J. Webster (ed.), *Wiley Encyclopedia of Electrical and Electronics Engineering*

LEAST-SQUARES APPROXIMATIONS

The method of least squares dates back about two hundred years. The main stimulus for its invention and development was provided by studies in astronomy. The first to publish on the method was the French mathematician Adrien-Marie Legendre, who studied the orbits of comets. It seems, however, that the first to invent the least-squares approximation method was the German mathematician Karl Friedrich Gauss, who used it in 1795 (Legendre discovered the method independently). Ever since, the method has been one of the basic tools for data analysis by scientists and engineers, and in its class it is the most popular.

In practice, data measured for studying physical phenomena are often erroneous due to the lack of absolutely accurate measurement devices. For example, when Legendre and Gauss studied the motion of planets, it was important to estimate their orbits accurately from imperfect observations. In general, mathematical models are built for observed phenomena, which are represented by functions with unknown parameters. These models have to approximate observed data as closely as possible, so the objective is to fit the model to the data in the best possible way. A criterion for goodness of fit for this purpose is often the total sum of squared residuals. The smaller the sum, the better the model. Clearly, then, for a given model it is important to estimate its parameters that yield the smallest sum of squared residuals; therefore we want to apply least-squares estimation methods.

In this article we discuss selected topics in the field of least-squares approximation. We start with preliminaries related to quantifying the approximations and definitions of the concept of norms. Then we introduce the notions of linear independence and inner product, and describe some basic linear-algebra concepts. The least-squares method is first explained with a couple of simple examples of curve fitting. It is followed by presentation of the general least-squares problem and approaches for solving it. Many important practical cases of linear least-squares involve polynomial fitting and spline interpolation; thus, we present some basic information for their use. Next, the focus of our attention shifts to nonlinear least-squares methods, and we address methods with reduced complexity as well as iterative techniques. The next topic, sequential least-squares estimation, involves implementation of the least-squares method recursively in time. The last two sections are on predictive least squares and the bootstrap method. The former is important for choosing the correct model from a set of candidates, and the latter, for assessing the accuracy of the least-squares estimates.

Preliminaries

In general, three main components are needed to specify every approximation problem: (1) the function $y(t)$, which is to be approximated over a given closed interval $[a,b]$, (2) the class Ψ of approximating functions $\psi(t)$, and (3) the norm $\|\cdot\|$ that gives some measure of magnitude of functions. The goal of best approximation is to find a function $\hat{\psi}$ such that

$$\|y(t) - \hat{\psi}(t)\| \leq \|y(t) - \psi(t)\| \qquad \text{for all} \quad \psi(t) \in \Psi \tag{1}$$

If the above is satisfied, the function $\hat{\psi}$ is the *best approximation* to $y(t)$ from the class Ψ with respect to the norm $\|\cdot\|$.

The class Ψ is called a *real linear space* if for any two functions $\psi_1(t),\psi_2(t) \in \Psi$, it also contains $\theta_1\psi_1(t) + \theta_2\psi_2(t)$ for any real θ_1 and θ_2. A linear space Ψ_p of finite dimension p can be defined given a set of (constituent) *basis functions* $h_i(t) \in \Psi$, $i = 1,2,\ldots, p$. Any function $\psi(t) \in \Psi_p$ can be represented as a linear combination of the basis functions $h_i(t)$:

$$\psi(t) = \sum_{i=1}^{p} \theta_i h_i(t) \tag{2}$$

for any real θ_i.

A typical example for a linear space is the set of polynomials of finite degree. They are very convenient for approximating functions over bounded-support domains. Any continuous function defined over a bounded support (or closed interval, in the single-variable case) can be approximated to any level of accuracy with a polynomial of sufficiently large degree (Weierstrass's theorem).

As mentioned previously, the norm is the third important component needed for specifying an approximation problem. It is the criterion that determines the goodness of each approximating function. A function that is a good approximant in one norm may turn out to be a bad approximant under a different norm. The following are some possible norms for a function $\varepsilon(t)$, defined over the finite closed interval $[a,b]$ (in the subsequent sections the function $\varepsilon(t)$ denotes the approximation error, and $w(t)$ is some nonnegative "weight function" defined on $[a,b]$):

$$L_\infty: \qquad \|\varepsilon\|_\infty = \max_{a \le t \le b} |\varepsilon(t)| \tag{3}$$

$$L_1: \qquad \|\varepsilon\|_1 = \int_a^b |\varepsilon(t)|\, dt \tag{4}$$

$$\text{weighted } L_1: \qquad \|\varepsilon\|_{1,w} = \int_a^b w(t)|\varepsilon(t)|\, dt \tag{5}$$

$$L_2 \text{ (least squares)}: \qquad \|\varepsilon\|_2 = \sqrt{\int_a^b |\varepsilon(t)|^2\, dt} \tag{6}$$

$$\text{weighted } L_2 \text{ (weighted least squares)}: \qquad \|\varepsilon\|_{2,w} = \sqrt{\int_a^b w(t)|\varepsilon(t)|^2\, dt} \tag{7}$$

