatrix} \tag{7.1.2}$$

has full rank n.

The idea behind the different algorithms for the LSE problem is to reduce it to an unconstrained (if possible lower-dimensional) LSQ problem. To use Lagrange multipliers is of course an option, but we will instead describe two more direct methods using orthogonal transformations, each with different advantages.

One option is to reformulate the LSE problem as a weighted LSQ problem, assigning large weights (or penalties) for the constraints, thus enforcing that they are almost satisfied:

$$\min_{\boldsymbol{x}} \left\| \begin{pmatrix} \lambda C^T \\ A \end{pmatrix} \boldsymbol{x} - \begin{pmatrix} \lambda \boldsymbol{d} \\ \boldsymbol{b} \end{pmatrix} \right\|_2, \qquad \lambda \text{ large.} \tag{7.1.3}$$

Using the generalized singular value decomposition (GSVD), it can be proved that if $\boldsymbol{x}(\lambda)$ is the solution to (7.1.3) then $\|\boldsymbol{x}(\lambda) - \boldsymbol{x}\|_2 = \mathcal{O}(\lambda^{-2})$, so that in fact $\boldsymbol{x}(\lambda) \to \boldsymbol{x}$ as $\lambda \to \infty$. The LU factorization algorithm from Section 4.2 is well suited to solve this problem, because p steps of Gaussian elimination will usually produce a well-conditioned L matrix.

Although in principle only a general LSQ solver is needed, there are numerical difficulties because the matrix becomes poorly conditioned for increasing values of λ. However, a strategy described in [150], based on Householder transformations combined with appropriate row and column interchanges has been found to give satisfactory results. In practice, as [20] mentions, if one uses the LSE equations in the form (7.1.3), it is sufficient to apply a QR factorization with column permutations.

To get an idea of the size of the weight λ we refer to an example by van Loan described in [20], pp. 193. One can obtain almost 14 digits accuracy with a weight of $\lambda = 10^7$ using a standard QR factorization with permutations (e.g., MATLAB's function "qr"), if the constraints are placed in the first rows. Inverting the order in the equations, though, gives only 10 digits for the same weight λ and an increase in λ only degrades the computed solution. In addition, Björck [20] mentions a QR decomposition based on self-scaling Givens rotations that can be used without the "risk of overshooting the optimal weights."

Another commonly used algorithm directly eliminates some of the variables by using the constraints. The actual steps to solve problem (7.1.1) are:

1. Compute the QR factorization of C to obtain:

$$C = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \qquad \Longleftrightarrow \qquad C^T = \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T.$$

2. Use the orthogonal matrix to define new variables:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = Q^T \boldsymbol{x} \qquad \Longleftrightarrow \qquad \boldsymbol{x} = Q \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \tag{7.1.4}$$

and also to compute the value of the unknown p-vector $\boldsymbol{u}$ by solving the lower triangular system $R_1^T \boldsymbol{u} = \boldsymbol{d}$.

3. Introduce the new variables into the equation

$$\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 = \left\|\boldsymbol{b} - AQQ^T\boldsymbol{x}\right\|_2 \equiv \left\|\boldsymbol{b} - \widetilde{A} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}\right\|_2,$$

where the matrix $\widetilde{A} = AQ$ is partitioned according to the dimensions of $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix}$: $\widetilde{A} = \begin{pmatrix} \widetilde{A}_1 & \widetilde{A}_2 \end{pmatrix}$, allowing the reduced $n-p$ dimensional LSQ problem to be written as

$$\min_{\boldsymbol{v}} \|(\boldsymbol{b} - \widetilde{A}_1 \boldsymbol{u}) - \widetilde{A}_2 \boldsymbol{v}\|_2. \tag{7.1.5}$$

4. Solve the unconstrained, lower-dimensional LSQ problem in (7.1.5) and obtain the original unknown vector $\boldsymbol{x}$ from (7.1.4).

This approach has the advantage that there are fewer unknowns in each system that need to be solved for and moreover the reformulated LSQ problem is better conditioned since, due to the interlacing property of the singular values: $\operatorname{cond}(\widetilde{A}_2) \leq \operatorname{cond}(\widetilde{A}) = \operatorname{cond}(A)$. The drawback is that sparsity will be destroyed by this process.

If the augmented matrix in (7.1.2) has full rank, one obtains a unique solution $\boldsymbol{x}_C^*$, if not, one has to apply a QR factorization with column permutations, while solving problem (7.1.5) to obtain the minimum-length solution vector.

The method of direct elimination compares favorably with another QR-based procedure, the null space method (see, for example, [20, 150]), both in numerical stability and in operational count.

Example 74. *Here we return to the linear prediction problem from Example 28, where we saw that* 4 *coefficients were sufficient to describe the particular signal used there. Hence, if we use* $n = 4$ *unknown LP coefficients, then we obtain a full-rank problem with a unique solution given by*

$$x^* = (\,-1,\, 0.096,\, -0.728,\, -0.096\,)^T.$$

If we want the prediction scheme in (2.3.3) to preserve the mean of the predicted signal, then we should add a linear constraint to the LSQ problem, forcing the prediction coefficients sum to zero, i.e., $\sum_{i=1}^{4} x_i = 0$. *In the above notation this corresponds to the linear equality constraint*

$$C^T \boldsymbol{x} = \boldsymbol{d}, \qquad C^T = (1,1,1,1), \qquad \boldsymbol{d} = 0.$$

Following the direct elimination process described above, this corresponds to the following steps for the actual problem.

1. *Compute the QR factorization of* C*:*

$$C = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = Q\,R = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

2. *Solve* $R_1^T \boldsymbol{u} = \boldsymbol{d} \Leftrightarrow 2u = 0 \Leftrightarrow u = 0$.

3. *Solve* $\min_{\boldsymbol{v}} \left\| \boldsymbol{b} - \widetilde{A}_2 \boldsymbol{v} \right\|_2$ *with*

$$\widetilde{A}_2 = A \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 \end{pmatrix},$$

   giving $\boldsymbol{v} = (\,-0.149,\, -0.755,\, 0.324\,)^T$.

4. *Compute the constrained solution*

$$\begin{aligned} \boldsymbol{x}_C^* &= Q \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} \\ &= \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & -0.5 \\ -0.5 & 0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} -0.149 \\ -0.755 \\ 0.324 \end{pmatrix} = \begin{pmatrix} -0.290 \\ 0.141 \\ -0.465 \\ 0.614 \end{pmatrix}. \end{aligned}$$

It is easy to verify that the elements of $\boldsymbol{x}_C^*$ *sum to zero. Alternatively we can use the weighting approach in (7.1.3), with the row* λC^T *added on top of the matrix* A*; with* $\lambda = 10^2$, 10^4 *and* 10^8 *we obtain solutions almost identical to the above* $\boldsymbol{x}_C^*$*, with elements that sum to* $-1.72 \cdot 10^{-3}$, $-1.73 \cdot 10^{-7}$ *and* $-1.78 \cdot 10^{-15}$*, respectively.*

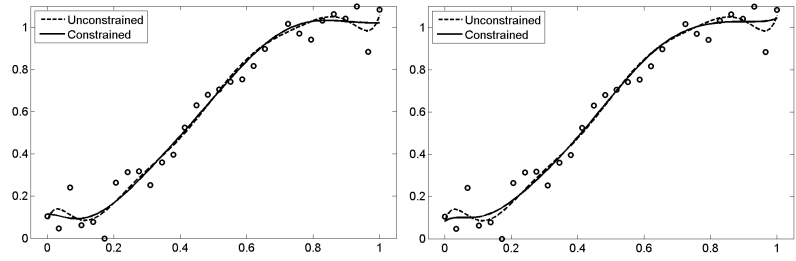


Figure 7.1.1: Constrained least squares polynomial fits ($m = 30$, $n = 10$). The unconstrained fit is shown by the dashed lines, while the constrained fit are shown by the solid lines. Left: equality constraints $M(\boldsymbol{x}, 0.5) = 0.65$ and $M'(\boldsymbol{x}, 0) = M'(\boldsymbol{x}, 1) = 0$. Right: inequality constraints $M'(\boldsymbol{x}, t_i) \geq 0$ for $i = 1, \ldots, m$.

Linear inequality constraints (LSI)

Instead of linear equality constraints we can impose linear inequality constraints on the least squares problem. Then the problem to be solved is

$$\boxed{\text{Problem LSI:} \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2 \quad \text{subject to} \quad \boldsymbol{l} \leq C^T \boldsymbol{x} \leq \boldsymbol{u},} \tag{7.1.6}$$

where the inequalities are understood to be component-wise.

There are several important cases extensively discussed in [20, 150], and Fortran subroutines are available from [88]. A good reference for this part is [91]. A constrained problem may have a minimizer of $\|\boldsymbol{b} - A\,\boldsymbol{x}\|_2$ that is feasible, in which case it can be solved as an unconstrained problem. Otherwise a constrained minimizer will be located on the boundary of the feasible region. At such a solution, one or more constraints will be active, i.e., they will be satisfied with equality.

Thus the solver needs to find which constraints are active at the solution. If those were known a priori, then the problem could be solved as an equality constrained one, using one of the methods we discussed above. If not, then a more elaborate algorithm is necessary to verify the status of the variables and modify its behavior according to which constraints become active or inactive at any given time.

As we showed above, a way to avoid all this complication is to use penalty functions that convert the problem into a sequence of unconstrained problems. After a long hiatus, this approach has become popular again in the form of interior point methods [259]. It is, of course, not devoid of its own complications (principle of conservation of difficulty!).

Example 75. *This example illustrates how equality and inequality constraints can be used to control the properties of the fitting model* $M(\boldsymbol{x}, t)$,

using the rather noisy data ($m = 30$) shown in Figure 7.1.1 and a polynomial of degree 9 (i.e., $n = 10$). In both plots the dashed line shows the unconstrained fit.

Assume that we require that the model $M(\boldsymbol{x}, t)$ must interpolate the point $(t_{\text{int}}, y_{\text{int}}) = (0.5, 0.65)$, have zero derivative at the end points $t = 0$, and $t = 1$, i.e., $M'(\boldsymbol{x}, 0) = M'(\boldsymbol{x}, 1) = 0$. The interpolation requirement correspond to the equality constraint

$$(t_{\text{int}}^9, t_{\text{int}}^8, \ldots, t_{\text{int}}, 1)\, \boldsymbol{x} = y_{\text{int}},$$

while the constraint on $M'(\boldsymbol{x}, t)$ has the form

$$(9t^8, 8t^7, \ldots, 2t, 1, 0)\, \boldsymbol{x} = M'(\boldsymbol{x}, t).$$

Hence the matrix and the right-hand side for the linear constraints in the LSE problem (7.1.1) has the specific form

$$C^T = \begin{pmatrix} t_{\text{int}}^9 & t_{\text{int}}^8 & \cdots & t_{\text{int}}^2 & t_{\text{int}} & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 9 & 8 & \cdots & 2 & 1 & 0 \end{pmatrix}, \qquad \boldsymbol{d} = \begin{pmatrix} y_{\text{int}} \\ 0 \\ 0 \end{pmatrix}.$$

The resulting constrained fit is shown as the solid line in the left plot in Figure 7.1.1.

Now assume instead that we require that the model $M(\boldsymbol{x}, t)$ be monotonically non-decreasing in the data interval, i.e., that $M'(\boldsymbol{x}, t) \geq 0$ for $t \in [0, 1]$. If we impose this requirement at the data points $t_1, \ldots, t_m$, we obtain a matrix C in the LSI problem (7.1.6) of the form

$$C^T = \begin{pmatrix} 9t_1^8 & 8t_1^7 & \ldots & 2t_1 & 0 \\ 9t_2^8 & 8t_2^7 & \ldots & 2t_2 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 9t_m^8 & 8t_m^7 & \ldots & 2t_m & 0 \end{pmatrix},$$

while the two vectors with the bounds are $\boldsymbol{l} = \boldsymbol{0}$ and $\boldsymbol{u} = \boldsymbol{\infty}$. The resulting monotonic fit is shown as the solid line in the right plot in Figure 7.1.1.

Bound constraints

It is worthwhile to discuss the special case of bounded variables, i.e., when $C = I$ in (7.1.6) – also known as box constraints. Starting from a feasible point (i.e., a vector $\boldsymbol{x}$ that satisfies the bounds), we iterate in the usual way for an unconstrained nonlinear problem (see Chapter 9), until a constraint is about to be violated. That identifies a face of the constraint box and a particular variable whose bound is becoming active. We set that variable to the bound value and project the search direction into the corresponding

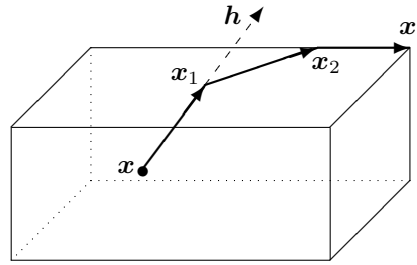


Figure 7.1.2: Gradient projection method for a bound-constrained problem with a solution at the upper right corner. From the starting point $\boldsymbol{x}$ we move along the search direction $\boldsymbol{h}$ until we hit the active constraint in the third coordinate, which brings us to the point $\boldsymbol{x}_1$. In the next step we hit the active constraint in the second coordinate, bringing us to the point $\boldsymbol{x}_2$. The third step brings us to the solution at $\boldsymbol{x}_3$, in which all three coordinates are at active constraints.

face, in order to continue our search for a constrained minimum on it. See Figure 7.1.2 for an illustration. If the method used is gradient oriented, then this technique is called the gradient projection method [210].

Example 76. *The use of bound constraints can sometimes have a profound impact on the LSQ solution, as illustrated in this example where we return to the CT problem from Example 72. Here we add noise* $\boldsymbol{e}$ *with relative noise levels* $\eta = \|\boldsymbol{e}\|_2/\|\boldsymbol{b}\|_2$ *equal to* 10^{-3}, $3 \cdot 10^{-3}$, 10^{-2} *and we enforce non-negativity constraints, i.e.,* $\boldsymbol{l} = \boldsymbol{0}$ *(and* $\boldsymbol{u} = \boldsymbol{\infty}$*). Figure 7.1.3 compares the LSQ solutions (bottom row) with the non-negativity constrained NNLS solutions (top row), and we see that even for the largest noise level* 10^{-2} *the NNLS solution includes recognizable small features – which are lost in the LS solution even for the smaller noise level* $3 \cdot 10^{-3}$.

Sensitivity analysis

Eldén [74] gave a complete sensitivity analysis for problem LSE (7.1.1), including perturbations of all quantities involved in this problem; here we specialize his results to the case where only the right-hand side $\boldsymbol{b}$ is perturbed. Specifically, if $\widetilde{\boldsymbol{x}}_C^*$ denotes the solution with the perturbed right-hand side $\boldsymbol{b} + \boldsymbol{e}$, then Eldén showed that

$$\|\boldsymbol{x}_C^* - \widetilde{\boldsymbol{x}}_C^*\|_2 \leq \|A_C^\dagger\|_2 \, \|\boldsymbol{e}\|_2, \qquad A_C = A\,(I - (C^\dagger)^T C^T).$$

To derive a simpler expression for the matrix A_C consider the QR factorization $C = Q_1 R_1$ introduced above. Then $C^\dagger = R_1^{-1} Q_1^T$ and it follows

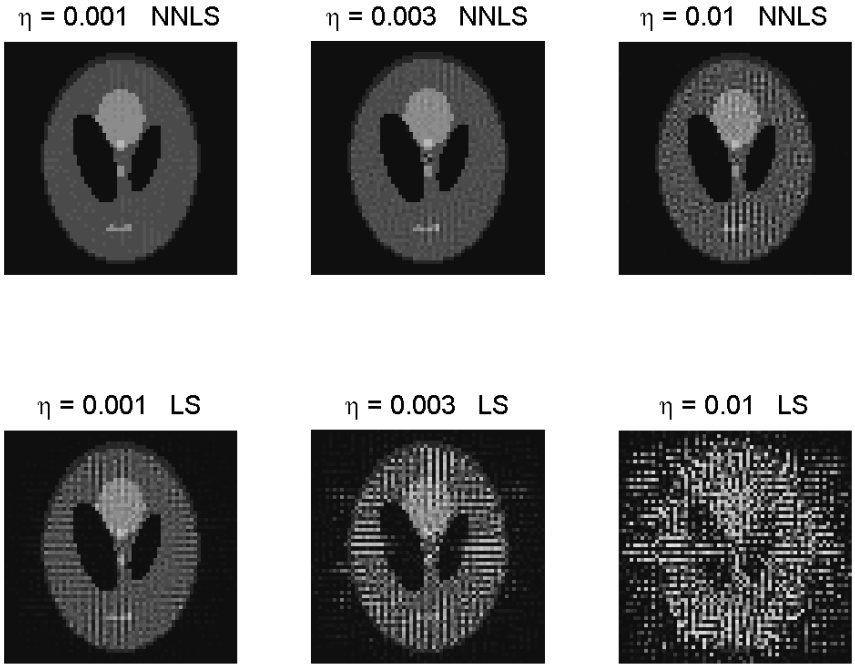


Figure 7.1.3: Reconstructions of the CT problem for three different relative noise levels $\eta = \|\boldsymbol{e}\|_2/\|\boldsymbol{b}\|_2$. Top: LSQ solutions with non-negativity constraints. Bottom: standard LSQ solutions.

that if $Q = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix}$ then $I - (C^\dagger)^T C^T = I - Q_1 Q_1^T = Q_2 Q_2^T$ and hence $A_C = A\,Q_2 Q_2^T = \widetilde{A}_2 Q_2^T$. Moreover, we have $A_C^\dagger = Q_2 \widetilde{A}_2^\dagger$ and thus $\|A_C^\dagger\|_2 = \|\widetilde{A}_2^\dagger\|_2$. The following theorem then follows immediately.

Theorem 77. *Let $\boldsymbol{x}_C^*$ and $\widetilde{\boldsymbol{x}}_C^*$ denote the solutions to problem LSE (7.1.1) with right-hand sides $\boldsymbol{b}$ and $\boldsymbol{b} + \boldsymbol{e}$, respectively. Then, neglecting second-order terms,*

$$\frac{\|\boldsymbol{x}_C^* - \widetilde{\boldsymbol{x}}_C^*\|_2}{\|\boldsymbol{x}_C^*\|_2} \leq \operatorname{cond}(\widetilde{A}_2)\, \frac{\|\boldsymbol{e}\|_2}{\|\widetilde{A}_2\|_2 \|\boldsymbol{x}_C^*\|_2}.$$

This implies that the equality-constrained LS solution is typically less sensitive to perturbations, since the condition number of $\widetilde{A}_2 = AQ_2$ is less than or equal to the condition number of A.

Least squares with quadratic constraints (LSQI)

If we add a quadratic constraint to the least squares problem, then we obtain a problem of the form

$$\boxed{\text{Problem LSQI: } \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_2^2 \text{ subject to } \|\boldsymbol{d} - B\,\boldsymbol{x}\|_2 \leq \alpha,} \quad (7.1.7)$$

where $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times n}$. We assume that

$$\operatorname{rank}(B) = r \qquad \text{and} \qquad \operatorname{rank}\begin{pmatrix} A \\ B \end{pmatrix} = n,$$

which guarantees a unique solution of the LSQI problem. Least squares problems with quadratic constraints arise in many applications, including ridge regression in statistics, Tikhonov regularization of inverse problems, and generalized cross-validation; we refer to [121] for more details.

To facilitate the analysis it is convenient to transform the problem into "diagonal" form by using the generalized singular value decomposition (GSVD) from Section 3.3:

$$U^T A\,X = D_A,\ VB\,X = D_B,\ \boldsymbol{b} = U^T\boldsymbol{b},\ \widetilde{\boldsymbol{d}} = V^T\boldsymbol{d},\ \boldsymbol{x} = X\boldsymbol{y}.$$

The matrices D_A, D_B are diagonal with non-negative elements $\alpha_1, \alpha_2, ..., \alpha_n$ and $\beta_1, \beta_2, ..., \beta_q$, where $q = \min\{p, n\}$ and there are r values $\beta_i > 0$. The reformulated problem is now

$$\min_{\boldsymbol{y}} \left\|\widetilde{\boldsymbol{b}} - D_A\boldsymbol{y}\right\|_2^2 = \min_{\boldsymbol{y}} \left(\sum_{i=1}^{n} \left(\alpha_i y_i - \widetilde{b}_i\right)^2 + \sum_{n+1}^{m} \widetilde{b}_i^2 \right) \quad (7.1.8)$$

$$\text{subject to } \left\|\widetilde{\boldsymbol{d}} - D_B\boldsymbol{y}\right\|_2^2 = \sum_{i=1}^{r} \left(\beta_i y_i - \widetilde{d}_i\right)^2 + \sum_{r+1}^{p} \widetilde{d}_i^2 \leq \alpha^2. \quad (7.1.9)$$

A necessary and sufficient condition for the existence of a solution is that $\sum_{i=r+1}^{p} \widetilde{d}_i^2 \leq \alpha^2$. The way to solve problem (7.1.8)–(7.1.9) will depend on the size of the term $\sum_{i=r+1}^{p} \widetilde{d}_i^2$.

1. If $\sum_{r+1}^{p} \widetilde{d}_i^2 = \alpha^2$, the only $\boldsymbol{y}$ that can satisfy the constraints has as first r elements $y_i = \widetilde{d}_i/\beta_i$, $i = 1, \ldots, r$. The remaining elements y_i are defined from the minimization of (7.1.8). The minimum is attained if $\sum_{i=1}^{n} (\alpha_i y_i - \widetilde{b}_i)^2 = 0$ or is as small as possible, so that for $i = r+1, \ldots, n$ we set $y_i = \widetilde{b}_i/\alpha_i$ if $\alpha_i \neq 0$ and $y_i = 0$ otherwise.

2. If $\sum_{i=r+1}^{p} \widetilde{d}_i^2 < \alpha^2$, one could use the Lagrange multipliers method directly – cf. Appendix C.2.1 – but it is also possible to try a simpler approach to reach a feasible solution: define the vector $\boldsymbol{y}$ that minimizes (7.1.8), which implies choosing $y_i = \widetilde{b}_i/\alpha_i$ if $\alpha_i \neq 0$ as before. If $\alpha_i = 0$, then try to make the left-hand side of the constraints as small as possible by defining $y_i = \widetilde{d}_i/\beta_i$ if $\beta_i \neq 0$ or else $y_i = 0$.

3. If the resulting solution $\boldsymbol{y}$ is feasible, i.e., it satisfies the constraints (7.1.9), then $\boldsymbol{x} = X\,\boldsymbol{y}$ is the LSQI solution.

4. If not, look for a solution on the boundary of the feasible set, i.e., where the constraint is satisfied with equality. That is the standard form for the use of Lagrange multipliers, so the problem is now, for

$$\Phi(\boldsymbol{y}; \mu) = \left\| \widetilde{\boldsymbol{b}} - D_A \boldsymbol{y} \right\|_2^2 - \mu \left(\left\| \boldsymbol{d} - D_B \boldsymbol{y} \right\|_2^2 - \alpha^2 \right) \tag{7.1.10}$$

find $\boldsymbol{y}_\mu$ and μ so that $\nabla \Phi = 0$.

It can be shown that the solution $\boldsymbol{y}_\mu$ in step 4 satisfies the "normal equations"

$$(D_A^T D_A + \mu D_B^T D_B)\, \boldsymbol{y}_\mu = D_A^T\, \widetilde{\boldsymbol{b}} + \mu D_B^T\, \widetilde{\boldsymbol{d}}, \tag{7.1.11}$$

where μ satisfies the *secular equation*:

$$\chi(\mu) = \left\| \widetilde{\boldsymbol{d}} - D_B \boldsymbol{y}_\mu \right\|_2^2 - \alpha^2 = 0.$$

An iterative Newton-based procedure can be defined as follows. Starting from an initial guess μ_0, at each successive step calculate $\boldsymbol{y}_{\mu_i}$ from (7.1.11), then compute a Newton step for the secular equation obtaining a new value μ_{i+1}. It can be shown that this iteration is monotonically convergent to a unique positive root if one starts with an appropriate positive initial value and if instead of $\chi(\mu) = 0$ one uses the more convenient form

$$\frac{1}{\left\| \widetilde{\boldsymbol{d}} - D_B \boldsymbol{y}_\mu \right\|_2^2} - \frac{1}{\alpha^2} = 0.$$

Therefore the procedure can be used to obtain the unique solution of the LSQI problem. This is the most stable, but also the most expensive numerical algorithm. If instead of using the GSVD reformulation one works with the original equations (7.1.7), an analogous Newton-based method can be applied, in which the first stage at each step is

$$\min_{\boldsymbol{x}_\mu} \left\| \begin{pmatrix} A \\ \sqrt{\mu} B \end{pmatrix} \boldsymbol{x}_\mu - \begin{pmatrix} \boldsymbol{b} \\ \sqrt{\mu} \boldsymbol{d} \end{pmatrix} \right\|_2 .$$

Efficient methods for solution of this kind of least squares problem have been studied for several particular cases; see [20] for details.

7.2 Missing data problems

The problem of missing data occurs frequently in scientific research. It may be the case that in some experimental plan, where observations had to be made at regular intervals, occasional omissions arise. Examples would be clinical trials with incomplete case histories, editing of incomplete surveys or, as in the example given at the end of this section, gene expression microarray data, where some values are missing. Let us assume that the missing data are MCAR (missing completely at random), i.e., the probability that a particular data element is missing is unrelated to its value or any of the variables. For example, in the case of data arrays, independent of the column or row.

The appropriate technique for data imputation (a statistical term, meaning the estimation of missing values), will depend among other factors, on the size of the data set, the proportion of missing values and on the type of missing data pattern. If only a few values are missing, say, 1–5%, one could use a single regression substitution: i.e., predict the missing values using linear regression with the available data and assign the predicted value to the missing score. The disadvantage of this approach is that this information is only determined from the – now reduced – available data set. However, with MCAR data, any subsequent statistical analysis remains unbiased. This method can be improved by adding to the predicted value a residual drawn to reflect uncertainty (see [152], Chapter 4).

Other classic processes to fill in the data to obtain a complete set are as follows:

- Listwise deletion: omit the cases with missing values and work with the remaining set. It may lead to a substantial decrease in the available sample size, but the parameter estimates are unbiased.

- Hot deck imputation: replace the missing data with a random value drawn from the collection of data of similar participants. Although

widely used in some applications there is scarce information about the theoretical properties of the method.

- Mean substitution: substitute a mean of the available data for the missing values. The mean may be formed within classes of data. The mean of the resulting set is unchanged, but the variance is underestimated.

More computationally intensive approaches based on least squares and maximum-likelihood principles have been studied extensively in the past decades and a number of software packages that implement the procedures have been developed.

Maximum likelihood estimation

These methods rely on probabilistic modeling, where we wish to find the maximum likelihood (ML) estimate for the parameters of a model, including both the observed and the missing data.

Expectation-maximization (or EM) algorithm

One ML algorithm is the expectation-maximization algorithm. This algorithm estimates the model parameters iteratively, starting from some initial guess of the ML parameters, using, for example, a model for the listwise deleted data. Then follows a recursion until the parameters stabilize, each step containing two processes.

- E-step: the distribution of the missing data is estimated given the observed data and the current estimate of the parameters.

- M-step: the parameters are re-estimated to those with maximum likelihood, assuming the complete data set generated in the E-step.

Once the iteration has converged, the final maximum likelihood estimates of the regression coefficients are used to estimate the final missing data. It has been proved that the method converges, because at each step the likelihood is non-decreasing, until a local maximum is reached, but the convergence may be slow and some acceleration method must be applied. The global maximum can be obtained by starting the iteration several times with randomly chosen initial estimates.

For additional details see [149, 152, 220]. For software IBM SPSS: missing value analysis module. Also free software such as NORM is available from [163].

Multiple imputation (MI)

Instead of filling in a single value for each missing value, Rubin's multiple imputation procedure [152], replaces each missing value with a set of plausible values that represent the uncertainty about the right value to impute. Multiple imputation (MI) is a Monte Carlo simulation process in which a number of full imputed data sets are generated. Statistical analysis is performed on each set, and the results are combined [68] to produce an overall analysis that takes the uncertainty of the imputation into consideration. Depending on the fraction of missing values, a number between 3 and 10 sets must be generated to give good final estimates.

The critical step is the generation of the imputed data set. The choice of the method used for this generation depends on the type of the missing data pattern (see, for example, [132]). For monotone patterns, a parametric regression method can be used, where a regression model is fitted for the variable with missing values with other variables as covariates (there is a hierarchy of missingness: if z_b is observed, then z_a for $a < b$, is also observed). This procedure allows the incorporation of additional information into the model, for example, to use predictors that one knows are related to the missing data. Based on the resulting model, a new regression model is simulated and used to impute the missing values.

For arbitrary missing data patterns, a computationally expensive Markov Chain Monte Carlo (MCMC) method, based on an assumption of multivariate normality, can be used (see [220]). MI is robust to minor departures from the normality assumption, and it gives reasonable results even in the case of a large fraction of missing data or small size of the samples.

For additional information see [132, 149, 152]. For software see [132].

Least squares approximation

We consider here the data as a matrix A with m rows and n columns, which is approximated by a low-rank model matrix. The disadvantage is that no information about the data distribution is included, which may be important when the data belong to a complex distribution. Two types of methods are in common use: SVD based and local least squares imputations.

SVD-based imputation

In this method, the singular value decomposition is used to obtain a set of orthogonal vectors that are linearly combined to estimate the missing data values. As the SVD can only be performed on complete matrices, one works with an auxiliary matrix $A^{'}$ obtained by substituting any missing position in A by a row average. The SVD of $A^{'}$ is as usual $A^{'} = U\Sigma V^T$. We select first the k most significant right singular vectors of $A^{'}$.

Then, one estimates the missing ij value of A by first regressing the row i against the significant right eigenvectors and then using a linear combination of the regression coefficients to compute an estimate for the element ij. (Note that the j components of the right eigenvectors are not used in the regression.) This procedure is repeated until it converges, and an important point is that the convergence depends on the configuration of the missing entries.

Local least squares imputation

We will illustrate the method with the imputation of a DNA microarray. The so-called DNA microarray, or DNA chip, is a technology used to experiment with many genes at once. To this end, single strands of complementary DNA for the genes of interest – which can be many thousands – are immobilized on spots arranged in a grid ("array") on a support that will typically be a glass slide, a quartz wafer or a nylon membrane.

The data from microarray experiments are usually in the form of large matrices of expression levels of genes (gene expression is the process by which information from a gene is used in the synthesis of a functional gene product), under different experimental conditions and frequently with some values missing. Missing values occur for diverse reasons, including insufficient resolution, image corruption or simply due to dust or scratches on the slide. Robust missing value estimation methods are needed, since many algorithms for gene expression analysis require a complete matrix of gene array values.

One method, developed by Kim, Golub and Park [145] for estimating the missing values, is a least squares-based imputation approach using the concept of coherent genes. The method is a local least squares (LLS) algorithm, since only the genes with high similarity with the target gene, the one with incomplete values, are used. The coherent genes are identified using similarity measures based on the ℓ_2-norm or the Pearson correlation coefficient. Once identified, two approaches are used to estimate the missing values, depending on the relative sizes between the number of selected similar genes and the available experiments:

1. Missing values can be estimated either by representing the target gene with missing values as a linear combination of the similar genes.

2. The target experiment that has missing values can be represented as a linear combination of related experiments.

Denote with $G \in \mathbb{R}^{m\times n}$ a gene expression data matrix with m genes and n experiments and assume $m \ggg n$. The row $\boldsymbol{g}_i^T$ of G represents expressions of the ith gene in n experiments. Assume for now that only

one value is missing and it corresponds to the first position of the first gene, $G(1,1) = \boldsymbol{g}_1(1)$, denoted by α for simplicity.

Now, among the genes with a known first position value, either the k nearest-neighbors gene vectors are located using the ℓ_2-norm or the k most similar genes are identified using the Pearson correlation coefficient. Then, based on these k closest gene vectors, a matrix $A \in \mathbb{R}^{k\times(q-1)}$ and two vectors $\boldsymbol{b} \in \mathbb{R}^k$ and $\boldsymbol{w} \in \mathbb{R}^{(q-1)}$ are formed. The k rows of the matrix A consist of the k closest gene vectors with their first values deleted; q varies depending on the number of known values in these similar genes (i.e., the number of experiments recorded successfully). The elements of $\boldsymbol{b}$ consist of the first values of these gene vectors, and the elements of $\boldsymbol{w}$ are the $q-1$ known elements of $\boldsymbol{g}_1$.

When $k < q-1$, the missing value in the target gene is approximated using the same-position value of the nearest genes:

1. Solve the local least squares problem $\min_{\boldsymbol{x}} \left\|A^T\boldsymbol{x} - \boldsymbol{w}\right\|_2$.
2. Estimate the missing value as a linear combination of the first values of the coherent genes $\alpha = \boldsymbol{b}^T\boldsymbol{x}$.

When $k \geq q-1$, on the other hand, the missing value is estimated using the experiments:

1. Solve the local least squares problem $\min_{\boldsymbol{x}} \|A\,\boldsymbol{x} - \boldsymbol{b}\|_2$.
2. Estimate the missing value as a linear combination of values of experiments, not taking into account the first experiment in the gene $\boldsymbol{g}_1$, i.e., $\alpha = \boldsymbol{w}^T\boldsymbol{x}$.

An improvement is to add weights of similarity for the k nearest neighbors, leading to weighted LSQ problems. In the actual DNA microarray data, missing values may occur in several locations. For each missing value the arrays A, $\boldsymbol{b}$ and $\boldsymbol{w}$ are generated and the LLS solved. When building the matrix A for a missing value, the already estimated values of the gene are taken into consideration.

An interesting result, based on experiments with data from the Stanford Microarray Database (SMD), is that the most robust missing value estimation method is the one based on representing a target experiment that has a missing value as a linear combination of the other experiments. A program called LSimpute is available from the authors of [145].

There are no theoretical results comparing different imputation algorithms, but the test results of [145, 238] are consistent and suggest that the above-described method is more robust for noisy data and less sensitive to k, the number of nearest neighbors used, than the SVD method. The appropriate choice of the k closest genes is still a matter of trial and error, although some experiments with random matrices point to thresholds for it [156]. See also [250].

7.3 Total least squares (TLS)

The assumption used until now for the LSQ problem is that errors are restricted to the right-hand side $\boldsymbol{b}$, i.e., the linear model is $A\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}$ where $\boldsymbol{r}$ is the residual. A new problem arises when the data matrix A is also not known exactly, so both A and $\boldsymbol{b}$ have random errors. For simplicity, the statistical hypothesis will be that the rows of the errors are independent and have a uniform distribution with zero mean and common variance (a more general case is treated in [106]). This leads to the total least squares (TLS) problem of calculating a vector $\boldsymbol{x}_{\mathrm{TLS}}$ so that the augmented residual matrix $(\, E \;\; \boldsymbol{r} \,)$ is minimized:

$$\boxed{\text{Problem TLS: } \min \left\| \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix} \right\|_{\mathrm{F}}^2 \text{ subject to } (A+E)\,\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}.} \quad (7.3.1)$$

We note that

$$\left\| \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix} \right\|_F^2 = \|E\|_F^2 + \|\boldsymbol{r}\|_2^2.$$

The relation to ordinary least squares problems is as follows:

- In the least squares approximation, one replaces the inconsistent problem $A\,\boldsymbol{x} = \boldsymbol{b}$ by the consistent system $A\,\boldsymbol{x} = \boldsymbol{b}'$, where $\boldsymbol{b}'$ is the vector closest to $\boldsymbol{b}$ in range(A), obtained by minimizing the orthogonal distance to range(A).
- In the total least squares approximation, one goes further and replaces the inconsistent problem $A\,\boldsymbol{x} = \boldsymbol{b}$ by the consistent system $A'\,\boldsymbol{x} = \boldsymbol{b}'$, finding the closest A' and $\boldsymbol{b}'$ to A and $\boldsymbol{b}$, respectively, by minimizing simultaneously the sum of squared orthogonal distances from the columns $\boldsymbol{a}_i$ to $\boldsymbol{a}_i'$ and $\boldsymbol{b}$ to $\boldsymbol{b}'$.

Example 78. *To illustrate the idea behind TLS we consider the following small problem:*

$$A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The TLS solution $\boldsymbol{x}_{\mathrm{TLS}} = 1.618$ *is obtained with*

$$E = \begin{pmatrix} -0.276 \\ 0.447 \end{pmatrix}, \qquad \boldsymbol{r} = \begin{pmatrix} 0.171 \\ -0.276 \end{pmatrix}.$$

Figure 7.3.1 illustrates the geometrical aspects of this simple problem. We see that the perturbed right-hand side $\boldsymbol{b}+\boldsymbol{r}$ *and the perturbed first (and only) column* $A + E$ *are both orthogonal projections of* $\boldsymbol{b}$ *and* A*, respectively, on the subspace* range$(A + E)$*. The perturbations* $\boldsymbol{r}$ *and* E *are orthogonal to* $\boldsymbol{b}$ *and* A*, respectively, and they are chosen such that their lengths are minimal. Then* $\boldsymbol{x}_{\mathrm{TLS}}$ *is the ratio between the lengths of these two projections.*

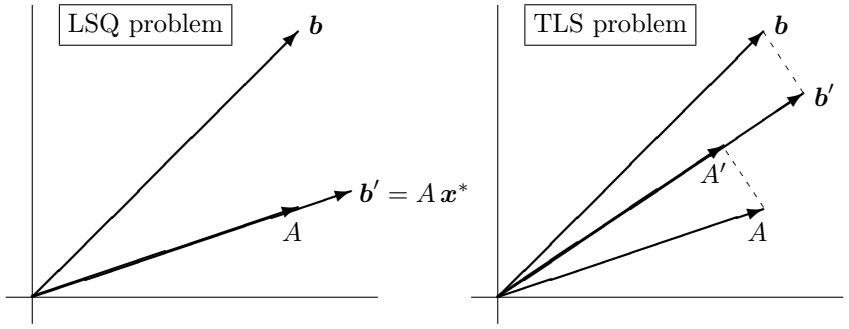


Figure 7.3.1: Illustration of the geometry underlying the LSQ problem (left) and the TLS problem (right). The LSQ solution $\boldsymbol{x}^*$ is chosen such that the residual $\boldsymbol{b}' - \boldsymbol{b}$ (the dashed line) is orthogonal to the vector $\boldsymbol{b}' = A\,\boldsymbol{x}^*$. The TLS solution $\boldsymbol{x}_{\mathrm{TLS}}$ is chosen such that the the residuals $\boldsymbol{b}' - \boldsymbol{b}$ and $A' - A$ (the dashed lines) are orthogonal to $\boldsymbol{b}' = A'\,\boldsymbol{x}_{\mathrm{TLS}}$ and A', respectively.

We will only consider the case when $\mathrm{rank}(A) = n$ and $\boldsymbol{b} \notin \mathrm{range}(A)$. The trivial case when $\boldsymbol{b} \in \mathrm{range}(A)$ means that the system is consistent, $(E\ \boldsymbol{r}) = 0$ and the TLS solution is identical to the LSQ solution.

We note that the rank-deficient matrix case, $\mathrm{rank}(A) < n$, has in principle only a solution in the trivial case $\boldsymbol{b} \in \mathrm{range}(A)$. In the general case it is treated by reducing the TLS problem to a smaller, full-rank problem using column subset selection techniques. More details are given in ([239], section 3.4). The work of Paige and Strakoš [180] on core problems is also relevant here. The following example taken from [105], p. 595, illustrates this case.

Example 79. *Consider the problem defined by*

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

The rank of A is 1 and the matrix is rank deficient, with $\boldsymbol{b} \notin \mathrm{range}(A)$, so there is no solution of the problem as is. Note that

$$E_\varepsilon = \begin{pmatrix} 0 & 0 \\ 0 & \varepsilon \\ 0 & \varepsilon \end{pmatrix}$$

is a perturbation, such that for any $\varepsilon \neq 0$ we have $\boldsymbol{b} \in \mathrm{range}(A + E_\varepsilon)$, but there is no smallest $\|E_\varepsilon\|_{\mathrm{F}}$.

The TLS problem and the singular value decomposition

Let us rewrite the TLS problem as a system:

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} + \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} = \mathbf{0},$$

or equivalently

$$C\,\boldsymbol{z} + F\,\boldsymbol{z} = \mathbf{0} \qquad \text{with} \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

where $C = \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ and $F = \begin{pmatrix} E & \boldsymbol{r} \end{pmatrix}$ are matrices of size $m \times (n+1)$. We seek a solution $\boldsymbol{z}$ to the homogeneous equation

$$(C + F)\,\boldsymbol{z} = \mathbf{0} \tag{7.3.2}$$

that minimizes $\|F\|_{\mathrm{F}}^2$. For the TLS problem to have a non-trivial solution $\boldsymbol{z}$, the matrix $C + F$ must be singular, i.e., $\operatorname{rank}(C + F) < n + 1$. To attain this, the SVD

$$C = \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = U \Sigma V^T = \sum_{i=1}^{n+1} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

can be used to determine an acceptable perturbation matrix F. The singular values of A, here denoted by $\sigma_1' \geq \sigma_2' \geq \ldots \geq \sigma_n'$, will also enter in the discussion.

The solution technique is easily understood in the case when the singular values of C satisfy $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > \sigma_{n+1}$, i.e., when the smallest singular value is isolated. Since we are considering the full-rank, non-trivial case, we also have $\sigma_{n+1} > 0$.

From the Eckart-Young-Mirski Theorem 43, the matrix nearest to C with a rank lower than $n + 1$ is at distance σ_{n+1} from C, and it is given by $\sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$. Thus, selecting $C + F = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ implies choosing a perturbation $F = -\sigma_{n+1} \boldsymbol{u}_{n+1} \boldsymbol{v}_{n+1}^T$ with minimal norm: $\|F\|_{\mathrm{F}} = \sigma_{n+1}$.

The solution $\boldsymbol{z}$ is now constructed using the fact that the $\boldsymbol{v}_i$ are orthonormal, and therefore $(C + F)\,\boldsymbol{v}_{n+1} = \mathbf{0}$. Thus, a general solution of the TLS problem is obtained by scaling the right singular vector $\boldsymbol{v}_{n+1}$ corresponding to the smallest singular value, in order to enforce the condition that $z_{n+1} = -1$:

$$z_i = \frac{v_{i,n+1}}{-v_{n+1,n+1}}, \qquad i = 1, 2, \ldots, n+1. \tag{7.3.3}$$

This is possible *provided that* $v_{n+1,n+1} \neq 0$. If $v_{n+1,n+1} = 0$, a solution does not exist and the problem is called *nongeneric*.

A theorem proved in [106] gives sufficient conditions for the existence of a unique solution. If the smallest singular value σ'_n of the full-rank matrix A is strictly larger than the smallest singular value of $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = C$, σ_{n+1}, then $v_{n+1,n+1} \neq 0$ and a unique solution exists.

Theorem 80. *Denote by $\sigma_1, \ldots, \sigma_{n+1}$ the singular values of the augmented matrix $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ and by $\sigma'_1, \ldots, \sigma'_n$ the singular values of the matrix A. If $\sigma'_n > \sigma_{n+1}$, then there is a TLS correction matrix $\begin{pmatrix} E & \boldsymbol{r} \end{pmatrix} = -\sigma_{n+1} u_{n+1} v_{n+1}^T$ and a unique solution vector*

$$\boxed{\boldsymbol{x}_{\mathrm{TLS}} = -(v_{1,n+1}, \ldots, v_{n,n+1})^T / v_{n+1,n+1}} \tag{7.3.4}$$

that solves the TLS problem.

Moreover, there are closed-form expressions for the solution:

$$\boldsymbol{x}_{\mathrm{TLS}} = (A^T A - \sigma_{n+1}^2 I)^{-1} A^T \boldsymbol{b}$$

and the residual norm:

$$\|A\,\boldsymbol{x}_{\mathrm{TLS}} - \boldsymbol{b}\|_2^2 = \sigma_{n+1}^2 \left(1 + \sum_{i=1}^{n} \frac{\left(\boldsymbol{u}_i^T \boldsymbol{b}\right)^2}{(\sigma'_i)^2 - \sigma_{n+1}^2} \right).$$

The following example, taken from pp. 179 in [20], illustrates the difficulty associated with the nongeneric case in terms of the SVDs.

Example 81. *Consider the augmented matrix:*

$$\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \qquad \text{where} \qquad A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The smallest singular value of A is $\sigma'_1 = 1$, and for the augmented matrix is $\sigma_2 = 1$. So here $\sigma'_n = \sigma_{n+1}$, and $v_{n+1,n+1} = 0$, and therefore no solution exists. The formal algorithm would choose the perturbation matrix from the dyadic form for $A + E = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$; then

$$\begin{pmatrix} A + E & \boldsymbol{b} + \boldsymbol{r} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$$

and it is easy to see that $\boldsymbol{b} + \boldsymbol{r} \notin \mathrm{range}(A + E)$, and there is no solution.

There are further complications with the nongeneric case, but we shall not go into these issues here primarily because they seem to have been resolved by the recent work of Paige and Strakoš [180].

In [239] p. 86, conditions for the existence and uniqueness of solutions are given, as well as closed form expressions for these solutions. In [239] p. 87, a general algorithm for the TLS problem is described. It solves

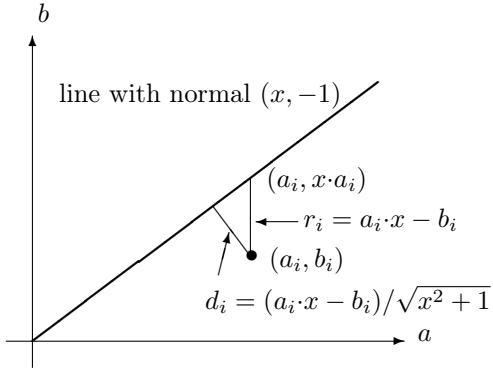


Figure 7.3.2: Illustration of LSQ and TLS for the case $n = 1$. The ith data point is (a_i, b_i) and the point vertically above it on the line is $(a_i, a_i{\cdot}x)$, hence the vertical distance is $r_i = a_i{\cdot}x - b_i$. The orthogonal distance to the line is $d_i = (x{\cdot}a_i - b_i)/\sqrt{x^2+1}$.

any generic and nongeneric TLS problem, including the case with multiple right-hand sides. The software developed by van Huffel is available from Netlib [263]. Subroutine PTLS solves the TLS problem by using the partial singular value decomposition (PSVD) mentioned in Section 5.6, thereby improving considerably the computational efficiency with respect to the classical TLS algorithm. A large-scale algorithm based on Rayleigh quotient iteration is described in [25]. See [155] for a recent survey of TLS methods.

Geometric interpretation

It is possible to give a geometric interpretation of the difference between the fits using the least squares and total least squares methods. Define the rows of the matrix $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ as the m points $\boldsymbol{c}_i = (a_{i1}, \ldots, a_{in}, b_i)^T \in \mathbb{R}^{n+1}$, to which we try to fit the linear subspace $\mathcal{S}_{\boldsymbol{x}}$ of dimension n that is orthogonal to the vector $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix}$:

$$\mathcal{S}_{\boldsymbol{x}} = \operatorname{span}\left\{ \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\}^{\perp} = \left\{ \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} \mid \boldsymbol{a} \in \mathbb{R}^n,\ \boldsymbol{b} \in \mathbb{R},\ \boldsymbol{b} = \boldsymbol{x}^T\boldsymbol{a} \right\}.$$

In LSQ, one minimizes

$$\left\| \begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2 = \sum_{k=1}^{m} \left(\boldsymbol{c}_k^T \boldsymbol{z} \right)^2 \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

which measures the sum of squared distances along the $n+1$-coordinate axis from the points $\boldsymbol{c}_k$ to the subspace $\mathcal{S}_{\boldsymbol{x}}$ (already known as the residuals r_i). For the case $n = 1$, where $A = (a_1, a_2, \ldots a_m)^T$ and $\boldsymbol{x} = x$ (a single unknown), the LSQ approximation minimizes the *vertical* distance of the points (a_i, b_i) to the line through the origin with slope x (whose normal vector is $(x, -1)^T$); see Figure 7.3.2.

The TLS approach can be formulated as an equivalent problem.

From the SVD of $(\; A \quad \boldsymbol{b} \;)$ and the definition of matrix 2-norm we have

$$\left\| (\; A \quad \boldsymbol{b} \;) \, \boldsymbol{z} \right\|_2 / \left\| \boldsymbol{z} \right\|_2 \geq \sigma_{n+1},$$

for any nonzero $\boldsymbol{z}$. If σ_{n+1} is isolated and the vector $\boldsymbol{z}$ has unit norm, then $\boldsymbol{z} = \pm \boldsymbol{v}_{n+1}$ and the inequality becomes an equality $\left\| (\; A \quad \boldsymbol{b} \;) \, \boldsymbol{z} \right\|_2^2 = \sigma_{n+1}^2$. So, in fact the TLS problem consists of finding a vector $\boldsymbol{x} \in \mathbb{R}^n$ such that

$$\frac{\left\| (\; A \quad \boldsymbol{b} \;) \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2}{\left\| \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} \right\|_2^2} = \frac{\| A \, \boldsymbol{x} - \boldsymbol{b} \|_2^2}{\| \boldsymbol{x} \|_2^2 + 1} = \sigma_{n+1}^2. \tag{7.3.5}$$

The left-hand quantity is $\sum_{i=1}^{m} \frac{(\boldsymbol{a}_i^T \boldsymbol{x} - b_i)^2}{\boldsymbol{x}^T \boldsymbol{x} + 1}$, where the ith term is the square of the *true* Euclidean (or orthogonal) distance from $\boldsymbol{c}_i$ to the nearest point in the subspace $\mathcal{S}_{\boldsymbol{x}}$. Again, see Figure 7.3.2 for an illustration in the case $n = 1$.

The equivalent TLS formulation $\min_{\boldsymbol{x}} \| A \, \boldsymbol{x} - \boldsymbol{b} \|_2^2 / (\| \boldsymbol{x} \|_2^2 + 1)$ is convenient for regularizing the TLS problem additively; see next section. For more details see Section 10.2.

Further aspects of TLS problems

The sensitivity of the TLS problem

A study of the sensitivity of the TLS problem when there is a unique solution, i.e., when $\sigma'_n > \sigma_{n+1}$, is given in [106]. The starting point for the analysis is the formulation of the TLS problem as an eigenvalue problem for $C^T C = \sum_{i=1}^{n+1} \boldsymbol{v}_i \sigma_i^2 \boldsymbol{v}_i^T$ (i.e., σ_i^2 are the eigenvalues of $C^T C$ and $\boldsymbol{v}_i$ are the corresponding eigenvectors). The main tool used is the singular value interlacing property, Theorem 4 from Appendix B. Thus, if $\boldsymbol{x} \in \mathbb{R}^n$ and

$$C^T C \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix},$$

then $\boldsymbol{x}$ solves the TLS problem.

One interesting result is that the total least squares problem can be considered as a *de-regularization* process, which is apparent when looking at the first row of the above eigenvalue problem:

$$(A^T A - \sigma_{n+1}^2 I)\, \boldsymbol{x}_{\mathrm{TLS}} = A^T \boldsymbol{b}.$$

This implies that the TLS problem is *worse conditioned* than the associated LSQ problem.

An upper bound for the difference of the LSQ and TLS solutions is

$$\frac{\|\boldsymbol{x}_{\mathrm{TLS}} - \boldsymbol{x}^*\|_2}{\|\boldsymbol{x}*\|_2} \leq \frac{\sigma_{n+1}^2}{{\sigma'_n}^2 - \sigma_{n+1}^2},$$

so that ${\sigma'_n}^2 - \sigma_{n+1}^2$ measures the sensitivity of the TLS problem. This suggests that the TLS solution is unstable (the bounds are large) if $\sigma_n^{'}$ is close to σ_{n+1}, for example, if σ_{n+1} is (almost) a multiple singular value.

Stewart [231] proves that up to second-order terms in the error, $\boldsymbol{x}_{\mathrm{TLS}}$ is insensitive to column scalings of A. Thus, unlike ordinary LSQ problems, TLS cannot be better conditioned by column scalings.

The mixed LS-TLS problem

Suppose that only some of the columns of A have errors. This leads to the model

$$\min_{E,\boldsymbol{r}} (\|E_2\|_{\mathrm{F}}^2 + \|\boldsymbol{r}\|_2^2) \quad \text{subject to} \quad \begin{pmatrix} A_1 & A_2 + E_2 \end{pmatrix} \boldsymbol{x} = \boldsymbol{b} + \boldsymbol{r}, \qquad (7.3.6)$$

with $A_1 \in \mathbb{R}^{m \times n_1}$ and $A_2, E_2 \in \mathbb{R}^{m \times n_2}$. This *mixed LS-TLS problem* formulation encompasses the LSQ problem (when $n_2 = 0$), the TLS problem (when $n_1 = 0$), as well as a combination of both.

How can one find the solution to (7.3.6)? The basic idea, developed in [95], is to use a QR factorization of $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix}$ to transform the problem to a block structure, thereby reducing it to the solution of a smaller TLS problem that can be solved independently, plus an LSQ problem. First, compute the QR factorization $\begin{pmatrix} A & \boldsymbol{b} \end{pmatrix} = Q\, R$ with

$$R = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \qquad R_{11} \in \mathbb{R}^{n_1 \times n_1}, \quad R_{22} \in \mathbb{R}^{(n_2+1)\times(n_2+1)}.$$

Note that $\|F\|_{\mathrm{F}} = \|Q^T F\|_{\mathrm{F}}$ and therefore the constraints on the original and transformed systems are equivalent. Now compute the SVD of R_{22} and let $\boldsymbol{z}_2 = \boldsymbol{v}_{n_2+1}$ be the rightmost singular vector. We can then set $F_{12} = 0$ and solve the standard least squares problem

$$R_{11}\, \boldsymbol{z}_1 = -R_{12}\, \boldsymbol{z}_2.$$

Note that, as the TLS part of $\boldsymbol{z}$ has been obtained already and therefore F_{22} is minimized, there is no loss of generality when $F_{12} = 0$. Finally, compute the mixed LS-TLS solution as

$$\boldsymbol{x} = -\boldsymbol{z}(1:n)/\boldsymbol{z}(n+1), \qquad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix}.$$

7.4 Convex optimization

In an earlier section of this chapter we have already met a convex optimization problem: least squares with quadratic constraints. But there are many more problems of interest and some of them are related to the subject of this book.

Convex optimization has gained much attention in recent years, thanks to recent advances in algorithms to solve such problems. Also, the efforts of Stephen Boyd and his group at Stanford have helped to popularize the subject and provide invaluable software to solve all sizes of problems. One can say now that convex optimization problems have reached the point where they can be solved as assuredly as the basic linear algebra problems of the past.

One of the reasons for the success is that, although nonlinear, strictly convex problems have unique solutions. General nonlinear programming problems can have goal landscapes that are very bumpy and therefore it can be hard to find a desired optimal point and even harder to find a global one.

An important contribution in recent times has been in producing tools to identify convexity and to find formulations of problems that are convex. These problems look like general nonlinear programming problems, but the objective function $f(\boldsymbol{x})$ and the constraints $g_i(\boldsymbol{x})$ and $h_i(\boldsymbol{x})$ are required to be convex:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \le 0, \quad i = 1, ..., m \\ & h_i(\boldsymbol{x}) = 0, \quad i = 1, ..., p. \end{aligned}$$

In the past 30 years several researchers discovered that by adding a property (self-concordance) to these problems made it possible to use interior point methods via Newton iterations that were guaranteed to converge in polynomial time [167]. That was the breakthrough that allows us now to solve large problems in a reasonable amount of computer time and has provided much impetus to the application of these techniques to an ever increasing set of engineering and scientific problems.

A notable effort stemming from the research group of S. Boyd at Stanford is M. Grant's cvx MATLAB package, available from http://cvxr.com. van den Berg and Friedlander [240, 241] have also some interesting contributions, including efficient software for the solution of medium- and large-scale convex optimization problems; we also mention the Python software cvxopt by Dahl and Vandenberghe [56]. A comprehensive book on the subject is [30].

7.5 Compressed sensing

The theory and praxis of compressed sensing has become an important subject in the past few years. Compressed sensing is predicated on the fact that we often collect too much data and then compress it, i.e., a good deal of that data is unnecessary in some sense. Think of a mega-pixel digital photography and its JPEG compressed version. Although this is a lossy compression scheme, going back and forth allows for almost perfect reconstruction of the original image. Compressed sensing addresses the question: is it possible to collect much less data and use reconstruction algorithms that provide a perfect image? The answer in many cases is yes.

The idea of compressed sensing is that solution vectors can be represented by much fewer coefficients if an appropriate basis is used, i.e., they are sparse in such a base, say, of wavelets [164]. Following [34, 35] we consider the general problem of reconstructing a vector $\boldsymbol{x} \in \mathbb{R}^N$ from linear measurements, $y_k = \boldsymbol{\varphi}_k^T \boldsymbol{x}$ for $k = 1, \ldots, K$. Assembling all these measurements in matrix form we obtain the relation: $\boldsymbol{y} = \Phi\, \boldsymbol{x}$, where $\boldsymbol{y} \in \mathbb{R}^K$ and $\boldsymbol{\varphi}_k$ are the rows of the matrix Φ.

The important next concept is that compressed sensing attempts to reconstruct the original vector $\boldsymbol{x}$ from a number of samples smaller than what the Nyquist-Shannon theorem says is possible. The latter theorem states that if a signal contains a maximum frequency f (i.e., a minimum wavelength $w = 1/f$), then in order to reconstruct it exactly it is necessary to have samples in the time domain that are spaced no more than $1/(2f)$ units apart. The key to overcoming this restriction in the compressed sensing approach is to use nonlinear reconstruction methods that combine l_2 and l_1 terms.

We emphasize that in this approach the system $\boldsymbol{y} = \Phi\, \boldsymbol{x}$ is underdetermined, i.e., the matrix Φ has more columns than rows. It turns out that if one assumes that the vector $\boldsymbol{x}$ is sparse in some basis B, then solving the following convex optimization problem:

$$\min_{\widetilde{\boldsymbol{x}}} \|\widetilde{\boldsymbol{x}}\|_1 \quad \text{subject to} \quad \Phi B\, \widetilde{\boldsymbol{x}} = \boldsymbol{y}, \tag{7.5.1}$$

reconstructs $\boldsymbol{x} = B\widetilde{\boldsymbol{x}}$ exactly with high probability. The precise statement

is as follows.

Theorem 82. *Assume that $\boldsymbol{x}$ has at most S nonzero elements and that we take K random linear measurements, where K satisfies*

$$K \geqslant C \cdot S \cdot \log N,$$

where $C = 22(\delta+1)$ and $\delta > 0$. Then, solving (7.5.1) reconstructs $\boldsymbol{x}$ exactly with probability greater than $1 - O(N^{-\delta})$.

The "secret" is that the Nyquist-Shannon theorem talks about a whole band of frequencies, from zero to the highest frequency, while here we are considering a sparse set of frequencies that may be in small disconnected intervals. Observe that for applications of interest, the dimension N in problem (7.5.1) can be very large and there resides its complexity. This problem is convex and can be reduced to a linear programming one. For large N it is proposed to use an interior-point method that basically solves the optimality conditions by Newton's method, taking care of staying feasible throughout the iteration. E. Candes (Caltech, Stanford) offers a number of free software packages for this type of applications. Some very good recent references are [240, 241].

As usual, in the presence of noise and even if the constraints in (7.5.1) are theoretically satisfied, it is more appropriate to replace that problem by $\min_{\widetilde{\boldsymbol{x}}} \|\widetilde{\boldsymbol{x}}\|_1$ subject to $\|\Phi\,\widetilde{\boldsymbol{x}} - \boldsymbol{y}\|_2 \leq \alpha$, where the positive parameter α is an estimate of the data noise level.

It is interesting to notice the connection with the basic solution of rank-deficient problems introduced in Section 2.3. Already in 1964 J. B. Rosen wrote a paper [209] about its calculation and in [193] an algorithm and implementation for calculating the minimal and basic solutions was described. That report contains extensive numerical experimentation and a complete Algol program. Unfortunately, the largest matrix size treated there was 40 and therefore without further investigation that algorithm may not be competitive today for these large problems. Basic solutions are used in some algorithms as initial guesses to solve this problem, where one generally expects many fewer nonzero elements in the solution than the rank of the matrix.

Chapter 8

Nonlinear Least Squares Problems

So far we have discussed data fitting problems in which all the unknown parameters appear linearly in the fitting model $M(\boldsymbol{x}, t)$, leading to linear least squares problems for which we can, in principle, write down a closed-form solution. We now turn to nonlinear least squares problems (NLLSQ) for which this is not true, due to (some of) the unknown parameters appearing nonlinearly in the model.

8.1 Introduction

To make precise what we mean by the term "nonlinear" we give the following definitions. A parameter α of the function f appears nonlinearly if the derivative $\partial f/\partial\alpha$ is a function of α. A parameterized fitting model $M(\boldsymbol{x}, t)$ is nonlinear if at least one of the parameters in $\boldsymbol{x}$ appear nonlinearly. For example, in the exponential decay model

$$M(x_1, x_2, t) = x_1 e^{-x_2 t}$$

we have $\partial M/\partial x_1 = e^{-x_2 t}$ (which is independent of x_1, so is linear in it) and $\partial M/\partial x_2 = -t\, x_1 e^{-x_2 t}$ (which depends on x_2) and thus M is a nonlinear model with the parameter x_2 appearing nonlinearly. We start with a few examples of nonlinear models.

Example 83. ***Gaussian model.*** *All measuring devices somehow influence the signal that they measure. If we measure a time signal g_{mes}, then we can often model it as a convolution of the true signal $g_{\text{true}}(t)$ and an*

instrument response function $\Gamma(t)$*:*

$$g_{\text{mes}}(t) = \int_{-\infty}^{\infty} \Gamma(t-\tau)\, g_{\text{true}}(\tau)\, d\tau.$$

A very common model for $\Gamma(t)$ *is the non-normalized Gaussian function*

$$M(\boldsymbol{x}, t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}, \tag{8.1.1}$$

where x_1 *is the amplitude,* x_2 *is the time shift and* x_3 *determines the width of the Gaussian function. The parameters* x_2 *and* x_3 *appear nonlinearly in this model. The Gaussian model also arises in many other data fitting problems.*

Example 84. ***Rational model.*** *Another model function that often arises in nonlinear data fitting problems is the rational function, i.e., a quotient of two polynomials of degree* p *and* q*, respectively,*

$$M(\boldsymbol{x}, t) = \frac{P(t)}{Q(t)} = \frac{x_1 t^p + x_2 t^{p-1} + \cdots + x_p t + x_{p+1}}{x_{p+2} t^q + x_{p+3} t^{q-1} + \cdots + x_{p+q+1} t + x_{p+q+2}}, \tag{8.1.2}$$

with a total of $n = p + q + 2$ *parameters.*

Rational models arise, e.g., in parameter estimation problems such as chemical reaction kinetics, signal analysis (through Laplace and z transforms), system identification and in general as a transfer function for a linear time-invariant system (similar to the above-mentioned instrument response function). In these models, the coefficients of the two polynomials or, equivalently, their roots, characterize the dynamical behavior of the system.

Rational functions are also commonly used as empirical data approximation functions. Their advantage is that they have a broader scope than polynomials, yet they are still simple to evaluate.

The basic idea of nonlinear data fitting is the same as described in Chapter 1, the only difference being that we now use a fitting model $M(\boldsymbol{x}, t)$ in which (some of) the parameters $\boldsymbol{x}$ appear nonlinearly and that leads to a nonlinear optimization problem. We are given noisy measurements $y_i = \Gamma(t_i) + e_i$ for $i = 1, \ldots, m$, where $\Gamma(t)$ is the pure-data function that gives the true relationship between t and the noise-free data, and we seek to identify the model parameters $\boldsymbol{x}$ such that $M(\boldsymbol{x}, t)$ gives the best fit to the noisy data. The likelihood function $P_{\boldsymbol{x}}$ for $\boldsymbol{x}$ is the same as before, cf. (1.3.1), leading us to consider the weighted residuals $r_i(\boldsymbol{x})/\varsigma_i = (y_i - M(\boldsymbol{x}, t_i)/\varsigma_i$ and to minimize the weighted sum-of-squares residuals.

Definition 85. ***Nonlinear least squares problem (NLLSQ).*** *In the case of white noise (where the errors have a common variance* ς^2*), find a*

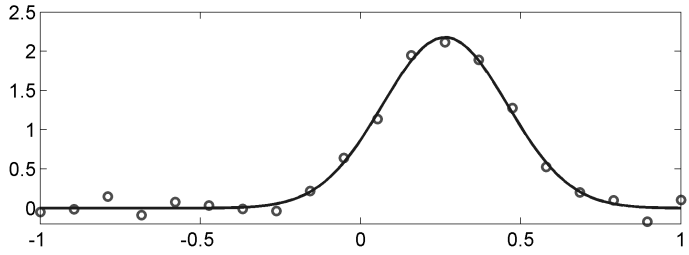


Figure 8.1.1: Fitting a non-normalized Gaussian function (full line) to noisy data (circles).

minimizer $\boldsymbol{x}^*$ *of the nonlinear objective function* f *with the special form*

$$\boxed{\min_{\boldsymbol{x}} f(\boldsymbol{x}) \equiv \min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \min_{\boldsymbol{x}} \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2,} \tag{8.1.3}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ *and* $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T$ *is a vector-valued function of the residuals for each data point:*

$$r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x}, t_i), \qquad i = 1, \ldots, m,$$

in which y_i *are the measured data corresponding to* t_i *and* $M(\boldsymbol{x}, t)$ *is the model function.*

The factor $\frac{1}{2}$ is added for practical reasons as will be clear later. Simply replace $r_i(\boldsymbol{x})$ by $w_i\, r_i(\boldsymbol{x})/\varsigma_i$ in (8.1.3), where $w_i = \varsigma_i^{-1}$ is the inverse of the standard deviation for the noise in y_i, to obtain the weighted problem $\min_{\boldsymbol{x}} \frac{1}{2} \|W\boldsymbol{r}(\boldsymbol{x})\|_2^2$ for problems when the errors have not the same variance values.

Example 86. ***Fitting with the Gaussian model.*** *As an example, we consider fitting the instrument response function in Example 83 by the Gaussian model (8.1.1). First note that if we choose* $g_{\text{true}}(\tau) = \delta(\tau)$*, a delta function, then we have* $g_{\text{mes}}(t) = \Gamma(t)$*, meaning that we ideally measure the instrument response function. In practice, by sampling* $g_{\text{mes}}(t)$ *for selected values* $t_1, t_2, \ldots, t_m$ *of* t *we obtain noisy data* $y_i = \Gamma(t_i) + e_i$ *for* $i = 1, \ldots, m$ *to which we fit the Gaussian model* $M(\boldsymbol{x}, t)$*. Figure 8.1.1 illustrates this. The circles are the noisy data generated from an exact Gaussian model with* $x_1 = 2.2$*,* $x_2 = 0.26$ *and* $x_3 = 0.2$*, to which we added Gaussian noise with standard deviation* $\varsigma = 0.1$*. The full line is the least squares fit with parameters* $x_1 = 2.179$*,* $x_2 = 0.264$ *and* $x_3 = 0.194$*. In the next chapter we discuss how to compute this fit.*

Example 87. ***Nuclear magnetic resonance spectroscopy.*** *This is a technique that measures the response of nuclei of certain atoms that possess spin when they are immersed in a static magnetic field and they are exposed to a second oscillating magnetic field. NMR spectroscopy, which studies the interaction of the electromagnetic radiation with matter, is used as a non-invasive technique to obtain in vivo information about chemical changes (e.g., concentration, pH), in living organisms. An NMR spectrum of, for example, the human brain can be used to identify pathology or biochemical processes or to monitor the effect of medication.*

The model used to represent the measured NMR signals $y_i \in \mathbb{C}$ in the frequency domain is a truncated Fourier series of the Lorentzian distribution:

$$y_i \simeq M(\boldsymbol{a}, \boldsymbol{\phi}, \boldsymbol{d}, \boldsymbol{\omega}, t_i) = \sum_{k=1}^{K} a_k e^{\hat{\imath}\phi_k} e^{(-d_k + j\omega_k)t_i}, \qquad i = 1, \ldots, m,$$

where $\hat{\imath}$ denotes the imaginary unit. The parameters of the NMR signal provide information about the molecules: K represents the number of different resonance frequencies, the angular frequency ω_k of the individual spectral components identifies the molecules, the damping d_k characterizes the mobility, the amplitude a_k is proportional to the number of molecules present, and ϕ_k is the phase. All these parameters are real. To determine the NMR parameters through a least squares fit, we minimize the squared modulus of the difference between the measured spectrum and the model function:

$$\min_{\boldsymbol{a},\boldsymbol{\phi},\boldsymbol{d},\boldsymbol{\omega}} \sum_{i=1}^{m} |y_i - M(\boldsymbol{a}, \boldsymbol{\phi}, \boldsymbol{d}, \boldsymbol{\omega}, t_i)|^2 .$$

This is another example of a nonlinear least squares problem.

8.2 Unconstrained problems

Generally, nonlinear least squares problems will have multiple solutions and without a priori knowledge of the objective function it would be too expensive to determine numerically a global minimum, as one can only calculate the function and its derivatives at a limited number of points. Thus, the algorithms in this book will be limited to the determination of local minima. A certain degree of smoothness of the objective function will be required, i.e., having either one or possibly two continuous derivatives, so that the Taylor expansion applies.

Optimality conditions

Recall (Appendix C) that the gradient $\nabla f(\boldsymbol{x})$ of a scalar function $f(\boldsymbol{x})$ of n variables is the vector with elements

$$\frac{\partial f(\boldsymbol{x})}{\partial x_j}, \qquad j = 1, \ldots, n$$

and the Hessian $\nabla^2 f(\boldsymbol{x})$ is the symmetric matrix with elements

$$[\nabla^2 f(\boldsymbol{x})]_{ij} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}, \qquad i, j = 1, \ldots, n.$$

The conditions for $\boldsymbol{x}^*$ to be a *local minimum* of a twice continouously differentiable function f are

1. **First-order necessary condition**. $\boldsymbol{x}^*$ is a stationary point, i.e., the gradient of f at $\boldsymbol{x}^*$ is zero: $\nabla f(\boldsymbol{x}^*) = 0$.
2. **Second-order sufficient conditions.** The Hessian $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite.

Now we consider the special case where f is the function in the nonlinear least squares problem (8.1.3), i.e.,

$$f(\boldsymbol{x}) = \tfrac{1}{2} \sum_{i=1}^{m} r_i(\boldsymbol{x})^2, \qquad r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x}, t_i).$$

The Jacobian $J(\boldsymbol{x})$ of the vector function $\boldsymbol{r}(\boldsymbol{x})$ is defined as the matrix with elements

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j} = -\frac{\partial M(\boldsymbol{x}, t_i)}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

Note that the ith row of $J(\boldsymbol{x})$ equals the transpose of the gradient of $r_i(\boldsymbol{x})$ and also:

$$\nabla r_i(\boldsymbol{x})^T = -\nabla M(\boldsymbol{x}, t_i)^T, \qquad i = 1, \ldots, m.$$

The elements of the gradient and the Hessian of f are given by

$$\begin{aligned} \frac{\partial f(\boldsymbol{x})}{\partial x_j} &= \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \\ [\nabla^2 f(\boldsymbol{x})]_{k\ell} \quad = \quad \frac{\partial^2 f(\boldsymbol{x})}{\partial x_k \partial x_\ell} &= \sum_{i=1}^{m} \frac{\partial r_i(\boldsymbol{x})}{\partial x_k} \frac{\partial r_i(\boldsymbol{x})}{\partial x_\ell} + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_k \partial x_\ell}, \end{aligned}$$

and it follows immediately that the gradient and Hessian can be written in matrix form as

$$\begin{aligned}
\nabla f(\boldsymbol{x}) &= J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}), \\
\nabla^2 f(\boldsymbol{x}) &= J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}), \\
\left[\nabla^2 r_i(\boldsymbol{x})\right]_{k\ell} &= -\left[\nabla^2 M(\boldsymbol{x}, t_i)\right]_{k\ell} = -\frac{\partial^2 M(\boldsymbol{x}, t_i)}{\partial x_k \partial x_\ell}, \quad k, \ell = 1, \ldots, m.
\end{aligned}$$

Notice the minus sign in the expression for $\nabla^2 r_i(\boldsymbol{x})$. The optimality conditions now take the special form

$$\boxed{\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0}} \tag{8.2.1}$$

and

$$\boxed{\nabla^2 f(\boldsymbol{x}) = J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}) \text{ is positive definite.}} \tag{8.2.2}$$

The fact that distinguishes the least squares problem from among the general optimization problems is that the first – and often dominant – term $J(\boldsymbol{x})^T J(\boldsymbol{x})$ of the Hessian $\nabla^2 f(\boldsymbol{x})$ contains only the Jacobian $J(\boldsymbol{x})$ of $\boldsymbol{r}(\boldsymbol{x})$, i.e., only first derivatives. Observe that in the second term the second derivatives are multiplied by the residuals. Thus, if the model is adequate to represent the data, then these residuals will be small near the solution and therefore this term will be of secondary importance. In this case one gets an important part of the Hessian "for free" if one has already computed the gradient. Most specialized algorithms exploit this structural property of the Hessian.

An inspection of the Hessian of f will show two comparatively easy NLLSQ cases. As we observed above, the term $\sum_{i=1}^{m} r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x})$ will be small if the residuals are small. An additional favorable case occurs when the problem is only mildly nonlinear, i.e., all the Hessians $\nabla^2 r_i(\boldsymbol{x}) = -\nabla^2 M(\boldsymbol{x}, t_i)$ have small norm. Most algorithms that neglect this second term will then work satisfactorily. The smallest eigenvalue $\lambda_{\min}$ of $J(\boldsymbol{x})^T J(\boldsymbol{x})$ can be used to quantify the relative importance of the two terms of $\nabla^2 f(\boldsymbol{x})$: the first term of the Hessian dominates if for all $\boldsymbol{x}$ near a minimum $\boldsymbol{x}^*$ the quantities $|r_i(\boldsymbol{x})| \, \|\nabla^2 r_i(\boldsymbol{x})\|_2$ for $i = 1, \ldots, m$ are small relative to $\lambda_{\min}$. This obviously holds in the special case of a consistent problem where $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$.

The optimality conditions can be interpreted geometrically by observing that the gradient $\nabla f(\boldsymbol{x})$ is always a vector normal to the level set that passes through the point $\boldsymbol{x}$ (we are assuming in what follows that $f(\boldsymbol{x})$ is

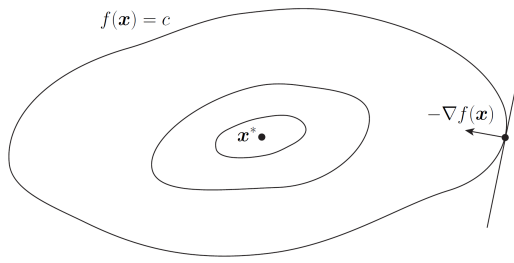


Figure 8.2.1: Level sets $L(c)$ (in $\mathbb{R}^2$ they are level curves) for a function f whose minimizer x^* is located at the black dot. The tangent plane is a line perpendicular to the gradient $\nabla f(\boldsymbol{x})$, and the negative gradient is the direction of steepest descent at $\boldsymbol{x}$.

twice differentiable, for simplicity). A *level set* $L(c)$ (level curve in $\mathbb{R}^2$) is defined formally as

$$L(c) = \{\boldsymbol{x} : \ f(\boldsymbol{x}) = c\}.$$

The tangent plane at $\boldsymbol{x}$ is $\{\, \boldsymbol{y} \in \mathbb{R}^n \,|\, \nabla f(\boldsymbol{x})^T(\boldsymbol{y}-\boldsymbol{x}) = 0\}$, which shows that $\nabla f(\boldsymbol{x})$ is its normal and thus normal to the level set at $\boldsymbol{x}$.

This leads to a geometric interpretation of directions of descent. A direction $\boldsymbol{p}$ with $\|\boldsymbol{p}\|_2 = 1$ is said to be of *descent* (from the point $\boldsymbol{x}$) if $f(\boldsymbol{x}+t\boldsymbol{p}) < f(\boldsymbol{x})$ for $0 < t < t_0$. It turns out that directions of descent can be characterized using the gradient vector. In fact, by Taylor's expansion we have

$$f(\boldsymbol{x} + t\boldsymbol{p}) = f(\boldsymbol{x}) + t\nabla f(\boldsymbol{x})^T\boldsymbol{p} + O(t^2).$$

Thus, for the descent condition to hold we need to have $\nabla f(\boldsymbol{x})^T\boldsymbol{p} < 0$, since for sufficiently small t the linear term will dominate. In addition we have $\nabla f(\boldsymbol{x})^T\boldsymbol{p} = \cos(\phi)\|\nabla f(\boldsymbol{x})\|_2$ where ϕ is the angle between $\boldsymbol{p}$ and the gradient. From this we conclude that the direction $\boldsymbol{p} = -\nabla f(\boldsymbol{x})$ is of maximum descent, or *steepest descent*, while any direction in the half-space defined by the tangent plane and containing the negative gradient is also a direction of descent, since $\cos(\phi)$ is negative there. Figure 8.2.1 illustrates this point. Note that a stationary point is one for which there are no descent directions, i.e., $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$.

We conclude with a useful geometric result when $J(\boldsymbol{x}^*)$ has full rank. In this case the characterization of a minimum can be expressed by using the so-called normal curvature matrix (see [9]) associated with the surface defined by $\boldsymbol{r}(\boldsymbol{x})$ and with respect to the normalized vector $\boldsymbol{r}(\boldsymbol{x})/\left\|\boldsymbol{r}(\boldsymbol{x})\right\|_2$, defined as

$$K(\boldsymbol{x}) = -\frac{1}{\|\boldsymbol{r}(\boldsymbol{x})\|_2}(J(\boldsymbol{x})^\dagger)^T\left(\sum_{i=1}^{m} r_i(\boldsymbol{x})\nabla^2 r_i(\boldsymbol{x})\right)J(\boldsymbol{x})^\dagger.$$

Then the condition for a minimum can be reformulated as follows.

Condition 88. *If $J(\boldsymbol{x}^*)^T(I - \|\boldsymbol{r}(\boldsymbol{x}^*)\|_2 K(\boldsymbol{x}^*))J(\boldsymbol{x}^*)$ is positive definite, then $\boldsymbol{x}^*$ is a local minimum.*

The role of local approximations

Local approximations always play an important role in the treatment of nonlinear problems and this case is no exception. We start with the Taylor expansion of the model function:

$$M(\boldsymbol{x}+\boldsymbol{h},t) = M(\boldsymbol{x},t) + \nabla M(\boldsymbol{x},t)^T\boldsymbol{h} + \tfrac{1}{2}\boldsymbol{h}^T\nabla^2 M(\boldsymbol{x},t)\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^3).$$

Now consider the ith residual $r_i(\boldsymbol{x}) = y_i - M(\boldsymbol{x},t_i)$, whose Taylor expansion for $i = 1,\ldots,m$ is given by

$$\begin{aligned} r_i(\boldsymbol{x}+\boldsymbol{h}) &= y_i - M(\boldsymbol{x}+\boldsymbol{h},t_i) \\ &= y_i - M(\boldsymbol{x},t_i) - \nabla M(\boldsymbol{x},t_i)^T\boldsymbol{h} - \\ &\quad \tfrac{1}{2}\boldsymbol{h}^T\nabla^2 M(\boldsymbol{x},t_i)\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^3). \end{aligned}$$

Hence we can write

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\,\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^2),$$

which is a local linear approximation at $\boldsymbol{x}$ valid for small $\boldsymbol{h}$. If we keep $\boldsymbol{x}$ fixed and consider the minimization problem

$$\min_{\boldsymbol{h}} \|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h})\|_2 \simeq \min_{\boldsymbol{h}} \|J(\boldsymbol{x})\,\boldsymbol{h} + \boldsymbol{r}(\boldsymbol{x})\|_2, \tag{8.2.3}$$

it is clear that we can approximate locally the nonlinear least squares problem with a linear one in the variable $\boldsymbol{h}$. Moreover, we see that the $\boldsymbol{h}$ that minimizes $\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h})\|_2$ can be approximated by

$$\boldsymbol{h} \approx -(J(\boldsymbol{x})^T J(\boldsymbol{x}))^{-1} J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = -J(\boldsymbol{x})^\dagger \boldsymbol{r}(\boldsymbol{x}),$$

where we have neglected higher-order terms. In the next chapter we shall see how this expression provides a basis for some of the numerical methods for solving the NLLSQ problem.

As a special case, let us now consider the local approximation (8.2.3) at a least squares solution $\boldsymbol{x}^*$ (a local minimizer) where the local, linear least squares problem takes the form

$$\min_{\boldsymbol{h}} \|J(\boldsymbol{x}^*)\,\boldsymbol{h} + \boldsymbol{r}(\boldsymbol{x}^*)\|_2.$$

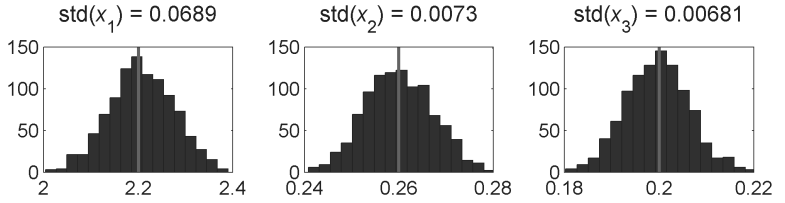


Figure 8.2.2: Histograms and standard deviations for the least squares solutions to 1000 noisy realizations of the Gaussian model fitting problem. The vertical lines show the exact values of the parameters.

If we introduce $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{h}$, then we can reformulate the above problem as one in $\boldsymbol{x}$,

$$\min_{\boldsymbol{x}} \|J(\boldsymbol{x}^*)\,\boldsymbol{x} - J(\boldsymbol{x}^*)\,\boldsymbol{x}^* + \boldsymbol{r}(\boldsymbol{x}^*)\|_2,$$

which leads to an approximation of the covariance matrix for $\boldsymbol{x}^*$ by using the covariance matrix for the above linear least squares problem, i.e.,

$$\begin{aligned} \mathrm{Cov}(\boldsymbol{x}^*) &\simeq J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(J(\boldsymbol{x}^*)\,\boldsymbol{x}^* - \boldsymbol{r}(\boldsymbol{x}^*)\big)(J(\boldsymbol{x}^*)^\dagger)^T \\ &= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}\big(\boldsymbol{r}(\boldsymbol{x}^*) - J(\boldsymbol{x}^*)\,\boldsymbol{x}^*\big)(J(\boldsymbol{x}^*)^\dagger)^T \\ &= J(\boldsymbol{x}^*)^\dagger \mathrm{Cov}(\boldsymbol{y})(J(\boldsymbol{x}^*)^\dagger)^T. \end{aligned} \tag{8.2.4}$$

Here, we have used that $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{y} - (M(\boldsymbol{x}^*, t_1), \ldots, M(\boldsymbol{x}^*, t_m))^T$ and that all the terms except $\boldsymbol{y}$ in $\boldsymbol{r}(\boldsymbol{x}^*) + J(\boldsymbol{x}^*)\,\boldsymbol{x}^*$ are considered constant.

The above equation provides a way to approximately assess the uncertainties in the least squares solution $\boldsymbol{x}^*$ for the nonlinear case, similar to the result in Section 2.1 for the linear case. In particular, we see that the Jacobian $J(\boldsymbol{x}^*)$ at the solution plays the role of the matrix in the linear case. If the errors $\boldsymbol{e}$ in the data are uncorrelated with the exact data and have covariance $\mathrm{Cov}(\boldsymbol{e}) = \varsigma^2 I$, then $\mathrm{Cov}(\boldsymbol{x}^*) \simeq \varsigma^2 \big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}$.

Example 89. *To illustrate the use of the above covariance matrix estimate, we return to the Gaussian model $M(\boldsymbol{x}, t)$ from Examples 83 and 86, with exact parameters $x_1 = 2.2$, $x_2 = 0.26$ and $x_3 = 0.2$ and with noise level $\varsigma = 0.1$. The elements of the Jacobian for this problem are, for $i = 1, \ldots, m$:*

$$\begin{aligned} [J(\boldsymbol{x})]_{i,1} &= e^{-(t_i - x_2)^2/(2x_3^2)}, \\ [J(\boldsymbol{x})]_{i,2} &= x_1 \frac{t_i - x_2}{x_3^2}\, e^{-(t_i - x_2)^2/(2x_3^2)}, \\ [J(\boldsymbol{x})]_{i,3} &= x_1 \frac{(t_i - x_2)^2}{x_3^3}\, e^{-(t_i - x_2)^2/(2x_3^2)}. \end{aligned}$$

The approximate standard deviations for the three estimated parameters are the square roots of the diagonal of $\varsigma^2\big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}$. *In this case we get the three values*

$$6.58 \cdot 10^{-2}, \qquad 6.82 \cdot 10^{-3}, \qquad 6.82 \cdot 10^{-3}.$$

We compute the least squares solution $\boldsymbol{x}^*$ *for 1000 realizations of the noise. Histograms of these parameters are shown in Figure 8.2.2. The standard deviations of these estimated parameters are*

$$\text{std}(x_1) = 6.89 \cdot 10^{-2}, \quad \text{std}(x_2) = 7.30 \cdot 10^{-3}, \quad \text{std}(x_3) = 6.81 \cdot 10^{-3}.$$

In this example, the theoretical standard deviation estimates are in very good agreement with the observed ones.

8.3 Optimality conditions for constrained problems

In the previous section we reviewed the conditions for a point $\boldsymbol{x}^*$ to be a local minimizer of an unconstrained nonlinear least squares problem. Now we consider a constrained problem

$$\boxed{\min_{\boldsymbol{x} \in C} \tfrac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|_2^2,} \tag{8.3.1}$$

where the residual vector $\boldsymbol{r}(\boldsymbol{x})$ is as above and the set $C \subset \mathbb{R}^n$ (called the feasible region) is defined by a set of inequality constraints:

$$C = \{\, \boldsymbol{x} \,|\, c_i(\boldsymbol{x}) \leq 0,\ i = 1, \ldots, p\},$$

where the functions $c_i(\boldsymbol{x})$ are twice differentiable. In the unconstrained case, it could be that the only minimum is at infinity (think of $f(x) = a + bx$). If instead we limit the variables to be in a closed, bounded set, then the Bolzano-Weierstrass theorem [79] ensures us that there will be at least one minimum point.

A simple way to define a bounded set C is to impose constraints on the variables. The simplest constraints are those that set bounds on the variables: $l_i \leq x_i \leq u_i$ for $i = 1, \ldots, p$. A full set of such constraints (i.e., $p = n$) defines a bounded box in $\mathbb{R}^n$ and then we are guaranteed that $f(\boldsymbol{x})$ will have a minimum in the set. For general constraints some of the functions $c_i(\boldsymbol{x})$ can be nonlinear.

The points that satisfy all the constraints are called *feasible*. A constraint is *active* if it is satisfied with equality, i.e., the point is on a boundary of the feasible set. If we ignore the constraints, unconstrained minima can

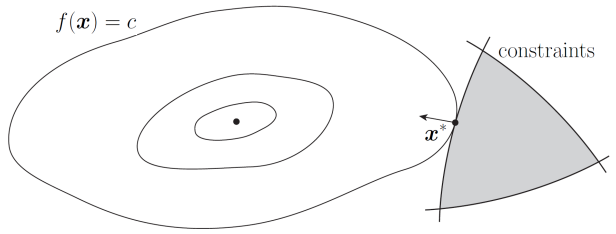


Figure 8.3.1: Optimality condition for constrained optimization. The shaded area illustrates the feasible region C defined by three constraints. One of the constraints is active at the solution $\boldsymbol{x}^*$ and at this point the gradient of f is collinear with the normal to the constraint.

occur inside or outside the feasible region. If they occur inside, then the optimality conditions are the same as in the unconstrained case. However, if all unconstrained minima are infeasible, then a constrained minimum must occur on the boundary of the feasible region and the optimality conditions will be different.

Let us think geometrically (better still, in $\mathbb{R}^2$, see Figure 8.3.1). The level curve values of $f(\boldsymbol{x})$ decrease toward the unconstrained minimum. Thus, a constrained minimum will lie on the level curve with the lowest value of $f(\boldsymbol{x})$ that is still in contact with the feasible region and at the constrained minimum there should not be any feasible descent directions. A moment of reflection tells us that the level curve should be tangent to the active constraint (the situation is more complicated if more than one constraint is active). That means that the normal to the active constraint and the gradient (pointing to the outside of the constraint region) at that point are collinear, which can be expressed as

$$\nabla f(\boldsymbol{x}) + \lambda \nabla c_i(\boldsymbol{x}) = 0, \tag{8.3.2}$$

where $c_i(\boldsymbol{x})$ represents the active constraint and $\lambda > 0$ is a so-called Lagrange multiplier for the constraint. (If the constraint were $c_i(\boldsymbol{x}) \geq 0$, then we should require $\lambda < 0$.) We see that (8.3.2) is true because, if $\boldsymbol{p}$ is a descent direction, we have

$$\boldsymbol{p}^T \nabla f(\boldsymbol{x}) + \lambda \boldsymbol{p}^T \nabla c_i(\boldsymbol{x}) = 0$$

or, since $\lambda > 0$,

$$\boldsymbol{p}^T \nabla c_i(\boldsymbol{x}) > 0,$$

and therefore $\boldsymbol{p}$ must point outside the feasible region; see Figure 8.3.1 for an illustration. This is a simplified version of the famous Karush-Kuhn-Tucker

first-order optimality conditions for general nonlinear optimization [139, 147]. Curiously enough, in the original Kuhn-Tucker paper these conditions are derived from considerations based on multiobjective optimization!

The general case when more than one constraint is active at the solution, can be discussed similarly. Now the infeasible directions instead of being in a half-space determined by the tangent plane of a single constraint will be in a cone formed by the normals to the tangent planes associated with the various active constraints. The corresponding optimality condition is

$$\boxed{\nabla f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i \nabla c_i(\boldsymbol{x}) = 0,} \tag{8.3.3}$$

where the Lagrange multipliers satisfy $\lambda_i > 0$ for the active constraints and $\lambda_i = 0$ for the inactive ones. We refer to [170] for more details about optimality constraints and Lagrange multipliers.

8.4 Separable nonlinear least squares problems

In many practical applications the unknown parameters of the NLLSQ problem can be separated into two disjoint sets, so that the optimization with respect to one set is easier than with respect to the other. This suggests the idea of eliminating the parameters in the easy set and minimizing the resulting functional, which will then depend only on the remaining variables. A natural situation that will be considered in detail here arises when some of the variables appear linearly.

The initial approach to this problem was considered by H. Scolnik in his Doctoral Thesis at the University of Zurich. A first paper with generalizations was [113]. This was followed by an extensive generalization that included a detailed algorithmic description [101, 102] and a computer program called VARPRO that become very popular and is still in use. For a recent detailed survey of applications see [103] and [198]. A paper that includes constraints is [142]. For multiple right hand sides see [98, 141].

A separable nonlinear least squares problem has a model of the special form

$$\boxed{M(\boldsymbol{a}, \boldsymbol{\alpha}, t) = \sum_{j=1}^{n} a_j \, \phi_j(\boldsymbol{\alpha}, t),} \tag{8.4.1}$$

where the two vectors $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{\alpha} \in \mathbb{R}^k$ contain the parameters to be determined and $\phi_j(\boldsymbol{\alpha}, t)$ are functions in which the parameters $\boldsymbol{\alpha}$ appear nonlinearly. Fitting this model to a set of m data points (t_i, y_i) with $m > n + k$ leads to the least squares problem

$$\min_{\mathbf{a},\boldsymbol{\alpha}} f(\boldsymbol{a},\, \boldsymbol{\alpha}) = \min_{\mathbf{a},\boldsymbol{\alpha}} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{a},\, \boldsymbol{\alpha})\|_2^2 = \min_{\mathbf{a},\boldsymbol{\alpha}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\, \boldsymbol{a}\|_2^2.$$

In this expression, $\Phi(\boldsymbol{\alpha})$ is an $m \times n$ matrix function with elements

$$\Phi(\boldsymbol{\alpha}) = \phi_j(\boldsymbol{\alpha},\, t_i), \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

The special case when $k = n$ and $\phi_j = e^{\alpha_j t}$ is called *exponential data fitting*. In this as in other separable nonlinear least squares problems the matrix $\Phi(\boldsymbol{\alpha})$ is often ill-conditioned.

The variable projection algorithm – to be discussed in detail in the next chapter – is based on the following ideas. For any fixed $\boldsymbol{\alpha}$, the problem

$$\min_{\mathbf{a}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\, \boldsymbol{a}\|_2^2$$

is linear with minimum-norm solution $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha})^\dagger \boldsymbol{y}$, where $\Phi(\boldsymbol{\alpha})^\dagger$ is the Moore-Penrose generalized inverse of the rectangular matrix $\Phi(\boldsymbol{\alpha})$, as we saw in Chapter 3.

Substituting this value of $\boldsymbol{a}$ into the original problem, we obtain a reduced nonlinear problem depending only on the nonlinear parameters $\boldsymbol{\alpha}$, with the associated function

$$f_2(\boldsymbol{\alpha}) = \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger\, \boldsymbol{y}\|_2^2 = \tfrac{1}{2}\|P_{\Phi(\boldsymbol{\alpha})}\, \boldsymbol{y}\|_2^2,$$

where, for a given $\boldsymbol{\alpha}$, the matrix

$$P_{\Phi(\boldsymbol{\alpha})} = I - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger$$

is the projector onto the orthogonal complement of the column space of the matrix $\Phi(\boldsymbol{\alpha})$. Thus the name variable projection given to this reduction procedure. The solution of the nonlinear least squares problem $\min_{\boldsymbol{\alpha}} f_2(\boldsymbol{\alpha})$ is discussed in the next chapter.

Once a minimizer $\boldsymbol{\alpha}^*$ for $f_2(\boldsymbol{\alpha})$ is obtained, it is substituted into the linear problem $\min_{\mathbf{a}} \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha}^*)\, \boldsymbol{a}\|_2^2$, which is then solved to obtain $\boldsymbol{a}^*$. The least squares solution to the original problem is then $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$.

The justification for the variable projection algorithm is the following theorem (Theorem 2.1 proved in [101]), which essentially states that the set of stationary points of the original functional and that of the reduced one are identical.

Theorem 90. *Let $f(\boldsymbol{a},\, \boldsymbol{\alpha})$ and $f_2(\boldsymbol{\alpha})$ be defined as above. We assume that in an open set Ω containing the solution, the matrix $\Phi(\boldsymbol{\alpha})$ has constant rank $r \leqq \min(m, n)$.*

1. *If $\boldsymbol{\alpha}^*$ is a critical point (or a global minimizer in Ω) of $f_2(\boldsymbol{\alpha})$ and $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha})^\dagger \boldsymbol{y}$, then $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$ is a critical point of $f(\boldsymbol{a},\, \boldsymbol{\alpha})$ (or a global minimizer in Ω) and $f(\boldsymbol{a}^*,\, \boldsymbol{\alpha}^*) = f_2(\boldsymbol{\alpha}^*)$.*

2. *If* $(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$ *is a global minimizer of* $f(\boldsymbol{a}, \boldsymbol{\alpha})$ *for* $\boldsymbol{\alpha} \in \Omega$*, then* $\boldsymbol{\alpha}^*$ *is a global minimizer of* $f_2(\boldsymbol{\alpha})$ *in* Ω *and* $f_2(\boldsymbol{\alpha}^*) = f(\boldsymbol{a}^*, \boldsymbol{\alpha}^*)$*. Furthermore, if there is a unique* $\boldsymbol{a}^*$ *among the minimizing pairs of* $f(\boldsymbol{a}, \boldsymbol{\alpha})$ *then* $\boldsymbol{a}^*$ *must satisfy* $\boldsymbol{a}^* = \Phi(\boldsymbol{\alpha}^*)^\dagger \boldsymbol{y}$*.*

8.5 Multiobjective optimization

This is a subject that is seldom treated in optimization books, although it arises frequently in financial optimization, decision and game theory and other areas. In recent times it has increasingly been recognized that many engineering problems are also of this kind. We have also already mentioned that most regularization procedures are actually bi-objective problems (see [240, 241] for an interesting theory and algorithm).

The basic unconstrained problem is

$$\min_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}),$$

where now $\boldsymbol{f} \in \mathbb{R}^k$ is a vector function. Such problems arise in areas of interest of this book, for instance, in cooperative fitting and inversion, when measurements of several physical processes on the same sample are used to determine properties that are interconnected.

In general, it will be unlikely that the k objective functions share a common minimum point, so the theory of multiobjective optimization is different from the single-objective optimization case. The objectives are usually in conflict and a compromise must be chosen. The optimality concept is here the so-called Pareto equilibrium: *a point* $\boldsymbol{x}$ *is a Pareto equilibrium point if there is no other point for which all functions have smaller values.*

This notion can be global (as stated) or local if it is restricted to a neighborhood of $\boldsymbol{x}$. In general there will be infinitely many such points. In the absence of additional information, each Pareto point would be equally acceptable. Let us now see if we can give a more geometric interpretation of this condition, based on some of the familiar concepts used earlier.

For simplicity let us consider the case $k = 2$. First of all we observe that the individual minimum points

$$\boldsymbol{x}_i^* = \arg\min_{\boldsymbol{x}} f_i(\boldsymbol{x}), \qquad i = 1, 2$$

are Pareto optimal. We consider the level sets corresponding to the two functions and observe that at a point at which these two level sets are tangent and their corresponding normals are in opposite directions, there will be no common directions of descent for both functions. But that means that there is no direction in which we can move so that both functions are improved, i.e., this is a Pareto equilibrium point. Analytically this is expressed as

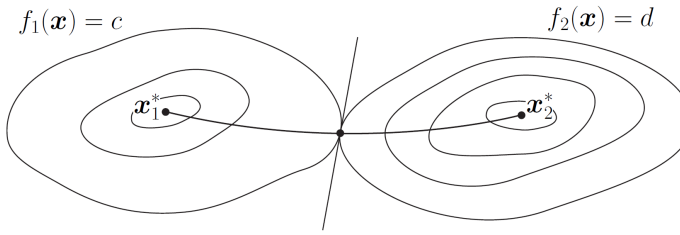


Figure 8.5.1: Illustration of a multiobjective optimization problem. The two functions $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ have minimizers $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$. The curve between these two points is the set of Pareto points.

$$\frac{\nabla f_1(\boldsymbol{x})}{\|\nabla f_1(\boldsymbol{x})\|_2} + \frac{\nabla f_2(\boldsymbol{x})}{\|\nabla f_2(\boldsymbol{x})\|_2} = \mathbf{0},$$

which can be rewritten as

$$\lambda \nabla f_1(\boldsymbol{x}) + (1-\lambda)\nabla f_2(\boldsymbol{x}) = \mathbf{0}, \qquad 0 \le \lambda \le 1.$$

It can be shown that for convex functions, all the Pareto points are parametrized by this aggregated formula, which coincides with the optimality condition for the scalar optimization problems

$$\min_{\boldsymbol{x}} \left[\lambda f_1(\boldsymbol{x}) + (1-\lambda) f_2(\boldsymbol{x})\right], \qquad 0 \le \lambda \le 1.$$

This furnishes a way to obtain the Pareto points by solving a number of scalar optimization problems. Figure 8.5.1 illustrates this; the curve between the two minimizers $\boldsymbol{x}_1^*$ and $\boldsymbol{x}_2^*$ is the set of Pareto points.

Curiously enough, these optimality conditions were already derived in the seminal paper on nonlinear programming by Kuhn and Tucker in 1951 [147].

A useful tool is given by the graph in phase space of the Pareto points, the so-called Pareto front (see 8.5.2.) In fact, this graph gives a complete picture of all possible solutions, and it is then usually straightforward to make a decision by choosing an appropriate trade-off between the objectives. Since one will usually be able to calculate only a limited number of Pareto points, it is important that the Pareto front be uniformly sampled. In [190, 194], a method based on continuation in λ with added distance constraints produces automatically equispaced representations of the Pareto front. In Figure 8.5.2 we show a uniformly sampled Pareto front for a bi-objective problem.

The ϵ-constraint method of Haimes [114] is frequently used to solve this type of problem in the case that a hierarchical order of the objectives is

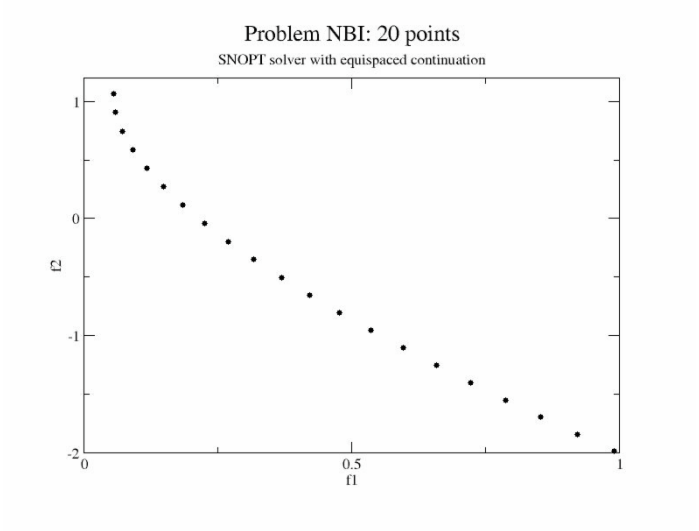


Figure 8.5.2: Evenly spaced Pareto front (in (f_1, f_2) space) for a bi-objective problem [190].

known and one can make an a priori call on a compromise upper value for the secondary objective. It transforms the bi-objective problem into a single-objective constrained minimization of the main goal. The constraint is the upper bound of the second objective. In other words, one minimizes the first objective subject to an acceptable level in the second objective (that better be larger than the minimum). From what we saw before, the resulting solution may be sub-optimal.

A good reference for the theoretical and practical aspects of nonlinear multiobjective optimization is [161].

Chapter 9

Algorithms for Solving Nonlinear LSQ Problems

The classical method of Newton and its variants can be used to solve the nonlinear least squares problem formulated in the previous chapter. Newton's method for optimization is based on a second-order Taylor approximation of the objective function $f(\boldsymbol{x})$ and subsequent minimization of the resulting approximate function. Alternatively, one can apply Newton's method to solve the nonlinear system of equations $\nabla f(\boldsymbol{x}) = \mathbf{0}$.

The Gauss-Newton method is a simplification of the latter approach for problems that are almost consistent or mildly nonlinear, and for which the second term in the Hessian $\nabla^2 f(\boldsymbol{x})$ can be safely neglected. The Levenberg-Marquardt method can be considered a variant of the Gauss-Newton method in which stabilization (in the form of Tikhonov regularization, cf. Section 10) is applied to the linearized steps in order to solve problems with ill-conditioned or rank-deficient Jacobian $J(\boldsymbol{x})$.

Due to the local character of the Taylor approximation one can only obtain local convergence, in general. In order to get global convergence, the methods need to be combined with a line search. Global convergence means convergence to a local minimum from any initial point. For convergence to a global minimum one needs to resort, for instance, to a Monte Carlo technique that provides multiple random initial points, or to other costlier methods [191].

We will describe first the different methods and then consider the common difficulties, such as how to start and end and how to ensure descent at each step.

9.1 Newton's method

If we assume that $f(\boldsymbol{x})$ is twice continuously differentiable, then we can use Newton's method to solve the system of nonlinear equations

$$\nabla f(\boldsymbol{x}) = J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{0},$$

which provides local stationary points for $f(\boldsymbol{x})$. Written in terms of derivatives of $\boldsymbol{r}(\boldsymbol{x})$ and starting from an initial guess $\boldsymbol{x}_0$ this version of the Newton iteration takes the form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \left(\nabla^2 f(\boldsymbol{x}_k)\right)^{-1} \nabla f(\boldsymbol{x}_k) \\ &= \boldsymbol{x}_k - \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right)^{-1} J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k), \\ & \quad k = 0, 1, 2, \ldots \end{aligned}$$

where $S(\boldsymbol{x}_k)$ denotes the matrix

$$S(\boldsymbol{x}_k) = \sum_{i=1}^{m} r_i(\boldsymbol{x}_k) \nabla^2 r_i(\boldsymbol{x}_k). \tag{9.1.1}$$

As usual, no inverse is calculated to obtain a new iterate, but rather a linear system is solved by a direct or iterative method to obtain the correction $\Delta \boldsymbol{x}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)\right) \Delta \boldsymbol{x}_k = -J^T(\boldsymbol{x}_k) \boldsymbol{r}(\boldsymbol{x}_k). \tag{9.1.2}$$

The method is locally quadratically convergent as long as $\nabla^2 f(\boldsymbol{x})$ is Lipschitz continuous and positive definite in a neighborhood of $\boldsymbol{x}^*$. This follows from a simple adaptation of the Newton convergence theorem in [66], which leads to the result

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leq \gamma \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2, \qquad k = 0, 1, 2, \ldots .$$

The constant γ is a measure of the nonlinearity of $\nabla f(\boldsymbol{x})$, it depends on the Lipschitz constant for $\nabla^2 f(\boldsymbol{x})$ and a bound for $\left\|\nabla^2 f(\mathbf{x}^*)^{-1}\right\|_2^2$, the size of the residual does not appear. The convergence rate will depend on the nonlinearity, but the convergence itself will not. The foregoing result implies that Newton's method is usually very fast in the final stages, close to the solution.

In practical applications Newton's iteration may not be performed exactly and theorems 4.22 and 4.30 in [143, 183] give convergence results when the errors are controlled appropriately.

Note that the Newton correction $\Delta \boldsymbol{x}_k$ will be a descent direction (as explained in the previous chapter), as long as $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) +$

$S(\boldsymbol{x}_k)$ is positive definite. In fact, using the definition of positive definite matrices one obtains, by multiplying both sides of (9.1.2)

$$0 < \Delta\boldsymbol{x}_k^T \left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k) \right) \Delta\boldsymbol{x}_k = -\Delta\boldsymbol{x}_k^T J^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

Therefore, the correction $\Delta\boldsymbol{x}_k$ is in the same half-space as the steepest descent direction $-J^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k)$. Although the Newton correction is a descent direction, the step size may be too large, since the linear model is only locally valid. Therefore to ensure convergence, Newton's method is used with step-length control to produce a more robust algorithm.

One of the reasons why Newton's method is not used more frequently for nonlinear least squares problem is that its good convergence properties come at a price: the mn^2 derivatives appearing in $S(\boldsymbol{x}_k)$ must be computed! This can be expensive and often the derivatives are not even available and thus must be substituted by finite differences. Special cases where Newton's method is a good option are when $S(\boldsymbol{x}_k)$ is sparse, which happens frequently if $J(\boldsymbol{x}_k)$ is sparse or when $S(\boldsymbol{x}_k)$ involves exponentials or trigonometric functions that are easy to compute. Also, if one has access to the code that calculates the model, automatic differentiation can be used [111].

If the second derivatives term in $\nabla^2 f(\boldsymbol{x}_k) = J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + S(\boldsymbol{x}_k)$ is unavailable or too expensive to compute and hence approximated, the resulting algorithm is called a quasi-Newton method and although the convergence will no longer be quadratic, superlinear convergence can be attained. It is important to point out that only the second term of $\nabla^2 f(\boldsymbol{x}_k)$ needs to be approximated since the first term $J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)$ has already been computed. A successful strategy is to approximate $S(\boldsymbol{x}_k)$ by a secant-type term, using updated gradient evaluations:

$$S(\boldsymbol{x}_k) \simeq \sum_{i=1}^{m} r_i(\boldsymbol{x}_k)\widetilde{G}_i(\boldsymbol{x}_k),$$

where the Hessian terms $\nabla^2 r_i(\boldsymbol{x}_k)$ are approximated from the condition

$$\widetilde{G}_i(\boldsymbol{x}_k)\,(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) = \nabla r_i(\boldsymbol{x}_k) - \nabla r_i(\boldsymbol{x}_{k-1}).$$

This condition would determine a secant approximation for $n = 1$, but in the higher-dimensional case it must be complemented with additional requirements on the matrix $\widetilde{G}_i(\boldsymbol{x}_k)$: it must be symmetric and it must satisfy the so-called least-change secant condition, i.e., that it be most similar to the approximation $\widetilde{G}_i(\boldsymbol{x}_{k-1})$ in the previous step. In [67], local superlinear convergence is proved, under the assumptions of Lipschitz continuity and bounded inverse of $\nabla^2 f(\boldsymbol{x})$.

9.2 The Gauss-Newton method

If the problem is only mildly nonlinear or if the residual at the solution (and therefore in a reasonable-sized neighborhood) is small, a good alternative to Newton's method is to neglect the second term $S(\boldsymbol{x}_k)$ of the Hessian altogether. The resulting method is referred to as the Gauss-Newton method, where the computation of the step $\Delta\boldsymbol{x}_k$ involves the solution of the linear system

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\right) \Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k) \tag{9.2.1}$$

and $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$.

Note that in the full-rank case these are actually the normal equations for the linear least squares problem

$$\min_{\Delta\boldsymbol{x}_{k+1}} \|J(\boldsymbol{x}_k)\,\Delta\boldsymbol{x}_k - (-\boldsymbol{r}(\boldsymbol{x}_k))\|_2 \tag{9.2.2}$$

and thus $\Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^\dagger \boldsymbol{r}(\boldsymbol{x}_k)$. By the same argument as used for the Newton method, this is a descent direction if $J(\boldsymbol{x}_k)$ has full rank.

We note that the linear least squares problem in (9.2.2), which defines the Gauss-Newton direction $\Delta\boldsymbol{x}_k$, can also be derived from the principle of local approximation discussed in the previous chapter. When we linearize the residual vector $\boldsymbol{r}(\boldsymbol{x}_k)$ in the kth step we obtain the approximation in (8.2.3) with $\boldsymbol{x} = \boldsymbol{x}_k$, which is identical to (9.2.2).

The convergence properties of the Gauss-Newton method can be summarized as follows [66] (see also [184] for an early proof of convergence):

Theorem 91. *Assume that*

- *$f(\boldsymbol{x})$ is twice continuously differentiable in an open convex set $\mathcal{D}$.*
- *$J(\boldsymbol{x})$ is Lipschitz continuous with constant γ, and $\|J(\boldsymbol{x})\|_2 \leqslant \alpha$.*
- *There is a stationary point $\boldsymbol{x}^* \in \mathcal{D}$, and*
- *for all $\boldsymbol{x} \in \mathcal{D}$ there exists a constant σ such that*

$$\left\|(J(\boldsymbol{x}) - J(\boldsymbol{x}^*))^T \boldsymbol{r}(\boldsymbol{x}^*)\right\|_2 \leqslant \sigma\|\boldsymbol{x} - \boldsymbol{x}^*\|_2.$$

If σ is smaller than λ, the smallest eigenvalue of $J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)$, then for any $c \in (1, \lambda/\sigma)$ there exists a neighborhood so that the Gauss-Newton sequence converges linearly to $\boldsymbol{x}^*$ starting from any initial point $\boldsymbol{x}_0$ in $\mathcal{D}$:

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c\sigma}{\lambda}\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 + \frac{c\alpha\gamma}{2\lambda}\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2$$

and

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c\sigma + \lambda}{2\lambda}\|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 < \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 .$$

If the problem is consistent, i.e., $\boldsymbol{r}(\boldsymbol{x}^*) = \mathbf{0}$, there exists a (maybe smaller) neighborhood where the convergence will be quadratic.

The Gauss-Newton method can produce the whole spectrum of convergence: if $S(\boldsymbol{x}_k) = 0$, or is very small, the convergence can be quadratic, but if $S(\boldsymbol{x}_k)$ is too large there may not even be local convergence. The important constant is σ, which can be considered as an approximation of $\|S(\boldsymbol{x}^*)\|_2$ since, for $\boldsymbol{x}$ sufficiently close to $\boldsymbol{x}^*$, it can be shown that

$$\left\|(J(\boldsymbol{x}) - J(\boldsymbol{x}^*))^T \boldsymbol{r}(\boldsymbol{x}^*)\right\|_2 \simeq \|S(\boldsymbol{x}^*)\|_2 \, \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 \, .$$

The ratio σ/λ must be less than 1 for convergence. The rate of convergence is inversely proportional to the "size" of the nonlinearity or the residual, i.e., the larger $\|S(\boldsymbol{x}_k)\|_2$ is in comparison to $\|J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\|_2$, the slower the convergence.

As we saw above, the Gauss-Newton method produces a direction $\Delta\boldsymbol{x}_k$ that is of descent, but due to the local character of the underlying approximation, the step length may be incorrect, i.e., $f(x_k + \Delta\boldsymbol{x}_k) > f(\boldsymbol{x}_k)$. This suggests a way to improve the algorithm, namely, to use damping, which is a simple mechanism that controls the step length to ensure a sufficient reduction of the function. An alternative is to use a trust-region strategy to define the direction; this leads to the Levenberg-Marquardt algorithm discussed later on.

Algorithm damped Gauss-Newton

- Start with an initial guess $\boldsymbol{x}_0$ and iterate for $k = 0, 1, 2, \ldots$
- Solve $\min_{\Delta\boldsymbol{x}_k} \|J(\boldsymbol{x}_k)\, \Delta\boldsymbol{x}_k + \boldsymbol{r}(\boldsymbol{x}_k)\|_2$ to compute the correction $\Delta\boldsymbol{x}_k$.
- Choose a step length α_k so that there is enough descent.
- Calculate the new iterate: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \Delta\boldsymbol{x}_k$.
- Check for convergence.

Several methods for choosing the step-length parameter α_k have been proposed and the key idea is to ensure descent by the correction step $\alpha_k \Delta\boldsymbol{x}_k$. One popular choice is the Armijo condition (see Section 9.4), which uses a constant $\alpha_k \in (0, 1)$ to ensure enough descent in the value of the objective:

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha_k \Delta\boldsymbol{x}_k) &< f(\boldsymbol{x}_k) + \alpha_k \nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k \\ &= f(\boldsymbol{x}_k) + \alpha_k \boldsymbol{r}(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k. \end{aligned} \qquad (9.2.3)$$

This condition ensures that the reduction is (at least) proportional to both the parameter α_k and the directional derivative $\nabla f(\boldsymbol{x}_k)^T \Delta\boldsymbol{x}_k$.

Using the two properties above, a descent direction and an appropriate step length, the damped Gauss-Newton method is locally convergent

y	x^*	$f(x^*)$	$J(x^*)^T J(x^*)$	$S(x^*)$
8	0.6932	0	644.00	0
3	0.4401	1.639	151.83	9.0527
-1	0.0447	6.977	17.6492	8.3527
-8	-0.7915	41.145	0.4520	2.9605

Table 9.1: Data for the test problem for four values of the scalar y. Note that x^*, $J(x^*)$ and $S(x^*)$ are scalars in this example.

and often globally convergent. Still, the convergence rate may be slow for problems for which the standard algorithm is inadequate. Also, the inefficiency of the Gauss-Newton method when applied to problems with ill-conditioned or rank-deficient Jacobian has not been addressed; the next algorithm, Levenberg-Marquardt, will consider this issue.

Example 92. *The following example from [66] will clarify the above convergence results. We fit the one-parameter model $M(x,t) = e^{xt}$ to the data set*

$$(t_1, y_1) = (1, 2), \qquad (t_2, y_2) = (2, 4), \qquad (t_3, y_3) = (3, y),$$

where y can take one of the four values $8, 3, -1, -8$. The least squares problem is, for every y, to determine the single parameter x, to minimize the function

$$f(x) = \tfrac{1}{2} \sum_{i=1}^{3} r_i(x)^2 = \tfrac{1}{2} \left[(e^x - 2)^2 + (e^{2x} - 4)^2 + (e^{3x} - y)^2 \right].$$

For this function of a single variable x, both the simplified and full Hessian are scalars:

$$J(x)^T J(x) = \sum_{i=1}^{3} (t_i e^{xt_i})^2 \qquad \text{and} \qquad S(x) = \sum_{i=1}^{3} r_i(x) t_i^2 e^{xt_i}.$$

Table 9.1 lists, for each of the four values of y, the minimizer x^* and values of several functions at the minima.

We use the Newton and Gauss-Newton methods with several starting values x_0. Table 9.2 lists the different convergence behaviors for every case, with two different starting values. The stopping criterion for the iterations was $|\nabla f(x_k)| \leq 10^{-10}$. Note that for the consistent problem with $y = 8$, Gauss-Newton achieves quadratic convergence, since $S(x^*) = 0$. For $y = 3$, the convergence factor for Gauss-Newton is $\sigma/\lambda \simeq 0.06$, which is small compared to 0.47 for $y = -1$, although for this value of y there is still linear but slow convergence. For $y = -8$ the ratio is $6.5 \gg 1$ and there is no convergence.

		Newton		Gauss-Newton	
y	x_0	# iter.	rate	# iter.	rate
8	1	7	quadratic	5	quadratic
	0.6	6	quadratic	4	quadratic
3	1	9	quadratic	12	linear
	0.5	5	quadratic	9	linear
-1	1	10	quadratic	34	linear (slow)
	0	4	quadratic	32	linear (slow)
-8	1	12	quadratic	no convergence	
	-0.7	4	quadratic	no convergence	

Table 9.2: Convergence behavior for the four cases and two different starting values.

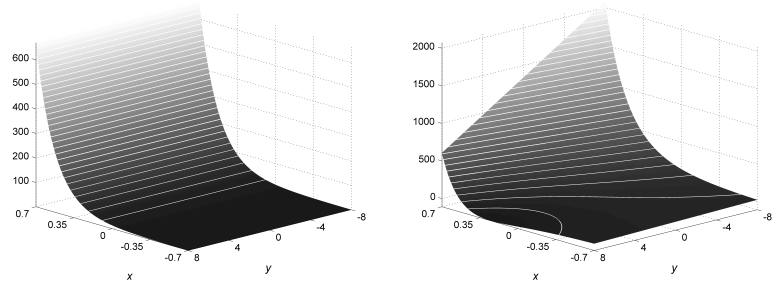


Figure 9.2.1: Single-parameter exponential fit example: the scalars $J(x)^T J(x)$ (left) and $\nabla^2 f(x)$ (right) as functions of x and y (the white lines are level curves). The second term in $\nabla^2 f(x)$ has increasing importance as y decreases.

To support the above results, Figure 9.2.1 shows plots of the Hessian $\nabla^2 f(x)$ and its first term $J(x)^T J(x)$ (recall that both are scalars) as functions of x and y, for $x \in [-0.7, 0.7]$ and $y \in [-8, 8]$. We see that the deterioration of the convergence rate of Gauss-Newton's method, compared to that of Newton's method, indeed coincides with the increasing importance of the term $S(x)$ in $\nabla^2 f(x)$, due to larger residuals at the solution as y decreases.

For large-scale problems, where memory and computing time are limiting factors, it may be infeasible to solve the linear LSQ problem for the correction $\Delta \boldsymbol{x}_k$ to high accuracy in each iteration. Instead one can use one of the iterative methods discussed in Chapter 6 to compute an approximate step $\triangle \tilde{\boldsymbol{x}}_k$, by terminating the iterations once the approximation is "accu-

rate enough," e.g., when the residual in the normal equations is sufficiently small:

$$\|J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)\,\Delta\tilde{\boldsymbol{x}}_k + J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\|_2 < \tau \|J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\|_2,$$

for some $\tau \in (0,1)$. In this case, the algorithm is referred to as an *inexact Gauss-Newton method*. For these nested iterations an approximate Newton method theory applies [61, 143, 183].

9.3 The Levenberg-Marquardt method

While the Gauss-Newton method can be used for ill-conditioned problems, it is not efficient, as one can see from the above convergence relations when $\lambda \to 0$. The main difficulty is that the correction $\Delta\boldsymbol{x}_k$ is too large and goes in a "bad" direction that gives little reduction in the function. For such problems, it is common to add an inequality constraint to the linear least squares problem (9.2.2) that determines the step, namely, that the length of the step $\|\Delta\boldsymbol{x}_k\|_2$ should be bounded by some constant. This so-called trust region technique improves the quality of the step.

As we saw in the previous chapter, we can handle such an inequality constraint via the use of a Lagrange multiplier and thus replace the problem in (9.2.2) with a problem of the form

$$\min_{\Delta\boldsymbol{x}_{k+1}} \left\{ \|J(\boldsymbol{x}_k)\,\Delta\boldsymbol{x}_k + \boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 + \lambda_k \|\boldsymbol{x}_k\|_2^2 \right\},$$

where $\lambda_k > 0$ is the Lagrange multiplier for the constraint at the kth iteration. There are two equivalent forms of this problem, either the "modified" normal equations

$$\left(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \lambda_k I\right) \Delta\boldsymbol{x}_k = -J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k) \tag{9.3.1}$$

or the "modified" least squares problem

$$\min_{\Delta\boldsymbol{x}_k} \left\| \begin{pmatrix} J(\boldsymbol{x}_k) \\ \sqrt{\lambda_k} I \end{pmatrix} \Delta\boldsymbol{x}_k - \begin{pmatrix} -\boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{0} \end{pmatrix} \right\|_2 . \tag{9.3.2}$$

The latter is best suited for numerical computations. This approach, which also handles a rank-deficient Jacobian $J(\boldsymbol{x}_k)$, leads to the Levenberg-Marquardt method, which takes the following form:

Algorithm Levenberg-Marquardt

- Start with an initial guess $\boldsymbol{x}_0$ and iterate for $k = 0, 1, 2, \ldots$
- At each step k choose the Lagrange multiplier λ_k.

- Solve 9.3.1 or 9.3.2 for $\Delta \boldsymbol{x}_k$.
- Calculate the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta \boldsymbol{x}_k$.
- Check for convergence.

The parameter λ_k influences both the direction and the length of the step $\Delta \boldsymbol{x}_k$. Depending on the size of λ_k, the correction $\Delta \boldsymbol{x}_k$ can vary from a Gauss-Newton step for $\lambda_k = 0$, to a very short step approximately in the steepest descent direction for large values of λ_k. As we see from these considerations, the LM parameter acts similarly to the step control for the damped Gauss-Newton method, but it also changes the direction of the correction.

The Levenberg-Marquardt step can be interpreted as solving the normal equations used in the Gauss-Newton method, but "shifted" by a scaled identity matrix, so as to convert the problem from having an ill-conditioned (or positive semidefinite) matrix $J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k)$ into a positive definite one. Notice that the positive definiteness implies that the Levenberg-Marquardt direction is always of descent and therefore the method is well defined.

Another way of looking at the Levenberg-Marquardt iteration is to consider the matrix $\sqrt{\lambda_k} I$ as an approximation to the second derivative term $S(\boldsymbol{x}_k)$ that was neglected in the definition of the Gauss-Newton method.

The local convergence of the Levenberg-Marquardt method is summarized in the following theorem.

Theorem 93. *Assume the same conditions as in Theorem 91 and in addition assume that the Lagrange multipliers λ_k for $k = 0, 1, 2, \ldots$ are nonnegative and bounded by $b > 0$. If $\sigma < \lambda$, then for any $c \in (1, (\lambda+b)/(\sigma+b))$, there exists an open ball D around $\boldsymbol{x}^*$ such that the Levenberg-Marquardt sequence converges linearly to $\boldsymbol{x}^*$, starting from any initial point $\boldsymbol{x}_0 \in \mathcal{D}$:*

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c(\sigma + b)}{\lambda + b} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 + \frac{c\alpha\gamma}{2(\lambda + b)} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2^2$$

and

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|_2 \leqslant \frac{c(\sigma + b) + (\lambda + b)}{2(\lambda + b)} \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 < \|\boldsymbol{x}_k - \boldsymbol{x}^*\|_2 .$$

If $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\lambda_k = O\left(\left\|J(\boldsymbol{x}_k)^T \boldsymbol{r}(\boldsymbol{x}_k)\right\|_2\right)$, then the iterates $\boldsymbol{x}_k$ converge quadratically to $\boldsymbol{x}^*$.

Within the trust-region framework introduced above, there are several general step-length determination techniques, see, for example, [170]. Here we give the original strategy devised by Marquardt for the choice of the parameter λ_k, which is simple and often works well. The underlying principles are

- The initial value λ_0 should be of the same size as $\|J(\boldsymbol{x}_0)^T J(\boldsymbol{x}_0)\|_2$.
- For subsequent steps, an improvement ratio is defined as in the trust region approach:

$$\varrho_k = \frac{\text{actual reduction}}{\text{predicted reduction}} = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})}{\frac{1}{2}\Delta\boldsymbol{x}_k^T(J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k) - \lambda_k\Delta\mathbf{x}_k)}.$$

Here, the denominator is the reduction in f predicted by the local linear model. If ϱ_k is large, then the pure Gauss-Newton model is good enough, so λ_{k+1} can be made smaller than at the previous step. If ϱ_k is small (or even negative), then a short, steepest-descent step should be used, i.e., λ_{k+1} should be increased. Marquard's updating strategy is widely used with some variations in the thresholds.

Algorithm Levenberg-Marquardt's parameter updating

- If $\rho_k > 0.75$, then $\lambda_{k+1} = \lambda_k/3$.
- If $\rho_k < 0.25$, then $\lambda_k = 2\,\lambda_k$.
- Otherwise use $\lambda_{k+1} = \lambda_k$.
- If $\rho_k > 0$, then perform the update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$.

A detailed description can be found in [154].

The software package MINPACK-1 available from Netlib [263] includes a robust implementation based on the Levenberg-Marquardt algorithm.

Similar to the Gauss-Newton method, we have inexact versions of the Levenberg-Marquardt method, where the modified least squares problem (9.3.2) for the correction is solved only to sufficient accuracy, i.e., for some $\tau \in (0, 1)$ we accept the solution if:

$$\|(J(\boldsymbol{x}_k)^T J(\boldsymbol{x}_k) + \lambda_k I)\,\Delta\boldsymbol{x}_k + J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k)\|_2 < \tau\|J(\boldsymbol{x}_k)^T\boldsymbol{r}(\boldsymbol{x}_k)\|_2.$$

If this system is to be solved for several values of the Lagrange parameter λ_k, then the bidiagonalization strategy from Section 5.5 can be utilized such that only one partial bidiagonalization needs to be performed; for more details see [121, 260].

Example 94. *We return to Example 92, and this time we use the Levenberg-Marquardt method with the above parameter-updating algorithm and the same starting values and stopping criterion as before. Table 9.3 compares the performance of Levenberg-Marquardt's algorithm with that of the Gauss-Newton algorithm. For the consistent problem with $y = 8$, the convergence is still quadratic, but slower. As the residual increases, the advantage of the Levenberg-Marquardt strategy sets in, and we are now able to solve the large-residual problem for $y = -8$, although the convergence is very slow.*

		Gauss-Newton		Levenberg-Marquardt	
y	x_0	# iter.	rate	# iter.	rate
8	1	5	quadratic	10	quadratic
	0.6	4	quadratic	7	quadratic
3	1	12	linear	13	linear
	0.5	9	linear	10	linear
-1	1	34	linear (slow)	26	linear (slow)
	0	32	linear (slow)	24	linear (slow)
-8	1	no convergence		125	linear (very slow)
	-0.7	no convergence		120	linear (very slow)

Table 9.3: Comparison of the convergence behavior of the Gauss-Newton and Levenberg-Marquardt methods for the same test problem as in Table 9.2.

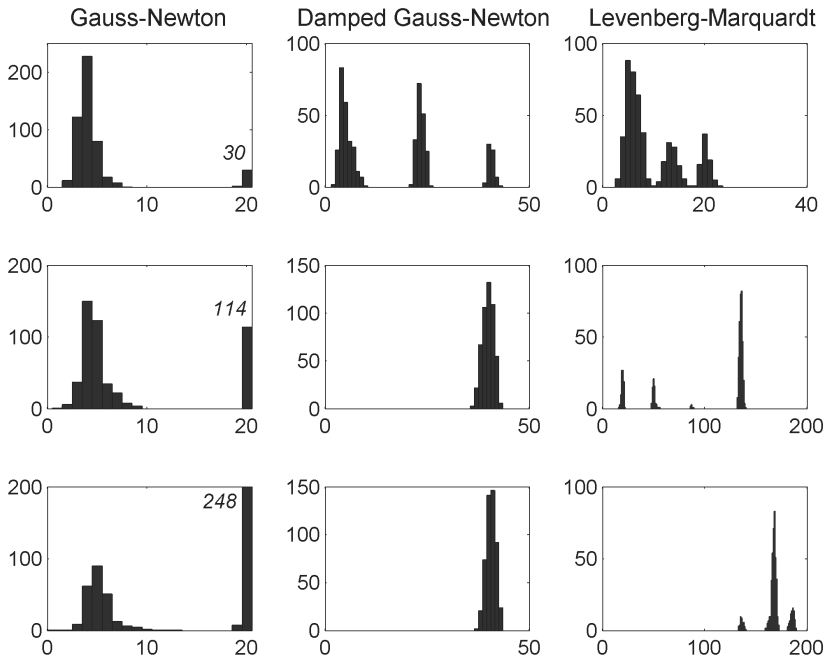


Figure 9.3.1: Number of iterations to reach an absolute accuracy of 10^{-6} in the solution by three NLLSQ different methods. Each histogram shows results for a particular method and a particular value of ζ in the perturbation of the starting point; top $\zeta = 0.1$, middle $\zeta = 0.2$, bottom $\zeta = 0.3$.

Example 95. *This example is mainly concerned with a comparison of the robustness of the iterative methods with regards to the initial guess* $\boldsymbol{x}_0$*. We use the Gaussian model from Examples 83 and 86 with noise level* $\varsigma = 0.1$*, and we use three different algorithms to solve the NLLSQ problem:*

- *The standard Gauss-Newton method.*
- *An implementation of the damped Gauss-Newton algorithm from MATLAB's Optimization Toolbox, available via the options* `'LargeScale'='off'` *and* `'LevenbergMarquardt'='off'` *in the function* `lsqnonlin`*.*
- *An implementation of the Levenberg-Marquardt algorithm from MATLAB's Optimization Toolbox, available via the options* `'Jacobian'='on'` *and* `'Algorithm'='levenberg-marquardt'` *in the function* `lsqnonlin`*.*

The starting guess was chosen equal to the exact parameters $(2.2,\ 0.26,\ 0.2)^T$ *plus a Gaussian perturbation with standard deviation* ζ*. We used* $\zeta = 0.1$*,* 0.2*,* 0.3*, and for each value we created 500 different realizations of the initial point. Figure 9.3.1 shows the number of iterations in the form of histograms – notice the different axes in the nine plots. We give the number of iterations necessary to reach an absolute accuracy of* 10^{-6}*, compared to a reference solution computed with much higher accuracy. The italic numbers in the three left plots are the number of times the Gauss-Newton method did not converge.*

We see that when the standard Gauss-Newton method converges, it converges rapidly – but also that it may not converge. Moreover, as the starting point moves farther from the minimizer, the number of instances of non-convergence increases dramatically.

The damped Gauss-Newton method used here always converged, thus was much more robust than the undamped version. For starting points close to the minimum, it may converge quickly, but it may also require about 40 iterations. For starting points further away from the solution, this method always uses about 40 iterations.

The Levenberg-Marquardt method used here is also robust, in that it converges for all starting points. In fact, for starting points close to the solution it requires, on the average, fewer iterations than the damped Gauss-Newton method. For starting points farther from the solution it still converges, but requires many more iterations. The main advantage of the Levenberg-Marquardt algorithm, namely, to handle ill-conditioned and rank-deficient Jacobian matrices, does not come into play here as the particular problem is well conditioned.

We emphasize that this example should not be considered as representative for these methods in general – rather, it is an example of the perfor-

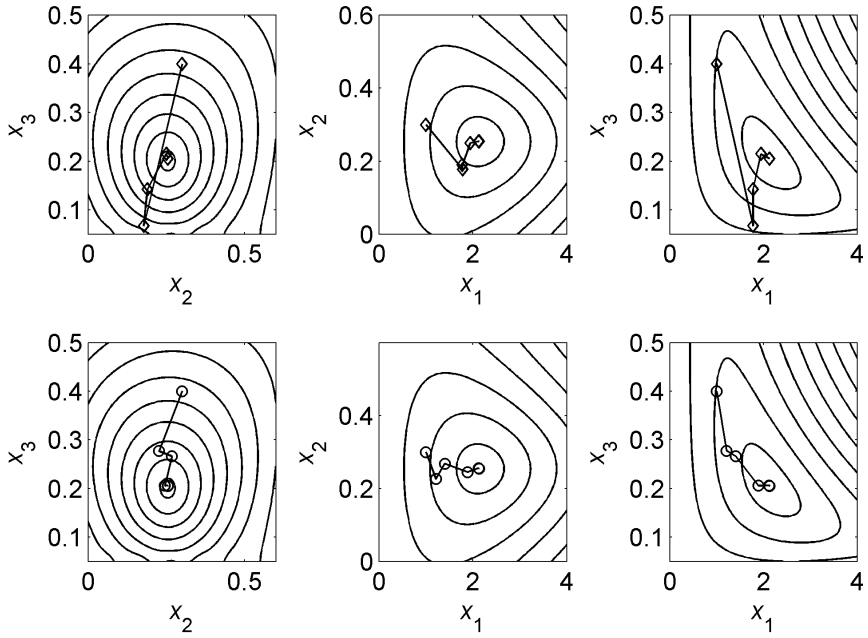


Figure 9.3.2: Convergence histories from the solution of a particular NLLSQ problem with three parameters by means of the standard Gauss-Newton (top) and the Levenberg-Marquardt algorithm (bottom). Each figure shows the behavior of pairs of the three components of the iterates $\boldsymbol{x}_k$ together with level sets of the objective function.

mance of specific implementations for one particular (and well-conditioned) problem.

Example 96. *We illustrate the robustness of the Levenberg-Marquardt algorithm by solving the same test problem as in the previous example by means of the standard Gauss-Newton and Levenberg-Marquardt algorithms. The progress of a typical convergence history for these two methods is shown in Figure 9.3.2. There are three unknowns in the model, and the starting point is* $\boldsymbol{x}_0 = (1, 0.3, 0.4)^T$. *The top figures show different component pairs of the iterates* $\boldsymbol{x}_k$ *for* $k = 1, 2, \ldots$ *for the Gauss-Newton algorithm (for example, the leftmost plot shows the third component of* $\boldsymbol{x}_k$ *versus its second component). Clearly, the Gauss-Newton iterates overshoot before they finally converge to the LSQ solution. The iterates of the Levenberg-Marquardt algorithm are shown in the bottom figures, and we see that they converge much faster toward the LSQ solution without any big "detour."*

Characteristics	Newton	G-N	L-M
Ill-conditioned Jacobian	yes	yes (but slow)	yes
Rank-deficient Jacobian	yes	no	yes
Convergence $S(\boldsymbol{x}_k) = 0$	quadratic	quadratic	quadratic
Convergence $S(\boldsymbol{x}_k)$ small	quadratic	linear	linear
Convergence $S(\boldsymbol{x}_k)$ large	quadratic	slow or none	slow or none

Table 9.4: Comparison of some properties of the Newton, Gauss-Newton (G-N) and Levenberg-Marquardt (L-M) algorithms for nonlinear least squares problems.

9.4 Additional considerations and software

An overall comparison between the methods discussed so far is given in Table 9.4. The "yes" or "no" indicates whether the corresponding algorithm is appropriate or not for the particular problem. Below we comment on some further issues that are relevant for these methods.

Hybrid methods

During the iterations we do not know whether we are in the region where the convergence conditions of a particular method hold, so the idea in hybrid methods is to combine a "fast" method, such as Newton (if second derivatives are available), with a "safe" method such as steepest descent. One hybrid strategy is to combine Gauss-Newton or Levenberg-Marquardt, which in general is only linearly convergent, with a superlinearly convergent quasi-Newton method, where $S(\boldsymbol{x}_k)$ is approximated by a secant term.

An example of this efficient hybrid method is the algorithm NL2SOL by Dennis, Gay and Welsch [65]. It contains a trust-region strategy for global convergence and uses Gauss-Newton and Levenberg-Marquardt steps initially, until it has enough good second-order information. Its performance for large and very nonlinear problems is somewhat better than Levenberg-Marquardt, in that it requires fewer iterations. For more details see [66].

Starting and stopping

In many cases there will be no good a priori estimates available for the initial point x_0. As mentioned in the previous section, nonlinear least squares problems are usually non-convex and may have several local minima. Therefore some global optimization technique must be used to descend from undesired local minima if the global minimum is required. One possibility is a simple Monte Carlo strategy, in which multiple initial points are chosen at random and the least squares algorithm is started several times. It is

hoped that some of these iterations will converge, and one can then choose the best solution. See, e.g., [131] for an overview of global optimization algorithms.

In order to make such a process more efficient, especially when function evaluations are costly, a procedure has been described in [191] that saves all iterates and confidence radiuses. An iteration is stopped if an iterate lands in a previously calculated iterate's confidence sphere. The assumption underlying this decision is that the iteration will lead to a minimum already calculated. The algorithm is trivially parallelizable and runs very efficiently in a distributed network. Up to 40% savings have been observed from using the early termination strategy. This simple approach is easily parallelizable, but it can only be applied to low-dimensional problems.

Several criteria ought to be taken into consideration for stopping the iterative methods described above.

- The sequence convergence criterion: $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 \leq$ tolerance (not a particularly good one)!
- The consistency criterion: $|f(\boldsymbol{x}_{k+1})| \leq$ tolerance (relevant only for problems with $\boldsymbol{r}(\boldsymbol{x}^*) = \boldsymbol{0}$).
- The absolute function criterion: $\|\nabla f(\boldsymbol{x}_{k+1})\|_2 \leq$ tolerance.
- The maximum iteration count criterion: $k > k_{\max}$.

The absolute function criterion has a special interpretation for NLLSQ problems, namely, that the residual at $\boldsymbol{x}_{k+1}$ is nearly orthogonal to the subspace generated by the columns of the Jacobian. For the Gauss-Newton and Levenberg-Marquardt algorithms, the necessary information to check near-orthogonality is easily available.

Methods for step-length determination

Control of the step length is important for ensuring a robust algorithm that converges from starting points far from the minimum. Given an iterate $\boldsymbol{x}_k$ and a descent direction $\boldsymbol{p}$, there are two common ways to choose the step-length α_k.

- Take α_k as the solution to the one-dimensional minimization problem

$$\min_\alpha \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p})\|_2.$$

  This is expensive if one tries to find the exact solution. Fortunately, this is not necessary, and so-called soft or inexact line search strategies can be used.

- Inexact line searches: an α_k is accepted if a sufficient descent condition is satisfied for the new point $\boldsymbol{x}_k + \alpha_k \boldsymbol{p}$ and if α_k is large enough that there is a gain in the step. Use the so-called Armijo-Goldstein step-length principle, where α_k is chosen as the largest number from the sequence $\alpha = 1, \frac{1}{2}, \frac{1}{4}, \ldots$ for which the inequality

$$\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2 - \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p})\|_2 \geq \tfrac{1}{2}\alpha_k \|J(\boldsymbol{x}_k)\,\boldsymbol{p}\|_2$$

  holds.

For details see, for example, chapter 3 in [170] and for a brief overview see [20] p. 344.

Software

In addition to the software already mentioned, the PORT library, available from AT&T Bell Labs and partly from Netlib [263], has a range of codes for nonlinear least squares problems, some requiring the Jacobian and others that need only information about the objective function. The Gauss-Newton algorithm is not robust enough to be used on its own, but most of the software for nonlinear least squares that can be found in NAG, MATLAB and TENSOLVE (using tensor methods), have enhanced Gauss-Newton codes. MINPACK-1, MATLAB and IMSL contain Levenberg-Marquardt algorithms. The NEOS Guide on the Web [264] is a source of information about optimization in general, with a section on NLLSQ. Also, the National Institute of Standards and Technology Web page at [265] is very useful, as well as the book [165].

Some software packages for large-scale problems with a sparse Jacobian are VE10 and LANCELOT. VE10 [264], developed by P. Toint, implements a line search method, where the search direction is obtained by a truncated conjugate gradient technique. It uses an inexact Newton method for partially separable nonlinear problems with sparse structure. LANCELOT [148] is a software package for nonlinear optimization developed by A. Conn, N. Gould and P. Toint. The algorithm combines a trust-region approach, adapted to handle possible bound constraints and projected gradient techniques. In addition it has preconditioning and scaling provisions.

9.5 Iteratively reweighted LSQ algorithms for robust data fitting problems

In Chapter 1 we introduced the robust data fitting problem based on the principle of M-estimation, leading to the nonlinear minimization problem $\min_{\boldsymbol{x}} \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right)$ where the function ρ is chosen so that it gives less

weight to residuals $r_i(\boldsymbol{x})$ with large absolute value. Here we consider the important case of robust linear data fitting, where the residuals are the elements of the residual vector $\boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}$. Below we describe several iterative algorithms that, in spite of their differences, are commonly referred to as *iteratively reweighted least squares algorithms* due to their use of a sequence of linear least squares problems with weights that change during the iterations.

To derive the algorithms we consider the problem of minimizing the function

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right), \qquad \boldsymbol{r} = \boldsymbol{b} - A\,\boldsymbol{x}.$$

We introduce the vector $\boldsymbol{g} \in \mathbb{R}^m$ and the diagonal matrix $D \in \mathbb{R}^{m\times m}$ with elements defined by

$$g_i = \rho'(r_i), \qquad d_{ii} = \rho''(r_i), \qquad i = 1, \ldots, m, \tag{9.5.1}$$

where ρ' and ρ'' denote the first and second derivatives of $\rho(r)$ with respect to r. Then the gradient and the Hessian of f are given by

$$\nabla f(\boldsymbol{x}) = -A^T \boldsymbol{g}(\boldsymbol{x}) \qquad \text{and} \qquad \nabla^2 f(\boldsymbol{x}) = A^T D(\boldsymbol{x})\, A, \tag{9.5.2}$$

where we use the notation $\boldsymbol{g}(\boldsymbol{x})$ and $D(\boldsymbol{x})$ to emphasize that these quantities depend on $\boldsymbol{x}$.

The original iteratively reweighted least squares algorithm [8] uses a fixed-point iteration to solve $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$. From (9.5.1) and (9.5.2) and introducing the diagonal $m \times m$ weight matrix

$$W(\boldsymbol{r}) = \operatorname{diag}\left(\rho'(r_i)/r_i\right) \qquad \text{for} \qquad i = 1, \ldots, m, \tag{9.5.3}$$

we obtain the nonlinear equations

$$A^T \boldsymbol{g}(\boldsymbol{x}) = A^T W(\boldsymbol{r})\, \boldsymbol{r} = A^T W(\boldsymbol{r})\, (\boldsymbol{b} - A\,\boldsymbol{x}) = \boldsymbol{0}$$

or

$$A^T W(\boldsymbol{r})\, A\,\boldsymbol{x} = A^T W(\boldsymbol{r})\, \boldsymbol{b}.$$

The fixed-point iteration scheme used in [8] is

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \left(A^T W(\boldsymbol{r}_k)\, A\right)^{-1} A^T W(\boldsymbol{r}_k)\, \boldsymbol{b} \\ &= \operatorname{argmin}_{\boldsymbol{x}} \| W(\boldsymbol{r}_k)^{1/2} \left(\boldsymbol{b} - A\,\boldsymbol{x}_k\right) \|_2, \quad k = 1, 2, \ldots \end{aligned} \tag{9.5.4}$$

where $\boldsymbol{r}_k = \boldsymbol{b} - A\,\boldsymbol{x}_k$ is the residual from the previous iteration. Hence, each new iterate is the solution of a weighted least squares problem, in which the weights depend on the solution from the previous iteration. This algorithm, with seven different choices of the function ρ, is implemented

in a Fortran software package described in [50] where more details can be found.

A faster algorithm for solving the robust linear data fitting problem – also commonly referred to as an iteratively reweighted least squares algorithm – is obtained by applying Newton's method from Section 9.1 to the function $f(\boldsymbol{x}) = \sum_{i=1}^{m} \rho\left(r_i(\boldsymbol{x})\right)$; see ([20], section 4.5.3) for details. According to (9.5.2), the Newton iteration is

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left(A^T D(\boldsymbol{x}_k)\, A\right)^{-1} A^T \boldsymbol{g}(\boldsymbol{x}_k), \qquad k = 1, 2, \ldots . \tag{9.5.5}$$

We emphasize that the Newton update is not a least squares solution, because the diagonal matrix $D(\boldsymbol{x}_k)$ appears in front of the vector $\boldsymbol{g}(\boldsymbol{x}_k)$. O'Leary [171] suggests a variant of this algorithm where, instead of updating the solution vectors, the residual vector is updated as

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k\, A \left(A^T D(\boldsymbol{x}_k)\, A\right)^{-1} A^T \boldsymbol{g}(\boldsymbol{x}_k), \qquad k = 1, 2, \ldots, \tag{9.5.6}$$

and the step-length α_k is determined via line search. Upon convergence the robust solution $\boldsymbol{x}_k$ is computed from the final residual vector $\boldsymbol{r}_k$ as the solution to the consistent system $A\,\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{r}_k$. Five different numerical methods for computing the search direction in (9.5.6) are compared in [171], where it is demonstrated that the choice of the best method depends on the function ρ. Wolke and Schwetlick [258] extend the algorithm to also estimate the parameter β that appears in the function ρ and which must reflect the noise level in the data.

As starting vector for the above iterative methods one can use the ordinary LSQ solution or, alternatively, the solution to the 1-norm problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_1$ (see below).

The iteratively reweighted least squares formalism can also be used to solve linear p-norm problem $\min_{\boldsymbol{x}} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_p$ for $1 < p < 2$. To see this, we note that

$$\begin{aligned} \|\boldsymbol{b} - A\,\boldsymbol{x}\|_p^p &= \sum_{i=1}^{m} |r_i(\boldsymbol{x})|^p = \sum_{i=1}^{m} |r_i(\boldsymbol{x})|^{p-2} r_i(\boldsymbol{x})^2 \\ &= \|W_p(\boldsymbol{r})^{1/2}\,(\boldsymbol{b} - A\,\boldsymbol{x})\|_2^2 \end{aligned}$$

where $W_p(\boldsymbol{r}) = \mathrm{diag}(|r_i|^{p-2})$. The iterations of the corresponding Newton-type algorithm take the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k, \qquad k = 1, 2, \ldots , \tag{9.5.7}$$

where $\Delta\boldsymbol{x}_k$ is the solution to the weighted LSQ problem

$$\min_{\Delta\boldsymbol{x}} \|W_p(\boldsymbol{r}_k)^{1/2}\,(\boldsymbol{r}_k - A\,\Delta\boldsymbol{x})\|_2, \tag{9.5.8}$$

see (Björck [20], section 4.5.2) for details. Experience shows that, for p close to 1, the LSQ problem (9.5.8) tends to become ill-conditioned as the iterations converge to the robust solution. Special algorithms are needed for the case $p = 1$; some references are [6, 51, 153].

For early use in geophysical prospecting see [48, 219].

9.6 Variable projection algorithm

We finish this chapter with a brief discussion of algorithms to solve the separable NLLSQ problem introduced in Section 8.4, where we listed some important properties of this problem. The simplest method of solution that takes into account the special structure is the algorithm NIPALS [256], where an alternating optimization procedure is employed: the linear and nonlinear parameters are successively fixed and the problem is minimized over the complementary set. This iteration, however, only converges linearly.

The *variable projection* algorithm, developed by Golub and Pereyra [101], takes advantage instead of the explicit coupling between the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{a}$, in order to reduce the original minimization problem to two subproblems that are solved in sequence, without alternation. The nonlinear subproblem is smaller (although generally more complex) and involves only the k parameters in $\boldsymbol{\alpha}$. One linear subproblem is solved a posteriori to determine the n linear parameters in $\boldsymbol{a}$. The difference with NIPALS is perhaps subtle, but it makes the algorithm much more powerful, as we will see below.

For the special case when $\phi_j(\boldsymbol{\alpha}, t) = e^{\alpha_j t}$, the problem is called *exponential data fitting,* which is a very common and important application that is notoriously difficult to solve. Fast methods that originated with Prony [204] are occasionally used, but, unfortunately, the original method is not suitable for problems with noise. The best-modified versions are from M. Osborne [175, 176, 177], who uses the separability to achieve better results. As shown in [198], the other method frequently used with success is the variable projection algorithm. A detailed discussion of both methods and their relative merits is found in chapter 1 of [198], while the remaining chapters show a number of applications in very different fields and where either or both methods are used and compared.

The key idea behind the variable projection algorithm is to eliminate the linear parameters analytically by using the pseudoinverse, solve for the nonlinear parameters first and then solve a linear LSQ problem for the remaining parameters. Figure 9.6.1 illustrates the variable projection principle in action. We depict $I - \Phi(\boldsymbol{\alpha})\Phi^{\dagger}(\boldsymbol{\alpha})$ for a fixed $\boldsymbol{\alpha}$ as a linear mapping from $\mathbb{R}^n \to \mathbb{R}^m$. That is, its range is a linear subspace of dimension n (or less, if the matrix is rank deficient). When $\boldsymbol{\alpha}$ varies, this subspace

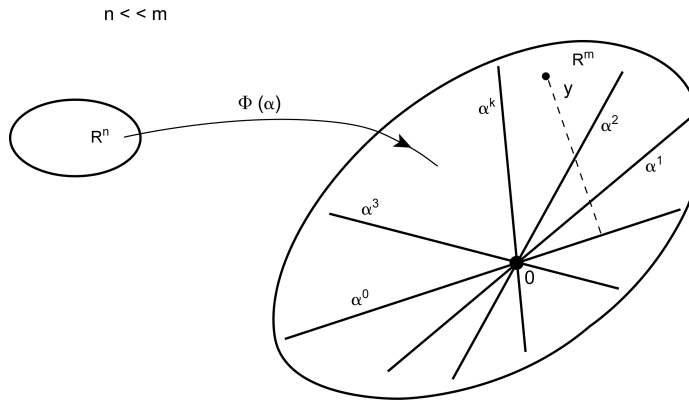


Figure 9.6.1: The geometry behind the variable projection principle. For each $\boldsymbol{\alpha}^i$ the projection $I - \Phi(\boldsymbol{\alpha})\Phi^\dagger(\boldsymbol{\alpha})$ maps $\mathbb{R}^n$ into a subspace in $\mathbb{R}^m$ (depicted as a line).

pivots around the origin. For each $\boldsymbol{\alpha}$ the residual is equal to the Euclidean distance from the data vector $\boldsymbol{y}$ to the corresponding subspace. As usual, there may not be any subspace to which the data belong, i.e., the problem is inconsistent and there is a nonzero residual at the solution. This residual is related to the l_2 approximation ability of the basis functions $\phi_j(\boldsymbol{\alpha}, t)$.

There are several important results proved in [101], [227] and [212], which show that the reduced function is better conditioned than the original one. Also, we observe that

- The reduction in dimensionality of the problem has as a consequence that fewer initial parameters need to be guessed to start a minimization procedure.

- The algorithm is valid in the rank-deficient case. To guarantee continuity of the Moore-Penrose generalized inverse, only local constant rank of $\Phi(\boldsymbol{\alpha})$ needs to be assumed.

- The reduced nonlinear function, although more complex and therefore apparently costlier to evaluate, gives rise to a better-conditioned problem, which always takes fewer iterations to converge than the full problem [212]. This may include convergence when the Levenberg-Marquardt algorithm for the full function does not converge.

- By careful implementation of the linear algebra involved and use of a simplification due to Kaufman [140], the cost per iteration for the reduced function is similar to that for the full function, and thus minimization of the reduced problem is always faster. However, in

hard problems this simplification may lead to a less robust algorithm [172].

The linear problem is easily solved by using the methods discussed in previous chapters. There are two types of iterative methods to solve the NLLSQ problem in the variable projection algorithm:

- **Derivative free methods** such as PRAXIS [263] require only an efficient computation of the nonlinear function

$$f_2(\boldsymbol{\alpha}) = \tfrac{1}{2}\|\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger\, \boldsymbol{y}\|_2^2 = \tfrac{1}{2}\|P_{\Phi(\boldsymbol{\alpha})}\, \boldsymbol{y}\|_2^2.$$

  Instead of the more expensive pseudoinverse computation it is possible to obtain $P_{\Phi(\boldsymbol{\alpha})}$ by orthogonally transforming $\Phi(\boldsymbol{\alpha})$ into trapezoidal form. One obtains then a simple expression for the evaluation of the function from the same orthogonal transformation when applied to $\boldsymbol{y}$. For details see [101].

- **Methods that need derivatives** of $\boldsymbol{r}(\boldsymbol{\alpha}) = \boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha})^\dagger\, \boldsymbol{y}$. In [101], a formula for the Fréchet derivative of the orthogonal projector $P_{\Phi(\boldsymbol{\alpha})}$ was developed and then used in the Levenberg-Marquardt algorithm, namely:

$$D\boldsymbol{r}(\boldsymbol{\alpha}) = -[P^{\perp}_{\Phi(\boldsymbol{\alpha})} D(\Phi(\boldsymbol{\alpha}))\Phi(\boldsymbol{\alpha})^\dagger + (P^{\perp}_{\Phi(\boldsymbol{\alpha})} D(\Phi(\boldsymbol{\alpha}))\Phi(\boldsymbol{\alpha})^\dagger)^T]\boldsymbol{y}.$$

  Kaufman [140] ignores the second term, producing a saving of up to 25% in computer time, without a significant increase in the number of iterations. The Levenberg-Marquardt algorithm used in [101] and [140] starts with an arbitrary $\boldsymbol{\alpha}_0$ and determines the iterates from the relation:

$$\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k - \begin{pmatrix} D\boldsymbol{r}(\boldsymbol{\alpha}_k) \\ \sqrt{\lambda_k} I \end{pmatrix}^\dagger \begin{pmatrix} \boldsymbol{r}(\boldsymbol{\alpha}_k) \\ \boldsymbol{0} \end{pmatrix}.$$

  At each iteration, this linear LSQ problem is solved by orthogonal transformations. Also, the Marquardt parameter λ_k can be determined so that divergence is prevented by enforcing descent:

$$\|\boldsymbol{r}(\boldsymbol{\alpha}_{k+1})\|_2 < \|\boldsymbol{r}(\boldsymbol{\alpha}_k)\|_2.$$

The original VARPRO program by Pereyra is listed in a Stanford University report [100]; there the minimization is done via the Osborne modification of the Levenberg-Marquardt algorithm. A public domain version, including modifications and additions by John Bolstadt, Linda Kaufman and Randy LeVeque, can be found in Netlib [263] under the same name. It incorporates the Kaufman modification, where the second term in the Fréchet derivative is ignored and the information provided by the program is used to generate a statistical analysis, including uncertainty bounds for

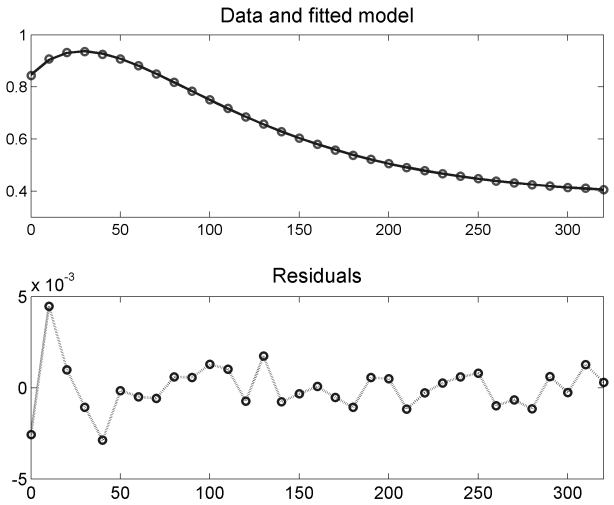


Figure 9.6.2: Data, fit and residuals for the exponential fitting problem.

the estimated linear and nonlinear parameters. Recent work by O'Leary and Rust indicates that in certain problems the Kaufman modification can make the algorithm less robust. They present in that work a modern and more modular implementation. In the PORT library [86] of Netlib there are careful implementations by Gay and Kaufman of variable projection versions for the case of unconstrained and constrained separable NLLSQ problems, including the option of using finite differences to approximate the derivatives. VARP2 [98, 141] is an extension for problems with multiple right-hand sides and it is also available in Netlib.

The VARPRO program was influenced by the state of the art in computing at the time of its writing, leading to a somewhat convoluted algorithm, and in a way most of the sequels inherited this approach. The computational constraints in memory and operation speeds have now been removed, since for most problems to which the algorithm is applicable in standard current machinery, results are produced quickly, even if multiple runs are executed in order to try to get a global minimum. In [198, 228] there is a description of a simplified approach for the case of complex exponentials, where some efficiency is sacrificed in order to achieve a clearer implementation that is easier to use and maintain.

Example 97. *The following example from [174] illustrates the competitiveness of the variable projection method, both in speed and robustness. Given a set of* $m = 33$ *data points* (t_i, y_i) *(the data set is listed in the VARPRO*

code in Netlib), fit the exponential model:

$$M(\boldsymbol{a}, \boldsymbol{\alpha}, t) = a_1 + a_2^{\alpha_1 t} + a_3^{\alpha_2 t}.$$

We compare the Levenberg-Marquardt algorithm for minimization of the full function with VARPRO, which uses the reduced function. The two algorithms needed 32 and 8 iterations, respectively, to reduce the objective function to less than $5 \cdot 10^{-5}$; this is a substantial saving, considering that the cost per iteration is similar for the two approaches. Moreover, the condition numbers of the respective linearized problems close to the solution are 48845 and 6.9; a clear example showing that the reduced problem is better conditioned than the original one. Figure 9.6.2 shows the fit and the residuals for the VP algorithm. The autocorrelation analysis from Section 1.4 gives $\varrho = -5 \cdot 10^{-6}$ and $T_\varrho = 9.7 \cdot 10^{-6}$, showing that the residuals can be considered as uncorrelated and therefore that the fit is acceptable. The least squares solution for this problem is

$$\boldsymbol{a}^* = (0.375, -1.46, 1.94)^T, \quad \boldsymbol{\alpha}^* = (0.0221, 0.0129)^T.$$

As explained in Section 3.5, the sensitivity of this solution can be assessed via the diagonal elements of the estimated covariance matrix

$$\varsigma^2 \big(J(\boldsymbol{x}^*)^T J(\boldsymbol{x}^*)\big)^{-1}.$$

In particular, the square roots of the diagonal elements of this matrix are estimates of the standard deviations of the five parameters. If we use the estimate $\varsigma^2 \simeq \|\boldsymbol{r}(\boldsymbol{x}^*)\|_2^2/m$, then these estimated standard deviations are

$$0.00191, \quad 0.204, \quad 0.203, \quad 0.000824, \quad 0.000413,$$

showing that the two linear parameters a_2 and a_3 are potentially very sensitive to perturbations. To illustrate this, the two alternative sets of parameters

$$\hat{\boldsymbol{a}} = (0.374, -1.29, 1.76)^T, \quad \hat{\boldsymbol{\alpha}} = (0.0231, 0.0125)^T$$

and

$$\tilde{\boldsymbol{a}} = (0.378, -2.29, 2.76)^T, \quad \tilde{\boldsymbol{\alpha}} = (0.0120, 0.0140)^T,$$

both give fits that are almost indistinguishable from the least squares fit. The corresponding residual norms are

$$\|\boldsymbol{r}(\boldsymbol{x}^*)\|_2 = 7.39 \cdot 10^{-3}, \quad \|\boldsymbol{r}(\hat{\boldsymbol{x}})\|_2 = 7.79 \cdot 10^{-3}, \quad \|\boldsymbol{r}(\tilde{\boldsymbol{x}})\|_2 = 8.24 \cdot 10^{-3},$$

showing that the large changes in a_2 and a_3 give rise to only small variations in the residuals, again demonstrating the lack of sensitivity of the residual to changes in those parameters.

9.7 Block methods for large-scale problems

We have already mentioned the inexact Gauss-Newton and Levenberg-Marquardt methods, based on truncated iterative solvers, as a way to deal with large computational problems. A different approach is to use a divide-and-conquer strategy that, by decomposing the problem into blocks, may allow the use of standard solvers for the (smaller) blocks. First, one subdivides the observations in appropriate non-overlapping groups. Through an SVD analysis one can select those variables that are more relevant to each subset of data; details of one such method for large-scale, ill-conditioned nonlinear least squares problems are given below and in [188, 189, 191].

The procedure works best if the data can be broken up in such a way that the associated variables have minimum overlap and only weak couplings are left with the variables outside the block. Of course, we cannot expect the blocks to be totally uncoupled; otherwise, the problem would decompose into a collection of problems that can be independently solved.

Thus, in general, the procedure consists of an outer block nonlinear Gauss-Seidel or Jacobi iteration, in which the NLLSQ problems corresponding to the individual blocks are solved approximately for their associated parameters. The block solver is initialized with the current value of the variables. The full parameter set is updated either after each block is processed (Gauss-Seidel strategy of information updating as soon as it is available), or after all the block solves have been completed (Jacobi strategy). The ill-conditioning is robustly addressed by the use of the Levenberg-Marquardt algorithm for the subproblems and by the threshold used to select the variables for each block.

The pre-processing for converting a large problem into block form starts by scanning the data and subdividing it. The ideal situation is one in which the data subsets are only sensitive to a small subset of the variables. Having performed that subdivision, we proceed to analyze the data blocks to determine which parameters are actually well determined by each subset of data. During this data analysis phase we compute the SVD of the Jacobian of each block; this potentially very large matrix is trimmed by deleting columns that are zero or smaller than a given threshold. Finally, the right singular vectors of the SVD are used to complete the analysis.

Selecting subspaces through the SVD

Jupp and Vozoff [138] introduced the idea of relevant and irrelevant parameters based on the SVD. We write first the Taylor expansion of the residual vector at a given point $\boldsymbol{x}$ (see §8.2):

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{r}(\boldsymbol{x}) + J(\boldsymbol{x})\,\boldsymbol{h} + O(\|\boldsymbol{h}\|_2^2). \tag{9.7.1}$$

Considering the SVD of the Jacobian $J(\boldsymbol{x}) = U\,\Sigma\,V^T$, we can introduce the so-called rotated perturbations

$$\boldsymbol{p} = \sigma_1 V^T \boldsymbol{h},$$

where σ_1 is the largest singular value of $J(\boldsymbol{x})$. Neglecting higher-order terms in (9.7.1) we can write this system as

$$\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{r}(\boldsymbol{x}) = J(\boldsymbol{x})\,\boldsymbol{h} = \sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_1}\right) p_i\,\boldsymbol{u}_i,$$

where σ_i/σ_1 are the normalized singular values and r is the rank of the Jacobian. This shows the direct relationship between the normalized singular values, the rotated perturbations $p_i = \sigma_1 \boldsymbol{v}_i^T \boldsymbol{h}$, and their influence on the variation of the residual vector. Thus,

$$\|\boldsymbol{r}(\boldsymbol{x}+\boldsymbol{h}) - \boldsymbol{r}(\boldsymbol{x})\|_2^2 = \sum_{i=1}^{r} \left(\frac{\sigma_i}{\sigma_1}\right)^2 p_i^2,$$

which shows that those parameters p_i that are associated with small normalized singular values will not contribute much to variations in the residual. Here we have assumed that all the components of the perturbation vector $\boldsymbol{h}$ are of similar size.

The above analysis is the key to the algorithm for partitioning the parameter set into blocks, once a partition of the data set has been chosen. Let RI_k denote the row indices for the kth block of data with $m^{[k]}$ elements. Given a base point $\boldsymbol{x}$, calculate the Jacobian $J(\boldsymbol{x})^{[k]}$ for this data set, i.e., $J(\boldsymbol{x})^{[k]}$ has only $m^{[k]}$ rows and less than n columns, since if the data have been partitioned appropriately and due to the local representation of the model, we expect that there will be a significant number of columns with small components that can be safely neglected. Then the procedure is as follows:

1. Compute the SVD of $J(\boldsymbol{x})^{[k]}$ and normalize the singular values by dividing them by the largest one.

2. Select the first $n^{[k]}$ normalized singular values that are above a given threshold, and their associated right singular vectors.

3. Inspect the set of chosen singular vectors and select the largest components of $V^{[k]}$in absolute value.

4. Choose the indices of the variables in parameter space corresponding to the columns of $V^{[k]}$ that contain large entries to form the set CI_k of column indices. This selects the subset of parameters that have most influence on the variation in the misfit functional for the given data set.

With this blocking strategy, variables may be repeated in different subsets. Observe also that it is possible for the union of all the subsets to be smaller than the entire set of variables; this will indicate that there are variables that cannot be resolved by the given data, at least in a neighborhood of the base point $\boldsymbol{x}$ and for the chosen threshold. Since this analysis is local, it should be periodically revised as the optimization process advances.

Once this partition has been completed, we use an outer block nonlinear Gauss-Seidel or Jacobi iteration [173] in order to obtain the solution of the full problem. To make this more precise, let us partition the index sets $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$ into the subsets $\{\mathsf{RI}_k\}$ and $\{\mathsf{CI}_k\}$, i.e., the index sets that describe the partition of our problem into blocks. Each subproblem can now be written as

$$\min_{\boldsymbol{x}} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})^{[k]}\|_2^2 \quad \text{subject to} \quad x = x_i^*, \quad i \in \{1, 2, \ldots, n\} - \mathsf{CI}_k, \tag{9.7.2}$$

where $\boldsymbol{r}(\boldsymbol{x})^{[k]}$ is the sub-vector of $\boldsymbol{r}(\boldsymbol{x})$ with elements $\{r(\boldsymbol{x})_i\}_{i \in \mathsf{RI}_k}$. In other words, we fix the values of the parameters that are *not* in block k to their current values in the global set of variables x^*. Observe that the dimension of the subproblem is then $m^{[k]} \times n^{[k]}$. By considering enough subsets $k = 1, \ldots, K$ we can make these dimensions small, especially $n^{[k]}$ and therefore make the subproblems (9.7.2) amenable to direct techniques (and global optimization, if necessary).

One step of a sequential block Gauss-Seidel iteration consists then in sweeping over all the blocks, solving the subproblems to a certain level of accuracy, and replacing the optimal values in the central repository of all the variables at once. A sequential block Jacobi iteration does the same, but it does not replace the values until the sweep over all the blocks is completed.

Since we allow repetitions of variables in the sub-blocks, it is prudent to introduce averaging of the multiple appearances of variables. In the case of Jacobi, this can be done naturally at the end of a sweep. In the case of Gauss-Seidel, one needs to keep a count of the repetitions and perform a running average for each repeated variable.

Parallel asynchronous block nonlinear Gauss-Seidel iteration

The idea of chaotic relaxation for linear systems originated with Rosenfeld in 1967 [211]. Other early actors in this important topic were A. Ostrowski [178] and S. Schechter [221]. Chazan and Miranker published in 1969 a detailed paper [43] describing and formalizing chaotic iterations for the parallel iterative solution of systems of linear equations. This was extended to the nonlinear case in [159, 160].

The purpose of chaotic relaxation is to facilitate the parallel implementation of iterative methods in a multi-processor system or in a network of computers by reducing the amount of communication and synchronization between cooperating processes and by allowing that assigned sub-tasks go unfulfilled. This is achieved by not requiring that the relaxation follow a pre-determined sequence of computations, but rather letting the different processes start their evaluations from a current, centrally managed value of the unknowns.

Baudet [10] defines the class of asynchronous iterative methods and shows that it includes chaotic iterations. Besides the classical Jacobi and Gauss-Seidel approaches he introduces the purely asynchronous method in which, at each iteration within a block, current values are always used. This is a stronger cooperative approach than Gauss-Seidel and he shows in numerical experimentation how it is more efficient with an increasing number of processors.

We paraphrase a somewhat more restricted definition for the case of block nonlinear optimization that concerns us. Although Baudet's method would apply directly to calculating the zeros of $\nabla f(\boldsymbol{x})$, we prefer to describe the method in the optimization context in which it will be used. We use the notation introduced above for our partitioned problem.

Let

$$f(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \sum_{k=1}^{K} \tfrac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})^{[k]}\|_2^2 = \sum_{k=1}^{K} f(\boldsymbol{x})^{[k]},$$

where the data are partitioned into K non-overlapping blocks. An asynchronous iteration for calculating $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ starting from the vector $\boldsymbol{x}_0$ is a sequence of vectors $\boldsymbol{x}_k$ with elements defined by

$$x_j^{[k]} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})^{[k]} \qquad \text{for} \qquad j \in \mathsf{CI}_k$$

subject to

$$x_j^{[k]} = x_j \qquad \text{for} \qquad j \notin \mathsf{CI}_k.$$

The initial vector for the kth minimization is

$$\boldsymbol{x}^{[k]} = (x_1(s_1(k)), \ldots, x_n(s_n(k))),$$

where $S = \{s_1(k), \ldots, s_n(k)\}$ is a sequence of elements in N^n that indicates at which iteration a particular component was last updated. In addition, the following conditions should be satisfied:

$$s_i(k) \leq k - 1 \qquad \text{and} \qquad s_i(k) \to \infty, \text{ as } k \to \infty.$$

These conditions guarantee that all the variables are updated often enough, while the formulation allows for the use of updated subsets of variables "as

they become available." Baudet [10] gives sufficient conditions for the convergence of this type of process for systems of nonlinear equations. The convergence of an asynchronous block Gauss-Seidel process and as a special case, the previously described sequential algorithm, follows from the following theorem proved in [188].

Theorem 98. ***Convergence of the asynchronous BGS L-M iteration.*** *Let the operator*

$$T(\boldsymbol{x}) = \boldsymbol{x} - (J(\boldsymbol{x})^T J(\boldsymbol{x}) + \lambda I)^{-1} J(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$$

be Lipschitz, with constant γ, and uniformly monotone with constant μ, in a neighborhood of a stationary point of $f(\boldsymbol{x})$, with $2\mu/\gamma^2 > 1$. Then T is vector contracting and the asynchronous Levenberg-Marquardt method for minimizing $\frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x})\|_2^2$ is convergent.

As we indicated above, an alternative to this procedure is a chaotic block Jacobi iteration, where all the processes use the same initial vector at the beginning of a sweep, at the end of which the full parameter vector is updated. In general, asynchronous block Gauss-Seidel with running averages is the preferred algorithm, since Jacobi requires a synchronization step at the end of each sweep that creates issues of load balancing.

Appendix A

Sensitivity Analysis

A.1 Floating-point arithmetic

The classic reference for finite precision arithmetic is Wilkinson's monograph "Rounding Errors in Algebraic Processes" [255], while a more recent treatment is Higham's "Accuracy and Stability of Numerical Algorithms" [128]. Almost any numerical analysis book has an introductory chapter about this topic. Here we list some of the basic ideas used in our text.

Digital computers use floating-point representation for real and complex numbers based on the binary system, i.e., the basis is 2. Real numbers are rewritten in a special normalized form, where the mantissa is less than 1. Usually there is the option to use single (t-digit) or double ($2t$-digit) length mantissa representation and arithmetic. If we denote by $\mathbf{fl}(x)$ the floating-point computer representation of a real number x, and by $\oplus$ the floating-point addition, then the *unit round-off* μ (for a given computer) is defined as the smallest ε such that in floating-point arithmetic: $\mathbf{fl}(1) \oplus \varepsilon > \mathbf{fl}(1)$.

For a binary t-digit floating-point system $\mu = 2^{-t}$. The machine epsilon $\varepsilon_M = 2\mu$ is the gap between 1 and the next larger floating-point number (and, thus, in a relative sense, gives an indication of the gap between the floating-point numbers). Several of the bounds in this book contain the unit round-off or the machine precision; it is therefore advisable to check the size of ε_M for a particular machine and word length. A small Fortran program is available from Netlib to compute the machine precision for double precision and can be adapted easily for single.

Representation error. The relative error in the computer representation $\mathbf{fl}(x)$ of a real number $x \neq 0$ satisfies

$$\frac{|\mathbf{fl}(x) - x|}{|x|} \leq \mu,$$

implying that $\mathbf{fl}(x) \in [x(1-\varepsilon_M),\, x(1+\varepsilon_M)]$.

Rounding error. The error in a given floating-point operation $\circledast$, corresponding to the real operation $*$, satisfies

$$\mathbf{fl}(x) \circledast \mathbf{fl}(y) = (x * y)(1+\varepsilon), \qquad \text{with} \quad |\varepsilon| \leq \mu.$$

To measure the cost of the different algorithms described in the book we use as the unit the *flop*. A word of warning though: its definition differs from one author to the other; here we follow the one used in [105, 128], which is also common in many articles in the literature.

Definition 1. A flop is roughly the work associated with a floating-point operation (addition, subtraction, multiplication, or division).

In March 2011 the cheapest cost per Gigaflop (10^9 flops) was \$1.80, achieved on the computer cluster HPU4Science, made of six dual Core 2 Quad off-the-shelf machines at a cost of \$30,000, with performance enhanced by combining the CPUs with the Graphical PUs. In comparison, the cost in 1984 was \$15 million on a Cray X-MP.

A.2 Stability, conditioning and accuracy

A clear and concise review of these topics can be found in [57, 128, 237]. One general comment first: given a t-digit arithmetic, there is a limit to the attainable accuracy of any computation, because even the data themselves may not be representable by a t-digit number. Additionally, in practical applications, one should not lose sight of the fact that usually the data, derived from observations, already have a physical error much larger than the one produced by the floating-point representation.

Let us formally define a *mathematical problem* by a function that relates a data space $\mathcal{X}$ with a solutions space $\mathcal{Y}$, i.e., $P : \mathcal{X}(\text{data}) \to \mathcal{Y}(\text{solutions})$. Let us also define a specific *algorithm* for this problem as a function $\widetilde{P} : \mathcal{X} \to \mathcal{Y}$. One is interested in evaluating how close the solution computed by the algorithm is to the exact solution of the mathematical problem.

The accuracy will depend on the sensitivity of the mathematical problem P to perturbations of its data, the *condition of the problem*, and on the sensitivity of the algorithm $\tilde{P}$ to perturbations of the input data, the *stability* of the algorith.

The condition of the mathematical problem is commonly measured by the *condition number* $\kappa(\boldsymbol{x})$. We emphasize the problem dependency, so, for example, the same matrix A may give rise to an ill-conditioned least squares problem and a well-conditioned eigenvector problem.

A formal definition of the condition number follows.

Definition 2. The condition number is defined by

$$\kappa(\boldsymbol{x}) = \sup_{\delta \boldsymbol{x}} \frac{\|P(\boldsymbol{x}+\delta\boldsymbol{x}) - P(\boldsymbol{x})\|_2}{\|P(\boldsymbol{x})\|_2} \Big/ \frac{\|\delta\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}.$$

If the mapping P is differentiable with Jacobian $[J(\boldsymbol{x})]_{ij} = \frac{\partial P_i}{\partial x_j}$, the above formula can be replaced by

$$\kappa\left(\boldsymbol{x}\right) = \frac{\|J(\boldsymbol{x})\|_2 \, \|\boldsymbol{x}\|_2}{\|P\left(\boldsymbol{x}\right)\|_2}.$$

The condition number $\kappa\left(\boldsymbol{x}\right)$ is the leading coefficient of the data perturbation in the error bound for the solution.

Among the possible formalizations of the stability concept, *backward stability* is a convenient one; an algorithm is backward stable if it computes the exact solution for slightly perturbed data, or in other words, quoting [237], an algorithm is backward stable if "[it] gives the right answer to nearly the right question": $\widetilde{P}(\boldsymbol{x}) = P(\tilde{\boldsymbol{x}}) = P(\boldsymbol{x} + \Delta\boldsymbol{x})$. The perturbation of the input data $\|\Delta\mathbf{x}\|_2$ is the *backward error*. Formally,

Definition 3. The algorithm $\widetilde{P}$ is backward stable if the computed solution $\tilde{\boldsymbol{x}}$ is the exact solution of a slightly perturbed problem; i.e., if

$$\widetilde{P}(\boldsymbol{x}) = P(\tilde{\boldsymbol{x}})$$

for some $\widetilde{\mathbf{x}}$ with

$$\frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = O(\varepsilon_M).$$

One can bound the actual error in the solution $P(\boldsymbol{x})$ of a problem with condition number $\kappa(\boldsymbol{x})$ if it is computed using a backward stable algorithm $\widetilde{P}\left(\boldsymbol{x}\right)$:

$$\frac{\left\|\widetilde{P}\left(\boldsymbol{x}\right) - P(\boldsymbol{x})\right\|_2}{\|P(\boldsymbol{x})\|_2} = O(\kappa(\boldsymbol{x})\varepsilon_M).$$

In words, a backward stable algorithm, when applied to a well-conditioned problem, yields an accurate solution, and if the backward error is smaller than the data errors, the problem is solved to the same extent that it is actually known. On the other hand, the computed solution to an ill-conditioned problem can have a very large error, even if the algorithm used was backward stable, the condition number acting as an amplification factor of the data errors.

Appendix B

Linear Algebra Background

B.1 Norms

Vector norms

The most commonly used norms for vectors are the l_1-, l_2- and l_∞-norms, denoted by $||\cdot||_1$, $||\cdot||_2$, and $||\cdot||_\infty$, respectively. These norms are defined by:

- $\|\boldsymbol{v}\|_1 = \sum_{i=1}^{n} |v_i|$.
- $\|\boldsymbol{v}\|_2 = \left(\sum_{i=1}^{n} |v_i|^2\right)^{\frac{1}{2}} = \left(\boldsymbol{v}^T\boldsymbol{v}\right)^{\frac{1}{2}}$ (Euclidean norm).
- $\|\boldsymbol{v}\|_\infty = \max_{1\le i\le n} |v_i|$ (Chebyshev norm).

A useful relationship between an inner product and the l_2-norms of its factors is the Cauchy-Schwartz inequality:

$$\left|\boldsymbol{x}^T\boldsymbol{y}\right| \le \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 .$$

Norms are continuous functions of the entries of their arguments. It follows that a sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ converges to a vector $\boldsymbol{x}$ if and only if $\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}\| = 0$ for any norm.

Matrix norms

A natural definition of a matrix norm is the so-called induced or operator norm that, starting from a vector norm $\|\cdot\|$, defines the matrix norm as the maximum amount that the matrix A can stretch a unit vector, or more formally: $\|A\| = \max_{\|\boldsymbol{v}\|=1} \|A\boldsymbol{v}\|$. Thus, the induced norms associated with the usual vector norms are

- $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$.
- $\|A\|_2 = [\max(\text{eigenvalue of } (A^T A))]^{\frac{1}{2}}$.
- $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|$.

In addition the so-called Frobenius norm (Euclidean length of A considered as an nm-vector) is

- $\|A\|_{\mathrm{F}} = \left(\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2\right)^{\frac{1}{2}} = \mathrm{trace}(A^T A)^{\frac{1}{2}}$.

For square, orthogonal matrices $Q \in \mathbb{R}^{n\times n}$ we have $\|Q\|_2 = 1$ and $\|Q\|_{\mathrm{F}} = \sqrt{n}$. Both the Frobenius and the matrix l_2-norms are compatible with the Euclidean vector norm. This means that $\|A\boldsymbol{x}\| \leq \|A\| \, \|\boldsymbol{x}\|$ is true when using the l_2-norm for the vector and either the l_2 or the Frobenius norm for the matrix. Also, both are invariant with respect to orthogonal transformations Q:

$$\|QA\|_2 = \|A\|_2 , \qquad \|QA\|_{\mathrm{F}} = \|A\|_{\mathrm{F}} .$$

In terms of the singular values of A, the l_2 norm can be expressed as $\|A\|_2 = \max_i \sigma_i = \sigma_1$, where σ_i, $i = 1, \ldots, \min(m, n)$ are the singular values of A in descending order of size. In the special case of symmetric matrices the l_2-norm reduces to $\|A\|_2 = \max_i |\lambda_i|$, with λ_i an eigenvalue of A. This is also called the *spectral radius* of the matrix A.

B.2 Condition number

The condition number of a general matrix A in norm $\|\cdot\|_p$ is

$$\kappa_p(A) = \|A\|_p \left\|A^\dagger\right\|_p .$$

For a vector-induced norm, the condition number of A is the ratio of the maximum to the minimum stretch produced by the linear transformation represented by this matrix, and therefore it is greater than or equal to 1. In the l_2-norm, $\kappa_2(A) = \sigma_1/\sigma_r$, where σ_r is the smallest nonzero singular value of A and r is the rank of A.

In finite precision arithmetic, a large condition number can be an indication that the "exact" matrix is close to singular, as some of the zero singular values may be represented by very small numbers.

B.3 Orthogonality

The notation used for the inner product of vectors is $\boldsymbol{v}^T\boldsymbol{w} = \sum_i v_i w_i$. Note that $\|\boldsymbol{v}\|_2^2 = \boldsymbol{v}^T\boldsymbol{v}$. If $\boldsymbol{v}, \boldsymbol{w} \neq \mathbf{0}$, and $\boldsymbol{v}^T\boldsymbol{w} = 0$, then these vectors are orthogonal, and they are orthonormal if in addition they have unit length.

Orthogonal matrices. A square matrix Q is orthogonal if $Q^TQ = I$ or $QQ^T = I$, i.e., the columns or rows are orthonormal vectors and thus $\|Q\|_2 = 1$. It follows that orthogonal matrices represent isometric transformations that can only change the direction of a vector (rotation, reflection), but not its Euclidean norm, a reason for their practical importance: $\|Q\boldsymbol{v}\|_2 = \|\boldsymbol{v}\|_2$, $\|QA\|_2 = \|AQ\|_2 = \|A\|_2$.

Permutation matrix. A permutation matrix is an identity matrix with permuted rows or columns. It is orthogonal, and products of permutation matrices are again permutations.

Orthogonal projection onto a subspace of an inner product space. Given an orthonormal basis, $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$ of a subspace $\mathcal{S} \subseteq \mathcal{X}$, where $\mathcal{X}$ is an inner product space, the orthogonal projection $\mathbf{P} : \mathcal{X} \to \mathcal{S}$ satisfies

$$\mathbf{P}\boldsymbol{x} = \sum_{i=1}^{n} (\boldsymbol{x}^T\boldsymbol{u}_i)\,\boldsymbol{u}_i.$$

The operator $\mathbf{P}$ is linear and satisfies $\mathbf{P}\boldsymbol{x} = \boldsymbol{x}$ if $\boldsymbol{x} \in \mathcal{S}$ (idempotent) and $\|\mathbf{P}\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_2\ \forall \boldsymbol{x} \in \mathcal{X}$. Therefore, the associated square matrix P is an orthogonal projection matrix if it is Hermitian and idempotent, i.e., if

$$P^T = P \text{ and } P^2 = P.$$

Note that an orthogonal projection divides the whole space into two orthogonal subspaces. If $\mathbf{P}$ projects a vector onto the subspace $\mathcal{S}$, then $\mathbf{I} - \mathbf{P}$ projects it onto $S^\perp$, the orthogonal complement of $\mathcal{S}$ with $S + S^\perp = \mathcal{X}$ and $S \cap S^\perp = 0$. An orthogonal projection matrix P is not necessarily an orthogonal matrix, but $I - 2P$ is orthogonal (see Householder transformations in Section 4.3).

An important projector is defined by a matrix of the form $P_U = UU^T$, where U has p orthonormal columns $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_p$. This is a projection onto the subspace spanned by the columns of U. In particular, the projection onto the subspace spanned by a single (not necessarily of norm 1) vector $\boldsymbol{u}$ is defined by the rank-one matrix $P_{\boldsymbol{u}} = \frac{\boldsymbol{u}\boldsymbol{u}^T}{\boldsymbol{u}^T\boldsymbol{u}}$.

Gram matrix. The Gram matrix or Grammian $\mathrm{Gram}(A)$ of an $m \times n$ matrix A is A^TA. Its elements are thus the n^2 possible inner products between pairs of columns of A.

B.4 Some additional matrix properties

The **Sherman-Morrison-Woodbury** formula gives a representation of the inverse of a rank-one perturbation of a matrix in terms of its inverse:

$$(A + \boldsymbol{u}\boldsymbol{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^T A^{-1}}{1 + \boldsymbol{v}^T A^{-1}\boldsymbol{u}},$$

provided that $1 + \boldsymbol{v}^T A^{-1}\boldsymbol{u} \neq 0$. As usual, for calculations, $A^{-1}\boldsymbol{u}$ is short hand for "solve the system" $A\boldsymbol{x} = \boldsymbol{u}$.

The next theorem presents the **interlacing property of the singular values** of A with those of matrices obtained by removing or adding a column or a row (see [20]).

Theorem 4. *Let A be bordered by a column $\boldsymbol{u} \in \mathbb{R}^m$, $\hat{A} = (A, \boldsymbol{u}) \in \mathbb{R}^{m\times n}$, $m \geq n$. Then, the ordered singular values σ_i of A separate the singular values of $\widehat{A}$ as follows:*

$$\widehat{\sigma}_1 \geq \sigma_1 \geq \widehat{\sigma}_2 \geq \sigma_2 \geq ... \geq \widehat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \widehat{\sigma}_n.$$

Similarly, if A is bordered by a row $\boldsymbol{v} \in \mathbb{R}^n$,

$$\hat{A} = \begin{pmatrix} A \\ \boldsymbol{v}^T \end{pmatrix} \in \mathbb{R}^{m\times n}, \qquad m \geq n,$$

then

$$\widehat{\sigma}_1 \geq \sigma_1 \geq \widehat{\sigma}_2 \geq \sigma_2 \geq ... \geq \widehat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \widehat{\sigma}_n \geq \sigma_n.$$

Appendix C

Advanced Calculus Background

C.1 Convergence rates

Definition 5. Let x^*, $x_k \in \mathbb{R}$ for $k = 0, 1, \ldots$ The sequence $\{x_k\}$ is said to converge to x^* if

$$\lim_{k \to \infty} |x_k - x^*| = 0.$$

The convergence is
linear if $\exists\ c \in [0, 1)$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|\,;$$

superlinear if $\exists\ c_k \longrightarrow 0$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c_k\,|x_k - x^*|\,;$$

quadratic if $\exists\ c \in [0, 1)$ and an integer $K > 0$ such that for $k \geq K$,

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|^2\,.$$

Definition 6. A locally convergent iterative algorithm converges to the correct answer if the iteration starts close enough. **A globally convergent iterative algorithm** converges when starting from almost any point. For minimization, this is not to be confused with finding the global minimum of a functional on a compact domain (see below).

Global and local minimum: $\boldsymbol{x}^*$ is a global minimizer of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ on a compact domain D if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^n$. $\boldsymbol{x}^*$ is a local minimizer inside a certain region, usually defined as an open "ball" of size δ around $\boldsymbol{x}^*$, if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \delta$.

C.2 Multivariable calculus

The gradient and Hessian of a scalar function of several variables $f(\boldsymbol{x})$ are a vector and a matrix, respectively, defined by

$$\nabla f(\boldsymbol{x}) \equiv (\partial f/\partial x_1, ..., \partial f/\partial x_n)^T, \qquad \nabla^2 f(\boldsymbol{x}) \equiv \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right].$$

For a vector function of several variables

$$\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^T,$$

with each $r_k(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, we denote by $J(\boldsymbol{x})$ the Jacobian of $\boldsymbol{r}(\boldsymbol{x})$ and by G_k the Hessian of a component function r_k:

$$J(\boldsymbol{x}) = \left[\frac{\partial r_i}{\partial x_j}\right], \qquad G_k(\boldsymbol{x}) = \left[\frac{\partial^2 r_k}{\partial x_i \partial x_j}\right].$$

Definition 7. Descent direction $\boldsymbol{p}$ of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$.

$\boldsymbol{p}$ is a descent direction at $\boldsymbol{x}_c$ for a function $f(\boldsymbol{x})$ if for sufficiently small and positive α $\|f(\boldsymbol{x}_c+\alpha\boldsymbol{p}\| < \|f(\boldsymbol{x}_c)\|$. Alternatively, $\boldsymbol{p}$ is a descent direction at $\boldsymbol{x}_c$ if the directional derivative (projection of the gradient on a given direction) of the function $f(\boldsymbol{x})$ at $\boldsymbol{x}_c$ in the direction $\boldsymbol{p}$ is negative: $\triangledown f(\boldsymbol{x}_c)^T\boldsymbol{p} < 0$.

Theorem 8. ***Taylor's theorem for a scalar function.***

If $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable, then, for some $t \in [0, 1]$,

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x} + t\boldsymbol{p})^T\boldsymbol{p}.$$

If $f(\boldsymbol{x})$ is twice continuously differentiable then, for some $t \in [0, 1]$,

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T\boldsymbol{p} + \boldsymbol{p}^T\nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p}.$$

A necessary condition for $\boldsymbol{x}^*$ to be a stationary point of $f(\boldsymbol{x})$ is that $\nabla f(\boldsymbol{x}^*) = 0$. The sufficient conditions for $\boldsymbol{x}^*$ to be a local minimizer are $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$ and $\nabla^2(\boldsymbol{x}^*)$ is positive definite.

The derivative $\mathbf{D}A(\boldsymbol{x})$ of an $m \times n$ nonlinear matrix function $A(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^k$ is a tri-dimensional tensor formed with k matrices (slabs) of dimension $m \times n$, each one containing the partial derivatives of the elements of A with respect to one of the variables of the vector $\boldsymbol{x}$. Thus, the second derivative of the vector function $\boldsymbol{r}(\boldsymbol{x})$ is the tri-dimensional tensor $\mathbf{G} = [\mathbf{G}_1, ..., \mathbf{G}_m]$.

The derivative of the orthogonal projector $P_{A(\boldsymbol{x})}$ onto the column space of a differentiable $m \times n$ matrix function $A(\boldsymbol{x})$ of local constant rank can be obtained as follows.

Lemma 9. (Lemma 4.1 in [101]) Let $A^\dagger(x)$ be the pseudoinverse of $A(\boldsymbol{x})$. Then $P_{A(\boldsymbol{x})} = AA^\dagger$ and

$$\mathbf{D}P_{A(\boldsymbol{x})} = P^\perp_{A(\boldsymbol{x})}\mathbf{D}AA^\dagger + (P^\perp_{A(\boldsymbol{x})}\mathbf{D}AA^\dagger)^T,$$

where $P^\perp_A = I - P_{A(\boldsymbol{x})}$.

Definition 10. A function f is Lipschitz continuous with constant γ in a set $X \subseteq \mathbb{R}$ if

$$\forall x,\, y \in X, \quad |f(x) - f(y)| \leq \gamma\,|x - y|\,. \tag{C.2.1}$$

Lipschitz continuity is an intermediary concept between continuity and differentiability. The operator $V : \mathbb{R}^n \to \mathbb{R}^n$, is Lipschitz continuous with constant γ in a set $X \subseteq \mathbb{R}^n$ if

$$\forall \boldsymbol{x},\, \boldsymbol{y} \in X, \quad \|V(\boldsymbol{x}) - V(\boldsymbol{y})\|_2 \leq \gamma\,\|\boldsymbol{x} - \boldsymbol{y}\|_2\,.$$

V is contracting if it is Lipschitz continuous with constant less than unity: $\gamma < 1$. V is uniformly monotone if there exists a positive m such that

$$m\,\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \leq (V(\boldsymbol{x}) - V(\boldsymbol{y}))^T(\boldsymbol{x} - \boldsymbol{y}).$$

V is vector Lipschitz if

$$|V(\boldsymbol{x}) - V(\boldsymbol{y})| \leq A\,|\boldsymbol{x} - \boldsymbol{y}|\,,$$

where A is a non-negative $n \times n$ matrix and the inequality is meant element-wise. V is vector contracting if it is vector Lipschitz continuous with $\|A\| < 1$.

Lagrange multipliers

Lagrange multipliers are used to find the local extrema of a function $f(\boldsymbol{x})$ of n variables subject to k equality constraints $g_i(\boldsymbol{x}) = c_i$, by reducing the problem to an $(n-k)$-variable problem without constraints. Formally, new scalar variables $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_k)^T$, one for each contraint, are introduced and a new function is defined as

$$F(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i(g_i(\boldsymbol{x}) - c_i).$$

The local extrema of this extended function F (the Lagrangian), occur at the points where its gradient is zero: $\nabla F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}$, $\boldsymbol{\lambda} \neq \mathbf{0}$, or equivalently,

$$\nabla_x F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}, \qquad \nabla_{\boldsymbol{\lambda}} F(\boldsymbol{x}, \boldsymbol{\lambda}) = \mathbf{0}.$$

This form encodes compactly the constraints, because

$$\nabla_{\lambda_i} F(\boldsymbol{x}, \boldsymbol{\lambda}) = 0 \Leftrightarrow g_i(\boldsymbol{x}) = c_i.$$

An alternative to this method is to use the equality constraints to eliminate k of the original variables. If the problem is large and sparse, this elimination may destroy the sparsity and thus the Lagrange multipliers approach is preferable.

Appendix D

Statistics

D.1 Definitions

We list some basic definitions and techniques taken from statistics that are needed to understand some sections of this book.

- A ***sample space*** is the set of possible outcomes of an experiment or a random trial. For some kind of experiments (trials), there may be several plausible sample spaces available. The complete set of outcomes can be constructed as a Cartesian product of the individual sample spaces. An ***event*** is a subset of a sample space.

- We denote by Pr{event} the ***probability of an event***. We often consider finite sample spaces, such that each outcome has the same probability (equiprobable).

- X is a ***random variable*** defined on a sample space if it assigns a unique numerical value to every outcome, i.e., it is a real-valued function defined on a sample space.

- X is a ***continuous random variable*** if it can assume every value in an interval, bounded or unbounded (continuous distribution). It can be characterized by a ***probability density function (pdf)*** $p(x)$ defined by
$$\Pr\{a \leq X \leq b\} = \int_a^b p(x)\,dx.$$

- X is a ***discrete random variable*** if it assumes at most a countable set of different values (discrete distribution). It can be characterized by its ***probability function (pf),*** which specifies the probability

that the random variable takes each of the possible values from say $\{x_1, x_2, \ldots, x_i, \ldots\}$:

$$p(x_i) = \Pr\{X = x_i\}.$$

- All distributions, discrete or continuous, can be characterized through their (cumulative) ***distribution function***, the total probability up to, and including, a point x. For a discrete random variable,

$$P(x_i) = \Pr\{X \leq x_i\}.$$

- The ***expectation*** or ***expected value*** $\mathcal{E}[X]$ for a random variable X or equivalently, the mean μ of its distribution, contains a summary of the probabilistic information about X. For a continuous variable X, the expectation is defined by

$$\mathcal{E}[X] = \int_{-\infty}^{\infty} x\, p(x)\, dx.$$

For a discrete random variable with probability function $p(x)$ the expectation is defined as

$$\mathcal{E}[X] = \sum_{\forall i} x_i\, p(x_i).$$

For a discrete random variable X with values $x_1, x_2, \ldots, x_N$ and with all $p(x_i)$ equal (all x_i equiprobable, $p(x_i) = \frac{1}{N}$), the expected value coincides with the ***arithmetic mean***

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i.$$

- The expectation is linear, i.e., for a, b, c constants and X, Y two random variables,

$$\begin{aligned} \mathcal{E}[X + c] &= \mathcal{E}[X] + c, \\ \mathcal{E}[aX + bY] &= \mathcal{E}[aX] + \mathcal{E}[bY]. \end{aligned}$$

- A convenient measure of the dispersion (or spread about the average) is the ***variance*** of the random variable, $\mathrm{var}\,[X]$ or ς^2:

$$\mathrm{var}\,[X] = \varsigma^2 = \mathcal{E}[(X - \mathcal{E}(X))^2].$$

In the equiprobable case this reduces to

$$\mathrm{var}\,[X] = \varsigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2.$$

- For a random sample of observations $x_1, x_2, \ldots, x_N$, the following formula is an estimate of the variance ς^2:

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2.$$

In the case of a sample from a normal distribution this is a particularly good estimate.

- The square root of the variance is the ***standard deviation,*** ς. It is known by physicists as RMS or ***root mean square*** in the equiprobable case:

$$\varsigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}.$$

- Given two random variables X and Y with expected values $\mathcal{E}[X]$ and $\mathcal{E}[Y]$, respectively, the ***covariance*** is a measure of how the values of X and Y are related:

$$\mathrm{Cov}\{X, Y\} = \mathcal{E}(XY) - \mathcal{E}(X)\mathcal{E}(Y).$$

For random vectors $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$ the covariance is the $m \times n$ matrix

$$\mathrm{Cov}\{X, Y\} = \mathcal{E}(XY^T) - \mathcal{E}(X)E(X)^T.$$

The (i, j)th element of this matrix is the covariance between the ith component of X and the jth component of Y.

- In order to estimate the degree of interrelation between variables in a manner not influenced by measurement units, the (Pearson) ***correlation coefficient*** is used:

$$c_{XY} = \frac{\mathrm{Cov}\{X, Y\}}{(\mathrm{var}[X]\,\mathrm{var}[Y])^{\frac{1}{2}}}.$$

Correlation is a measure of the strength of the linear relationship between the random variables; nonlinear ones are not measured satisfactorily.

- If $\mathrm{Cov}\{X, Y\} = 0$, the correlation coefficient is zero and the variables X, Y are ***uncorrelated***.

- The random variables X and Y, with distribution functions $P_X(x)$, $P_Y(y)$ and densities $p_X(x)$, $p_Y(y)$, respectively, are statistically ***independent*** if and only if the combined random variable (X, Y) has a joint cumulative distribution function $P_{X,Y}(x, y) = P_X(x)P_Y(y)$ or

equivalently a joint density $p_{X,Y}(x,y) = p_X(x)p_Y(y)$. The expectation and variance operators have the properties $\mathcal{E}[XY] = \mathcal{E}[X]\,\mathcal{E}[Y]$, $\mathrm{var}[X+Y] = \mathrm{var}[X] + \mathrm{var}[Y]$. It follows that independent random variables have zero covariance.

- Independent variables are uncorrelated, but the opposite is not true unless the variables belong to a normal distribution.

- A random vector $\boldsymbol{w}$ is a ***white noise*** vector if $\mathcal{E}(\boldsymbol{w}) = 0$ and $\mathcal{E}(\boldsymbol{w}\boldsymbol{w}^T) = \varsigma^2 I$. That is, it is a zero mean random vector where all the elements have identical variance. Its autocorrelation matrix is a multiple of the identity matrix; therefore the vector elements are uncorrelated. Note that Gaussian noise $\nLeftrightarrow$ white noise.

- The ***coefficient of variation*** is a normalized measure of the dispersion:
$$c_v = \frac{\varsigma}{\mu}.$$
In signal processing the reciprocal ratio $\frac{\mu}{\varsigma}$ is referred to as the signal to noise ratio.

- The ***coefficient of determination*** R^2 is used in the context of linear regression analysis (statistical modeling), as a measure of how well a linear model fits the data. Given a model M that predicts values M_i, $i = 1, \ldots, N$ for the observations $x_1, x_2, \ldots, x_N$ and the residual vector $\boldsymbol{r} = (M_1 - x_1, \ldots, M_N - x_N)^T$, the most general definition for the coefficient of determination is
$$R^2 = 1 - \frac{\|\boldsymbol{r}\|_2^2}{\sum_{i=1}^N (x_i - \overline{x})^2}.$$
In general, it is an approximation of the unexplained variance, since the second term compares the variance in the model's errors with the total variance of the data $x_1, \ldots, x_N$.

- The ***normal (or Gaussian)*** probability function $N(\mu, \varsigma^2)$ has mean μ and variance ς^2. Its probability density function takes the form
$$p(x) = \frac{1}{\sqrt{2\pi}\varsigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\varsigma}\right)^2\right].$$

- If X_i, $i = 1, \ldots, n$ are random variables with normal distributions $N(0,1)$, then $\sum_{i=1}^n X_i^2$ has a chi-square distribution with n degrees of freedom. Given two independent random variables Y, W with chi-square distributions with m and n degrees of freedom, respectively,

the random variable X,

$$X = \frac{Y/m}{W/n},$$

has an ***F-distribution*** with m and n degrees of freedom, $F(m, n)$.

- The ***Student's*** t_n random variable with n degrees of freedom is defined as

$$t_n = \frac{Z}{\sqrt{\chi_n^2/n}},$$

where Z is a standard normal random variable, χ_n^2 is a chi-square random variable with n degrees of freedom and Z and χ_n^2 are independent.

- The F-distribution becomes relevant when testing hypotheses about the variances of normal distributions. For example, assume that we have two independent random samples (of sizes n_1 and n_2 respectively) from two normal distributions; the ratio of the unbiased estimators s_1 and s_2 of the two variances,

$$\frac{s_1/n_1 - 1}{s_2/n_2 - 1},$$

is distributed according to an F-distribution $F(n_1, n_2)$ if the variances are equal: $\varsigma_1^2 = \varsigma_2^2$.

- A ***time series*** is an ordered sequence of observations. The ordering is usually in time, often in terms of equally spaced time intervals, but it can also be ordered in, for example, space. The time series elements are frequently considered random variables with the same mean and variance.

- A ***periodogram*** is a data analysis technique for examining the frequency domain of an equispaced time series and search for hidden periodicities. Given a time series vector of N observations $x(j)$, the discrete Fourier transform (DFT) is given by a complex vector of length N:

$$X(k) = \sum_{j=1}^{N} x(j)\varpi_N^{(j-1)(k-1)},\ 1 \leq k \leq N,$$

with $\varpi_N = \exp((-2\pi i)/N)$.

- The magnitude squared of the discrete Fourier transform components $|X(k)|^2$ is called the power. The periodogram is a plot of the power components versus the frequencies $\{\frac{1}{N},\ \frac{2}{N}, ...\frac{k}{N}, ...\}$.

D.2 Hypothesis testing

In ***hypothesis testing*** one uses statistics to determine whether a given hypothesis is true. The process consists of four steps:

- Formulate the null hypothesis H_0 and the alternative hypothesis H_a, which are mutually exclusive.
- Identify a test statistics t that can be used to assess the truth of the null hypothesis from the sample data. It can involve means, proportions, standard deviations, etc.
- Determine the corresponding distribution function, assuming that the null hypothesis is true and, after choosing an acceptable ***level of significance*** α (common choices are 0.01, 0.05, 0.1), determine the critical region for t (see details below).
- Compute the t for the observations. If the computed value of the test statistics falls into the critical region, reject H_0.

In other words, the test of a hypothesis on the significance level α is performed by means of a statistic t and critical values $\underline{t}_\alpha$ and $\bar{t}_\alpha$ so that

$$1 - \alpha = \Pr\left\{\underline{t}_\alpha \leq t \leq \bar{t}_\alpha\right\}, \qquad 0 \leq \alpha \leq 1$$

if H_0 holds. The hypothesis H_0 is rejected if t falls outside the range $[\underline{t}_\alpha, \bar{t}_\alpha]$. This guarantees that H_0 will only be erroneously rejected in $100 \times \alpha\%$ of the cases.

For example, assume that we have approximated m observations $y_1, y_2, \ldots, y_m$ with a linear model M_{full} with n terms and we want to know if k of these terms are redundant, i.e., if one could get a good enough model when setting k coefficient of the original model to 0. We have the residual norms ρ_{full} and ρ_{red} for both possible models, with $\rho_{\text{full}} < \rho_{\text{red}}$.

We formulate the hypothesis H_0: the reduced model is good enough, i.e., the reduction in residual when using all n terms is negligible.

Under the assumption that the errors in the data $y_1, y_2, \ldots, y_m$ are normally distributed with constant variance and zero mean, we can choose a statistic based on a proportion of variance estimates:

$$f_{\text{obs}} = \frac{\rho_{\text{red}} - \rho_{\text{full}}}{\rho_{\text{full}}} \frac{m-n}{k}.$$

This statistic follows an F-distribution with k and $m-n$ degrees of freedom: $F(k, m-n)$. The common practice is to denote by f_α the value of the statistic with cumulative probability $F_\alpha(k, m-n) = 1 - \alpha$.

If $f_{\text{obs}} > f_\alpha$, the computed statistic falls into the critical region for the F-distribution and the H_0 hypothesis should be rejected, with a possible error of α, i.e., we do need all the terms of the full model.

References

[1] J. J. Alonso, I. M. Kroo and A. Jameson, "Advanced algorithms for design and optimization of quiet supersonic platforms." AIAA Paper 02-0114, 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 2002.

[2] J. J. Alonso, P. Le Gresley, E. van der Weide and J. Martins, "pyMDO: a framework for high-fidelity multidisciplinary optimization." 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, 2004.

[3] A. Anda and H. Park, "Fast plane rotations with dynamic scaling." SIAM J. Matrix Anal. Appl. **15**:162–174, 1994.

[4] E. Angelosante, D. Giannakis and G. B. Grossi, "Compressed sensing of time-varying signals." Digital Signal Processing, 16th International Conference, Santorini, Greece, 2009.

[5] H. Ashley and M. Landahl, *Aerodynamics of Wings and Bodies.* Dover, New York, 1985.

[6] I. Barrodale and F. D. K. Roberts, "An improved algorithm for discrete ℓ_1 linear approximation." SIAM J. Numer. Anal. **10**:839–848, 1973.

[7] R. Bartels, J. Beaty and B. Barski, *An Introduction to Splines for Use in Computer Graphics and Parametric Modeling.* Morgan Kaufman Publishers, Los Altos, CA, 1987.

[8] A. E. Beaton and J. W. Tukey, "The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data." Technometrics **16**:147–185, 1974.

[9] D. M. Bates and D. G. Watts, *Nonlinear Regression Analysis and Its Applications.* J. Wiley, New York, 1988.

[10] G. M. Baudet, "Asynchronous iterative methods for multiprocessors." J. ACM **15**:226–244, 1978.

[11] A. Beck, A. Ben-Tal and M. Teboulle, "Finding a global optimal solution for a quadratically constrained fractional quadratic problem with applications to the regularized total least squares." SIAM J. Matrix Anal. Appl. **28**:425–445, 2006.

[12] M. Berry, "Large scale sparse singular value computations." International J. Supercomp. Appl. **6**:13–49, 1992.

[13] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences.* McGraw-Hill, New York, 1969.

[14] C. H. Bischof and G. Qintana-Orti, "Computing rank-revealing QR factorization of dense matrices." ACM TOMS **24**:226–253, 1998.

[15] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[16] Å. Björck, "Solving linear least squares problems by Gram-Schmidt orthogonalization." BIT **7**:1–21, 1967.

[17] Å. Björck, "Iterative refinement of linear least squares solutions." BIT **7**:257–278, 1967.

[18] Å. Björck, "Iterative refinement of linear least squares solutions II." BIT **8**:8–30, 1968.

[19] Å. Björck, "Stability analysis of the method of seminormal equations for linear least squares problems." Lin. Alg. Appl. **88/89**:31–48, 1987.

[20] Å. Björck, *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[21] Å. Björck, "The calculation of linear least squares problems." Acta Numerica **13**:1–53, 2004.

[22] Å. Björck and I. S. Duff, "A direct method for the solution of sparse linear least squares problems." Lin. Alg. Appl. **34**:43–67, 1980.

[23] Å. Björck and G. H. Golub, "Iterative refinement of linear least squares solution by Householder transformation." BIT **7**:322–337, 1967.

[24] Å. Björck, E. Grimme and P. Van Dooren, "An implicit shift bidiagonalization algorithm for ill-posed systems." BIT **34**:510–534, 1994.

[25] Å. Björck, P. Heggernes and P. Matstoms, "Methods for large scale total least squares problems." SIAM J. Matrix Anal. Appl. **22**:413–429, 2000.

[26] Å. Björck and C. C. Paige, "Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm." SIAM J. Matrix Anal. **13**:176–190, 1992.

[27] Å. Björck and V. Pereyra, "Solution of Vandermonde systems of equations." Math. Comp. **24**:893–904, 1970.

[28] C. Böckman, "A modification of the trust-region Gauss-Newton method for separable nonlinear least squares problems." J. Math. Systems, Estimation and Control **5**:1–16, 1995.

[29] C. de Boor, *A Practical Guide to Splines*. Applied Mathematical Sciences **27**. Springer, New York, 1994.

[30] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2003.

[31] R. Brent, *Algorithms for Minimization Without Derivatives*. Prentice Hall, Englewood Cliffs, NJ, 1973. Reprinted by Dover, New York, 2002.

[32] D. Calvetti, P. C. Hansen and L. Reichel, "L-curve curvature bounds via Lanczos bidiagonalization." Electr. Transact. on Num. Anal. **14**:134–149, 2002.

[33] D. Calvetti, G. H. Golub and L. Reichel, "Estimation of the L-curve via Lanczos bidiagonalization algorithm for ill-posed systems." BIT **39**:603–619, 1999.

[34] E. J. Candes, "Compressive sampling." Proceedings of the International Congress of Mathematicians, Madrid, Spain, 2006.

[35] E. J. Candes and M. B. Wakin, "People hearing without listening: An introduction to compressive sampling." Signal Processing Magazine, IEEE **25**:21–30, 2008.

[36] L. Carcione, J. Mould, V. Pereyra, D. Powell and G. Wojcik, "Nonlinear inversion of piezoelectrical transducer impedance data." J. Comp. Accoustics **9**:899–910, 2001.

[37] L. Carcione, V. Pereyra and D. Woods, *GO: Global Optimization*. Weidlinger Associates Report, 2005.

[38] R. I. Carmichael and L. I. Erickson, "A higher order panel method for predicting subsonic or supersonic linear potential flow about arbitrary configurations." American Institute of Aeronautics and Astronautics Paper 81–1255, 1981.

[39] M. Chan, "Supersonic aircraft optimization for minimizing drag and sonic boom." Ph.D. Thesis, Stanford University, Stanford, CA, 2003.

[40] T. F. Chan, "Rank revealing QR-factorizations." Lin. Alg. Appl. **88/89**:67–82, 1987.

[41] T. F. Chan and D. E. Foulser, "Effectively well-conditioned linear systems." SIAM J. Sci. Stat. Comput. **9**:963–969, 1988.

[42] T. F. Chan and P. C. Hansen, "Computing truncated SVD least squares solutions by rank revealing QR-factorizations." SIAM J. Sci. Statist. Comput. **11**:519–530, 1991.

[43] D. Chazan and W. L. Miranker, "Chaotic relaxation." Lin. Alg. Appl. **2**:199–222, 1969.

[44] W. Cheney and D. Kincaid, *Numerical Mathematics and Computing.* Brooks/Cole, Belmont, CA, 2007.

[45] S. Choi, J. J. Alonso and H. S. Chung, "Design of low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method." 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Palm Springs, CA, 2004.

[46] H. S. Chung, S. Choi and J. J. Alonso, "Supersonic business jet design using knowledge-based genetic algorithms with adaptive, unstructured grid methodology." AIAA 2003-3791, 21st Applied Aerodynamic Conference, Orlando, Fl., June 2003.

[47] H. S. Chung, "Multidisciplinary design optimization of supersonic business jets using approximation model-based genetic algorithms." Ph.D. Thesis, Stanford University, Stanford, CA, 2004.

[48] J. F. Claerbout and F. Muir, "Robust modeling with erratic data." Geophysics **38**:826–844, 1973.

[49] A. K. Cline, "An elimination method for the solution of linear least squares problems." SIAM J. Numer. Anal. **10:**283–289, 1973.

[50] D. Coleman, P. Holland, N. Kaden, V. Klema and S. C. Peters, "A system of subroutines for iteratively reweighted least squares computations." ACM TOMS **6**:328–336, 1980.

[51] T. F. Coleman and Y. Li, "A globally and quadratically convergent affine scaling method for linear ℓ_1 problems." Mathematical Programming, Series A **56**:189–222, 1992.

[52] T. P. Collignon, "Efficient iterative solution of large linear systems on heterogeneous computing systems." Ph. D. Thesis, TU Delft, The Netherlands, 2011.

[53] T. P. Collignon and M. B. van Gijzen. "Parallel scientific computing on loosely coupled networks of computers." In B. Koren and C. Vuik, editors, *Advanced Computational Methods in Science and Engineering*. Springer Series Lecture Notes in Computational Science and Engineering, **71**:79–106. Springer Verlag, Berlin/Heidelberg, Germany, 2010.

[54] G. Cybenko, "Approximation by superpositions of a sigmoidal function." Math. Control Signals Systems **2**:303–314, 1989.

[55] J. Dahl, P. C. Hansen, S. H. Jensen and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods." Numer. Algo. **53**:67–92, 2010.

[56] J. Dahl and L. Vanderberghe, CVXOPT: *A Python Package for Convex Optimization.* http://abel.ee.ucla.edu/cvxopt, 2012.

[57] G. Dahlquist and Å. Björck, *Numerical Methods in Scientific Computing.* SIAM, Philadelphia, 2008.

[58] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection." ACM TOMS **38**:1–25, 2011.

[59] K. Deb, A. Pratap, S. Agrawal and T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II.* Technical Report No. 2000001. Indian Institute of Technology, Kanpur, India, 2000.

[60] P. Deift, J. Demmel, L.-C. Li and C. Tomei, "The bidiagonal singular value decomposition and Hamiltonian mechanics." SIAM J. Numer. Anal. **28**:1463–1516, 1991.

[61] R. S. Dembo, S. C. Eisenstat and T. Steihaug, "Inexact Newton methods." SIAM J. Numer. Anal. **19**:400–408, 1982.

[62] C. J. Demeure and L. L. Scharf, "Fast least squares solution of Vandermonde systems of equations." Acoustics, Speech and Signal Processing **4**:2198–2210, 1989.

[63] J. W. Demmel, *Applied Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[64] J. Demmel, Y. Hida, W. Riedy and X. S. Li, "Extra-precise iterative refinement for overdetermined least squares problems." ACM TOMS **35**:1–32, 2009.

[65] J. Dennis, D. M. Gay and R. Welsch, "Algorithm 573 NL2SOL – An adaptive nonlinear least-squares algorithm." ACM TOMS **7**:369–383, 1981.

[66] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* SIAM, Philadelphia, 1996.

[67] J. E. Dennis and H. F. Walker, "Convergence theorems for least-change secant update methods." SIAM J. Num. Anal. **18**:949–987, 1981.

[68] A. P. Dempster, N. M. Laird and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm (with discussion)." Journal of the Royal Statistical Society B **39**:1–38, 1977.

[69] P. Dierckx, *Curve and Surface Fitting with Splines.* Clarendon Press, Oxford, 1993.

[70] J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *LINPACK User's Guide.* SIAM, Philadelphia, 1979.

[71] N. Draper and H. Smith, *Applied Regression Analysis.* J. Wiley, New York, 1981.

[72] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank." Psychometrika **1**:211–218, 1936.

[73] L. Eldén, "A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems." BIT **24**:467–472, 1984.

[74] L. Eldén, "Perturbation theory for the linear least squares problem with linear equality constraints." BIT **17**:338–350, 1980.

[75] L. Eldén, *Matrix Methods in Data Mining and Pattern Recognition.* SIAM, Philadelphia, 2007.

[76] J. Eriksson, P.-Å. Wedin, M. E. Gulliksson and I. Söderkvist, "Regularization methods for uniformly rank-deficient nonlinear least-squares problems." J. Optimiz. Theory and Appl. **127**:1–26, 2005.

[77] J. Eriksson and P.-Å. Wedin, "Truncated Gauss-Newton algorithms for ill-conditioned nonlinear least squares problems." Optimiz. Meth. and Software **19**:721–737, 2004.

[78] R. D. Fierro, P. C. Hansen and P. S. K. Hansen, "UTV tools: MATLAB templates for rank-revealing UTV decompositions." Numer. Algo. **20**:165–194, 1999. The software is available from: htpp://www.netlib.org/numeralgo.

[79] P. M. Fitzpatrick, *Advanced Calculus.* Second edition Brooks/Cole, Belmont, CA, 2006.

[80] D. Fong and M. A. Saunders, "LSMR: An iterative algorithm for sparse least-squares problems." SIAM J. Sci. Comput. **33**:2950–2971, 2011.

[81] G. E. Forsythe, "Generation and use of orthogonal polynomials for data-fitting with a digital computer." J. SIAM **5**:74–88, 1957.

[82] J. Fox and S. Weisberg, *Robust Regression in R*, An Appendix to *An R Companion to Applied Regression*, Second edition. http://socserv.socsci.mcmaster.ca/jfox/Books /Companion/appendix.html.

[83] C. F. Gauss, *Theory of the Combination of Observations Least Subject to Errors. Parts 1 and 2, Supplement,* G. W. Stewart. SIAM, Philadelphia, 1995.

[84] D. M. Gay, *Usage Summary for Selected Optimization Routines.* AT&T Bell Labs. Comp. Sc. Tech. Report, 1990.

[85] D. M. Gay and L. Kaufman, *Tradeoffs in Algorithms for Separable Nonlinear Least Squares.* AT&T Bell Labs. Num. Anal. Manuscript, 90–11, 1990.

[86] D. M. Gay, *NSF and NSG; PORT Library.* AT&T Bell Labs. http://www.netlib.org/port, 1997.

[87] P. E. Gill, G. H. Golub, W. Murray and M. A. Saunders, "Methods for modifying matrix factorisations." Math. Comp. **28**:505–535, 1974.

[88] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders and M. H. Wright, *Users Guide for LSSOL (Version 1.0): A Fortran Package for Constrained Linear Least-Squares and Convex Quadratic Programming.* Report 86-1 Department of Operation Research, Stanford University, CA, 1986.

[89] P. E. Gill, W. Murray and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization." SIAM Rev. **47**:99–131, 2005.

[90] P. E. Gill, W. Murray, M. A. Saunders and M. H. Wright, "Maintaining LU factors of a general sparse matrix." Linear Algebra and its Applications **88–89**:239–270, 1987.

[91] P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization.* Academic Press, 1981.

[92] G. H. Golub, "Numerical methods for solving linear least squares problems." Numer. Math. **7**:206–16, 1965.

[93] G. H. Golub, P. C. Hansen and D. P. O'Leary, "Tikhonov regularization and total least squares." SIAM J. Matrix Anal. Appl. **21**:185–194, 1999.

[94] G. H. Golub, M. Heath and G. Wahba, "Generalized cross-validation as a method for choosing a good ridge parameter." Technometrics **21**:215–223, 1979.

[95] G. H. Golub, A. Hoffman and G. W. Stewart, "A generalization of the Eckhard-Young-Mirsky matrix approximation theorem." Linear Algebra Appl. **88/89**:317–327, 1987.

[96] G. H. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix." SIAM J, Numer. Anal. Ser. B **2**:205–224, 1965.

[97] G. H. Golub, V. Klema and G. W. Stewart, *Rank Degeneracy and Least Squares.* Report TR-456, Computer Science Department, University of Maryland, College Park, 1977.

[98] G. H. Golub and R. Le Veque, "Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems." Proceedings of the Army Numerical Analysis and Computers Conference, White Sands Missile Range, New Mexico, pp. 1–12, 1979.

[99] G. H. Golub and U. von Matt, "Tikhonov regularization for large problems." Workshop on Scientific Computing, Ed. G. H. Golub, S. H. Lui, F. Luk and R. Plemmons, Springer, New York, 1997.

[100] G. H. Golub and V. Pereyra, *The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate.* STAN-CS-72-261, Stanford University, Computer Sciences Department, 1972. (It contains the original VARPRO computer code.)

[101] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate." SIAM J. Numer. Anal. **10**:413–432, 1973.

[102] G. H. Golub and V. Pereyra, "Differentiation of pseudoinverses, separable nonlinear least squares problems, and other tales." Proceedings MRC Seminar on Generalized Inverses and Their Applications, Ed. Z. Nashed, pp. 302–324, Academic Press, NY, 1976.

[103] G. H. Golub and V. Pereyra, "Separable nonlinear least squares: The variable projection method and its applications." Inverse Problems **19**:R1–R26, 2003.

[104] G. H. Golub and J. M. Varah, "On a characterization of the best ℓ_2-scaling of a matrix." SIAM J. Numer. Anal. **11**:472–279, 1974.

[105] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Third edition, John Hopkins University Press, Baltimore, 1996.

[106] G. H. Golub and C. F. Van Loan, "An analysis of the total least squares problem." SIAM J. Numer. Anal. **17**:883–893, 1980.

[107] G. H. Golub and J. H. Wilkinson, "Note on iterative refinement of least squares solutions." Numer. Math. **9**:139–148, 1966.

[108] J. F. Grcar, *Optimal Sensitivity Analysis of Linear Least Squares.* Lawrence Berkeley National Laboratory, Report LBNL-52434, 2003.

[109] J. F. Grcar, "Mathematicians of Gaussian elimination." Notices of the AMS **58**:782–792, 2011.

[110] J. F. Grcar, "John von Neumann's analysis of Gaussian elimination and the origins of modern Numerical Analysis." SIAM Review **53**:607–682, 2011.

[111] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Other Titles in Applied Mathematics **105**. Second edition, SIAM, Philadelphia, 2008.

[112] E. Grosse, "Tensor spline approximation." Linear Algebra Appl. **34**:29–41, 1980.

[113] I. Gutman, V. Pereyra and H. D. Scolnik, "Least squares estimation for a class of nonlinear models." Technometrics **15**:209–218, 1973.

[114] Y. Y. Haimes, L. S. Ladon and D. A. Wismer, "On a bicriterion formulation of the problem of integrated system identification and system optimization." IEEE Trans. on Systems, Man and Cybernetics **1**:296–297, 1971.

[115] S. Hammarling, "A note on modifications to the Givens plane rotations." J. Inst. Math. Applic. **13**:215–218, 1974.

[116] S. Hammarling, *A Survey of Numerical Aspects of Plane Rotations.* University of Manchester, MIMS Eprint 2008.69, 1977.

[117] M. Hanke and P. C. Hansen, "Regularization methods for large-scale problems." Surv. Math. Ind. **3**:253–315, 1993.

[118] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems.* SIAM, Philadelphia, 1998.

[119] P. C. Hansen, "The L-curve and its use in the numerical treatment of inverse problems." In *Comp. Inverse Problems in Electrocardiology.* Ed. P. Johnston, pp. 119–142, WIT Press, Southampton, UK, 2001.

[120] P. C. Hansen, "Regularization tools version 4.0 for MATLAB 7.3." Numer. Algorithms **46**:189–194, 2007.

[121] P. C. Hansen, *Discrete Inverse Problems – Insight and Algorithms.* SIAM, Philadelphia, 2010.

[122] P. C. Hansen, M. Kilmer and R. H. Kjeldsen, "Exploiting residual information in the parameter choice for discrete ill-posed problems." BIT **46**:41–59, 2006.

[123] P. C. Hansen and M. Saxild-Hansen, "AIR tools – A MATLAB package of algebraic iterative reconstruction methods." J. Comp. Appl. Math. **236**:2167–2178, 2011.

[124] P. C. Hansen and P. Yalamov, "Computing symmetric rank-revealing decompositions via triangular factorization." SIAM J. Matrix Anal. Appl. **23**:443–458, 2001.

[125] J. G. Hayes, ed., *Numerical Approximation to Functions and Data.* Athlone Press, London, 1970.

[126] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems." J. Res. Nat. Stan. B. **49**:409–432, 1952.

[127] N. Higham, "Analysis of the Cholesky decomposition of a semi-definite matrix." In *Reliable Numerical Computing*, ed. M. G. Cox and S. J. Hammarling, Oxford University Press, London, 1990.

[128] N. Higham, *Accuracy and Stability of Numerical Algorithms.* Second edition, SIAM, Philadelphia, 2002.

[129] H. P. Hong and C. T. Pan, "Rank-revealing QR factorization and SVD." Math. Comp. **58**:213–232, 1992.

[130] P. J. Huber and E. M. Ronchetti, *Robust Statistics.* Second edition, J. Wiley, NJ, 2009.

[131] R. Horst, P. M. Pardalos and N. Van Thoai, *Introduction to Global Optimization.* Second edition, Springer, New York, 2000.

[132] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice: Comparison of software packages for regression models with missing variables." The American Statistician **55**, 2011.

[133] I. C. F. Ipsen and C. D. Meyer, "The idea behind Krylov methods." Amer. Math. Monthly **105**:889–899, 1998.

[134] R. A. Johnson, *Miller & Freund's Probability and Statistics for Engineers.* Pearson Prentice Hall, Upper Saddle River, NJ, 2005.

[135] P. Jones, *Data Tables of Global and Hemispheric Temperature Anomalies*
http://cdiac.esd.ornl.gov/trends/temp/jonescru/data.html.

[136] T. L. Jordan, "Experiments on error growth associated with some linear least squares procedures." Math. Comp. **22**:579–588, 1968.

[137] D. L. Jupp, "Approximation to data by splines with free knots." SIAM J. Numer. Anal. **15**:328, 1978.

[138] D. L. Jupp and K. Vozoff, "Stable iterative methods for the inversion of geophysical data." J. R. Astr. Soc. **42**:957–976, 1975.

[139] W. Karush, "Minima of functions of several variables with inequalities as side constraints." M. Sc. Dissertation. Department of Mathematics, University of Chicago, Chicago, Illinois, 1939.

[140] L. Kaufman, "A variable projection method for solving nonlinear least squares problems." BIT **15**:49–57, 1975.

[141] L. Kaufman and G. Sylvester, "Separable nonlinear least squares with multiple right-hand sides." SIAM J. Matrix Anal. and Appl. **13**:68–89, 1992.

[142] L. Kaufman and V. Pereyra, "A method for separable nonlinear least squares problems with separable equality constraints." SIAM J. Numer. Anal. **15**:12–20, 1978.

[143] H. B. Keller and V. Pereyra, *Finite Difference Solution of Two-Point BVP.* Preprint 69, Department of Mathematics, University of Southern California, 1976.

[144] M. E. Kilmer and D. P. O'Leary, "Choosing regularization parameters in iterative methods for ill-posed problems." SIAM. J. Matrix Anal. Appl. **22**:1204–1221, 2001.

[145] H. Kim, G. H. Golub and H. Park, "Missing value estimation for DNA microarray expression data: Least squares imputation." In Proceedings of CSB'2004, Stanford, CA, pp. 572–573, 2004.

[146] S. Kotz, N. L. Johnson and A. B. Read, *Encyclopedia of Statistical Sciences.* **6**. Wiley-Interscience, New York, 1985.

[147] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming.* Proceedings of 2nd Berkeley Symposium, 481–492, University of California Press, Berkeley, 1951.

[148] *Lancelot: Nonlinear Programming Code*
http://www.numerical.rl.ac.uk/lancelot/

[149] K. Lange, *Numerical Analysis for Statisticians.* Second edition, Springer, New York, 2010.

[150] C. Lawson and R. Hanson, *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[151] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares." Quart. J. App. Math. **2**:164–168, 1948.

[152] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data.* Second edition. J. Wiley, Hoboken, NJ, 2002.

[153] K. Madsen and H. B. Nielsen, "A finite smoothing algorithm for linear ℓ_1 estimation." SIAM J. Optimiz. **3**:223–235, 1993.

[154] K. Madsen and H. B. Nielsen, *Introduction to Optimization and Data Fitting.* Lecture notes, Informatics and Mathematical Modelling, Technical University of Denmark, Lyngby, 2008.

[155] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods." Signal Processsing **87**:2283–2302, 2007.

[156] O. Mate, "Missing value problem." Master's Thesis, Stanford University, Stanford, CA, 2007.

[157] Matrix Market. http://math.nist.gov/MatrixMarket

[158] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, 2000.

[159] J.-C. Miellou, "Iterations chaotiques a retards." C. R. Acad. Sci. Paris **278**:957–960, 1974.

[160] J.-C. Miellou, "Algorithmes de relaxation chaotique a retards." RAIRO **R1**:55–82, 1975.

[161] K. M. Miettinen, *Nonlinear Multiobjective Optimization.* Kluwers Academic, Boston, 1999.

[162] L. Miranian and M. Gu, "Strong rank revealing LU factorizations." Lin. Alg. Appl. **367**:1–16, 2003.

[163] Software for multiple imputation. http://www.stat.psu.edu/~jls/misoftwa.html

[164] M. Mohlenkamp and M. C. Pereyra, *Wavelets, Their Friends, and What They Can Do For You.* EMS Lecture Series in Mathematics, European Mathematical Society, Zurich, Switzerland, 2008.

[165] J. Moré and S. J. Wright, *Optimization Software Guide.* SIAM, Philadelphia, 1993.

[166] A. Nedic, D. P. Bertsekas and V. S. Borkar, "Distributed asynchronous incremental subgradient methods." Studies in Computational Mathematics **8**:381–407, 2001.

[167] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Methods in Convex Programming.* Studies in Applied Mathematics 13. SIAM, Philadelphia, 1994.

[168] L. Ngia, "System modeling using basis functions and applications to echo cancellation." Ph.D. Thesis, Chalmers Institute of Technology, Sweden, Goteborg, 2000.

[169] L. Ngia, *Separable Nonlinear Least Squares Methods for On-Line Estimation of Neural Nets Hammerstein Models.* Department of Signals and Systems, Chalmers Institute of Technology, Sweden, 2001.

[170] J. Nocedal and S. J. Wright, *Numerical Optimization.* Springer, New York, second edition, 2006.

[171] D. P. O'Leary, "Robust regression computation using iteratively reweighted least squares." SIAM J. Matrix. Anal. Appl. **22**:466–480, 1990.

[172] D. P. O'Leary and B. W. Rust, "Variable projection for nonlinear least squares problems." Submitted for publication in Computational Optimization and Applications (2011). Also at: http://www.cs.umd.edu/users/oleary/software/varpro.pdf

[173] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[174] M. R. Osborne, "Some special nonlinear least squares problems." SIAM J. Numer. Anal. **12**:571–592, 1975.

[175] M. R. Osborne and G. K. Smyth, "A modified Prony algorithm for fitting functions defined by difference equations." SIAM J. Sci. Comp. **12**:362–382, 1991.

[176] M. R. Osborne and G. K. Smyth, "A modified Prony algorithm for exponential function fitting." SIAM J. Sci. Comp. **16**:119–138, 1995.

[177] M. R. Osborne, "Separable least squares, variable projections, and the Gauss-Newton algorithm." ETNA **28**:1–15, 2007.

[178] A. Ostrowski, "Determinanten mit ueberwiegender Hauptdiagonale and die absolute Konvergenz von linearen Iterationsprozessen." Comm. Math. Helv. **30**:175–210, 1955.

[179] C. C. Paige and M. A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares." ACM TOMS **8**:43–71, 1982.

[180] C. C. Paige and Z. Strakos, "Core problems in linear algebraic systems." SIAM J. Matrix Anal. Appl. **27**:861–875, 2006.

[181] R. Parisi, E. D. Di Claudio, G. Orlandi and B. D. Rao, "A generalized learning paradigm exploiting the structure of feedforward neural networks." IEEE Transactions on Neural Networks **7**:1450–1460, 1996.

[182] Y. C. Pati and P. S. Krishnaprasad, "Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations." IEEE Trans. Neural Networks **4**:73–85, 1993.

[183] V. Pereyra, "Accelerating the convergence of discretization algorithms." SIAM J. Numer. Anal. **4**:508–533, 1967.

[184] V. Pereyra, "Iterative methods for solving nonlinear least squares problems." SIAM J. Numer. Anal. **4**:27–36, 1967.

[185] V. Pereyra, "Stability of general systems of linear equations." Aeq. Math. **2**:194–206, 1969.

[186] V. Pereyra, "Stabilizing linear least squares problems." Proc. IFIP, Suppl. **68**:119–121, 1969.

[187] V. Pereyra, "Modeling, ray tracing, and block nonlinear travel-time inversion in 3D." Pure and App. Geoph. **48**:345–386, 1995.

[188] V. Pereyra, "Asynchronous distributed solution of large scale nonlinear inversion problems." J. App. Numer. Math. **30**:31–40, 1999.

[189] V. Pereyra, "Ray tracing methods for inverse problems." Invited topical review. Inverse Problems **16:**R1–R35, 2000.

[190] V. Pereyra, "Fast computation of equispaced Pareto manifolds and Pareto fronts for multiobjective optimization problems." Math. Comp. in Simulation **79**:1935–1947, 2009.

[191] V. Pereyra, M. Koshy and J. Meza, "Asynchronous global optimization techniques for medium and large inversion problems." SEG Annual Meeting Extended Abstracts **65**:1091–1094, 1995.

[192] V. Pereyra and P. Reynolds, "Application of optimization techniques to finite element analysis of piezocomposite devices." IEEE Ultrasonics Symposium, Montreal, CANADA, 2004.

[193] V. Pereyra and J. B. Rosen, *Computation of the Pseudoinverse of a Matrix of Unknown Rank.* Report CS13, 39 pp. Computer Science Department, Stanford University, Stanford, CA, 1964.

[194] V. Pereyra, M. Saunders and J. Castillo, *Equispaced Pareto Front Construction for Constrained Biobjective Optimization.* SOL Report-2010-1, Stanford University, Stanford, CA, 2010. Also CSRCR2009-05, San Diego State University, 2009. In Press, Mathematical and Computer Modelling, 2012.

[195] V. Pereyra and G. Scherer, "Efficient computer manipulation of tensor products with applications in multidimensional approximation." Math. Comp. **27**:595–605, 1973.

[196] V. Pereyra and G. Scherer, "Least squares scattered data fitting by truncated SVD." Applied Numer. Math. **40**:73–86, 2002.

[197] V. Pereyra and G. Scherer, "Large scale least squares data fitting." Applied Numer. Math. **44**:225–239, 2002.

[198] V. Pereyra and G. Scherer, "Exponential data fitting." In *Exponential Data Fitting and Its Applications*, Ed. V. Pereyra and G. Scherer. Bentham Books, Oak Park, IL, 2010.

[199] V. Pereyra, G. Scherer and F. Wong, "Variable projections neural network training." Mathematics and Computers in Simulation **73**:231–243, 2006.

[200] V. Pereyra, G. Wojcik, D. Powell, C. Purcell and L. Carcione, "Folded shell projectors and virtual optimization." US Navy Workshop on Acoustic Transduction Materials and Devices, Baltimore, 2001.

[201] G. Peters and J. H. Wilkinson, "The least squares problem and pseudo-inverses." The Computer Journal **13**:309–316, 1969.

[202] M. J. D. Powell, *Approximation Theory and Methods*. Cambridge University Press, New York, 1981.

[203] L. Prechelt, *Proben 1-A Set of Benchmark Neural Network Problems and Benchmarking Rules*. Technical Report 21, Fakultaet fuer Informatik, Universitaet Karlsruhe, 1994.

[204] Baron Gaspard Riche de Prony, "Essai experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de l'alkool, a differentes temperatures." J. Ecole Polyt. **1**:24–76, 1795.

[205] *PZFLEX: Weidlinger Associates Inc. Finite Element Code to Simulate Piezoelectric Phenomena*. http://www.wai.com, 2006.

[206] L. Reichel, "Fast QR decomposition of Vandermonde-like matrices and polynomial least squares approximations." SIAM J. Matrix Anal. Appl. **12**:552–564, 1991.

[207] R. A. Renaut and H. Guo, "Efficient algorithms for solution of regularized total least squares." SIAM J. Matrix Anal. Appl. **26**:457–476, 2005.

[208] J. R. Rice, "Experiments on Gram-Schmidt orthogonalization." Math. Comp. **20**:325–328, 1966.

[209] J. B. Rosen, "Minimum and basic solutions to singular systems." J. SIAM **12**:156–162, 1964.

[210] J. B. Rosen and J. Kreuser, "A gradient projection algorithm for nonlinear constraints." In Numerical Methods for Non-Linear Optimization. Ed. F. A. Lootsma, Academic Press, New York, pp. 297–300, 1972.

[211] J. Rosenfeld, *A Case Study on Programming for Parallel Processors*. Research Report RC-1864, IBM Watson Research Center, Yorktown Heights, New York, 1967.

[212] Å. Ruhe and P-Å. Wedin, "Algorithms for separable non-linear least squares problems." SIAM Rev. **22**:318–337, 1980.

[213] B. W. Rust, "Fitting nature's basis functions." Parts I-IV, Computing in Sci. and Eng. 2001–03.

[214] B. W. Rust, http://math.nist.gov/~BRust/Gallery.html, 2011.

[215] B. W. Rust, *Truncating the Singular Value Decomposition for Ill-Posed Problems.* Tech. Report NISTIR 6131, Mathematics and Computer Sciences Division, National Institute of Standards and Technology, 1998.

[216] B. W. Rust and D. P. O'Leary, "Residual periodograms for choosing regularization parameters for ill-posed problems." Inverse Problems **24**, 2008.

[217] S. Saarinen, R. Bramley and G. Cybenko, "Ill-conditioning in neural network training problems." SIAM J. Sci. Stat. Comput. **14**:693–714, 1993.

[218] S. A. Savari and D. P. Bertsekas, "Finite termination of asynchronous iterative algorithms." Parallel Comp. **22**:39–56, 1996.

[219] J. A. Scales, A. Gersztenkorn and S. Treitel, "Fast l_p solution of large, sparse, linear systems: application to seismic travel time tomography." J. Comp. Physics **75**:314–333, 1988.

[220] J. L. Schafer, *Analysis of Incomplete Multivariate Data.* Chapman & Hall, London, 1997.

[221] S. Schechter, "Relaxation methods for linear equations." Comm. Pure Appl. Math. **12**:313–335, 1959.

[222] M. E. Schlesinger and N. Ramankutty, "An oscillation in the global climate system of period 65–70 years." Nature **367**:723–726, 1994.

[223] G. A. F. Seber and C. J. Wild, *Nonlinear Regression.* J. Wiley, New York, 1989.

[224] R. Sheriff and L. Geldart, *Exploration Seismology.* Cambridge University Press, second edition, Cambridge, 1995.

[225] D. Sima, S. Van Huffel and G. Golub, "Regularized total least squares based on quadratic eigenvalue problem solvers." BIT **44**:793–812, 2004.

[226] Spline2. http://www.structureandchange.3me.tudelft.nl/.

[227] J. Sjöberg and M. Viberg, "Separable non-linear least squares minimization–possible improvements for neural net fitting." IEEE Workshop in Neural Networks for Signal Processing, Amelia Island Plantation, FL, 1997.

[228] N. Srivastava, R. Suaya, V. Pereyra and K. Banerjee, "Accurate calculations of the high-frequency impedance matrix for VLSI interconnects and inductors above a multi-layer substrate: A VARPRO success story." In *Exponential Data Fitting and Its Applications*, Eds. V. Pereyra and G. Scherer. Bentham Books, Oak Park, IL, 2010.

[229] T. Steihaug and Y. Yalcinkaya, "Asynchronous methods in least squares: An example of deteriorating convergence." Proc. 15 IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics, Berlin, Germany, 1997.

[230] G. W. Stewart, "Rank degeneracy." SIAM J. Sci. Stat. Comput. **5**:403–413, 1984.

[231] G. W. Stewart, "On the invariance of perturbed null vectors under column scaling." Numer. Math. **44**:61–65, 1984.

[232] G. W. Stewart, *Matrix Algorithms. Volume I: Basic Decompositions.* SIAM, Philadelphia, 1998.

[233] T. Strutz, *Data Fitting and Uncertainty.* Vieweg+Teubner Verlag, Wiesbaden, 2011.

[234] B. J. Thijsse and B. Rust, "Freestyle data fitting and global temperatures." Computing in Sci. and Eng. pp. 49–59, 2008.

[235] R. Tibshirani, "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society. Series B (Methodological) **58**:267–288, 1996.

[236] A. N. Tikhonov, "On the stability of inverse problems." Dokl. Akad. Nauk SSSR **39**:195–198, 1943.

[237] L. N. Trefethen and D. Bau, *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[238] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein and R. B. Altman, "Missing value estimation methods for DNA microarrays." Bioinformatics **17**:520–525, 2001.

[239] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem. Computational Aspects and Analysis,* SIAM, Philadelphia, 1991.

[240] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions." SIAM J. Sci. Comp. **31**:890–912, 2008.

[241] E. van den Berg and M. P. Friedlander, "Sparse optimization with least-squares constraints." SIAM J. Optim. **21**:1201–1229, 2011.

[242] A. van den Bos, *Parameter Estimation for Scientists and Engineers.* Wiley-Interscience, Hoboken, NJ, 2007.

[243] A. van der Sluis, "Condition numbers and equilibration of matrices." Numer. Math. **14**:14–23, 1969.

[244] J. W. van der Veen, R. de Beer, P. R. Luyten and D. Van Ormondt, "Accurate quantification of in vivo 31P NMR signals using the variable projection method and prior knowledge." Magn. Reson. Med. **6**:92–98, 1988.

[245] L. Vanhamme, A. van den Boogaart and S. Van Huffel, "Improved method for accurate and efficient quantification of MRS data with use of prior knowledge." J. Magn. Reson. **129**:35–43, 1997.

[246] L. Vanhamme, S. Van Huffel, P. Van Hecke and D. van Ormondt, "Time-domain quantification of series of biomedical magnetic resonance spectroscopy signals." J. Magn. Reson. **140**:120–130, 1999.

[247] L. Vanhamme, T. Sundin, P. Van Hecke, S. Van Huffel and R. Pintelon, "Frequency-selective quantification of biomedical magnetic resonance spectroscopy data." J. Magn. Reson. **143**:1–16, 2000.

[248] S. Van Huffel, 'Partial singular value decomposition algorithm." J. Comp. Appl. Math. **33**:105–112, 1990.

[249] B. Walden, R. Karlsson and J.-G. Sun, "Optimal backward perturbation bounds for the linear least squares problem." Numer. Linear Algebra Appl. **2**:271–286, 2005.

[250] I. Wasito and B. Mirkin, "Nearest neighbors in least squares data imputation algorithms with different missing patterns." Comp. Stat. & Data Analysis **50**: 926–949, 2006.

[251] D. S. Watkins, *Fundamentals of Matrix Computations.* J. Wiley, New York, 2002.

[252] K. Weigl and M. Berthod, *Neural Networks as Dynamical Bases in Function Space.* Report 2124, INRIA, Programe Robotique, Image et Vision, Sophia-Antipolis, France, 1993.

[253] K. Weigl, G. Giraudon and M. Berthod, *Application of Projection Learning to the Detection of Urban Areas in SPOT Satellite Images.* Report #2143, INRIA, Programe Robotique, Image et Vision, Sophia-Antipolis, France, 1993.

[254] K. Weigl and M. Berthod, "Projection learning: Alternative approach to the computation of the projection." Proceedings of the European Symposium on Artificial Neural Networks pp. 19–24, Brussels, Belgium, 1994.

[255] J. H. Wilkinson, *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1963. Reprint Dover, New York, 1994.

[256] H. Wold and E. Lyttkens, "Nonlinear iterative partial least squares (NIPALS) estimation procedures." Bull. ISI **43**:29–51, 1969.

[257] S. Wold, A. Ruhe, H. Wold and W. J. Dunn, "The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses." SIAM J. Sci. Stat. Comput. **5**:735–743, 1984.

[258] R. Wolke and H. Schwetlick, "Iteratively reweighted least squares: Algorithms, convergence analysis, and numerical comparisons." SIAM J. Sci. Stat. Comput. **9**:907–921, 1988.

[259] S. J. Wright, *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, 1997.

[260] S. J. Wright, J. N. Holt, "Algorithms for non-linear least squares with linear inequality constraints." SIAM J. Sci. and Stat. Comput. **6**:1033–1048, 1985.

[261] P. Zadunaisky and V. Pereyra, "On the convergence and precision of a process of successive differential corrections." Proc. IFIPS **65**, 1965.

[262] H. Zha, "Singular values of a classical matrix." The American Mathematical Monthly **104**:172–173, 1997.

[263] Netlib Repository at UTK and ORNL. http://www.netlib.org.

[264] NEOS Wiki. http://www.neos-guide.org.

[265] NIST/SEMATECH e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/strd/nls/nls_info.shtml, 2011.

Index