The discrete versions of the above equations correspond to the situations involving a set of N distinct points $t_1, t_2, \ldots, t_N$ and a set of N nonnegative weight factors $w_1, w_2, \ldots, w_N$:

$$L_\infty: \qquad \|\varepsilon\|_\infty = \max_{1\le n\le N} |\varepsilon[t_n]| \tag{8}$$

$$L_1: \qquad \|\varepsilon\|_1 = \sum_{n=1}^{N} |\varepsilon[t_n]| \tag{9}$$

$$\text{weighted } L_1: \qquad \|\varepsilon\|_{1,w} = \sum_{n=1}^{N} w_n|\varepsilon[t_n]| \tag{10}$$

$$L_2 \text{ (least squares)}: \qquad \|\varepsilon\|_2 = \sqrt{\sum_{n=1}^{N} |\varepsilon[t_n]|^2} \tag{11}$$

$$\text{weighted } L_2 \text{ (weighted least squares)}: \qquad \|\varepsilon\|_{2,w} = \sqrt{\sum_{n=1}^{N} w_n|\varepsilon[t_n]|^2}. \tag{12}$$

In our presentation, we study continuous- and discrete-variable functions. We discriminate them in the notation by using $\xi(t)$ for the continuous case, and $\xi[t_n]$, or simply $\xi[n]$, for the discrete case, where $t \in [a,b] \subset \Re$ and $n = \{1,2,\ldots, N\}$. In the continuous case, the given (approximated) function $y(t)$ and the approximating functions $\psi(t)$ of the class Ψ must be defined on the interval $[a,b]$, so that the chosen norm $\|y - \psi\|$ is also defined on the same interval. Similarly, in the discrete case, $y[t_n]$, $\psi[t_n]$, and $\|y - \psi\|$ must be defined at the N distinct support points. If the best approximant $\hat{\psi}$ in the discrete case satisfies $\|y - \hat{\psi}\| = 0$, then $\hat{\psi}[t_n] = y[t_n]$ for $n = 1,2,\ldots, N$, in which case it is said that $\hat{\psi}$ *interpolates* y at the points t_n. This sort of approximation problem is called an *interpolation problem*.

More Basic Concepts

Linear Independence. The set of functions $h_i(t)$ is said to be *linearly independent* on a given support S_t if

$$\sum_{i=1}^{p} \theta_i h_i(t) \equiv 0 \quad \text{for all } t \in S_t \quad \text{implies that} \quad \theta_i = 0, \quad i = 1, 2, \cdots, p$$

As an example, the set of functions $h_i(t) = t^{i-1}$, $i = 1,2,\ldots, p$, is linearly independent on the support $S_t = [a,b]$ where a and b are real and $a < b$. If, however, the support is $S_t = \{t_1, t_2, \ldots, t_N\}$, then the set is linearly

independent if and ony if $N \geq p$, because otherwise the sum of squared approximation errors could be made equal to zero at the N support points without necessarily implying that the coefficients θ_i are zero.

Inner Product. An *inner product* of two (real) functions $h_1(t)$ and $h_2(t)$ whose $\boldsymbol{L}_2$ norms are finite is defined as

$$\langle h_1, h_2 \rangle = \int_a^b h_1(t) h_2(t)\, dt \tag{13}$$

In the discrete case associated with the set of points $\{t_1, t_2, \ldots, t_N\}$ the inner product will be defined as

$$\langle h_1, h_2 \rangle = \sum_{n=1}^{N} h_1[t_n] h_2[t_n] \tag{14}$$

Note that the least-squares ($\boldsymbol{L}_2$) norm (squared) of a given function is simply its inner product with itself, i.e., $\|\varepsilon\|^2{}_2 = \langle \varepsilon, \varepsilon \rangle$.

Two functions $h_1(t)$ and $h_2(t)$ are said to be *orthogonal* if $\langle h_1, h_2 \rangle = 0$. An *orthogonal system* is defined as a set of functions $\{h_i(t)\}$, $i = 1, 2, \ldots, p$, that satisfy

$$\langle h_i, h_j \rangle = 0 \text{ if } i \neq j \text{ and } h_k(t) \neq 0 \text{ on the support } S_t, \qquad i, j, k = 1, 2, \cdots, p$$

It can be shown that every orthogonal system is linearly independent on the support S_t. An orthogonal system is called *orthonormal* if $\langle h_i, h_i \rangle = 1$, $i = 1, 2, \ldots, p$.

Basic Linear-Algebra Concepts

Every function can be represented by a vector of some sort. In the discrete case, this can be a vector of samples of the function taken at distinct values of t. A polynomial function can be represented by a vector of its polynomial coefficients. A periodic function can be represented by its Fourier series coefficients. The field of linear algebra covers the concepts associated with vectors and *vector spaces*. In this section we will cover a few basic notions. Almost regularly, the vector–matrix (discrete-valued) concepts have their equivalents in concepts associated with functions. These equivalencies can usually be established by simply substituting the word "vector" for "function." For instance, a set of vectors $\boldsymbol{h}_i$, $i = 1, 2, \ldots, p$, is said to be linearly independent if

$$\sum_{i=1}^{p} \theta_i \boldsymbol{h}_i = \boldsymbol{0}$$

holds only with $\theta_1 = \theta_2 = \cdots = \theta_p = 0$.

The set of all N-dimensional vectors is an *N-dimensional vector space*. Equivalently to the concept of linear (function) spaces described previously, if $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are members of this vector space, then so are $\boldsymbol{h}_1 + \boldsymbol{h}_2$ and $\theta\, \boldsymbol{h}_1$. (Note that there is no need for using "linear" in the case of vector spaces.) We call these two conditions *closure under vector addition and scalar–vector multiplication*. If a subset P of a vector space $\boldsymbol{Q}$ is closed under vector addition and scalar–vector multiplication, then P is called a *subspace*. The maximal number of linearly independent vectors in P is called the *dimension* of the subset P. A maximal-size set of linearly independent vectors in a subspace P is a *basis* for P. Given a p-dimensional subspace P and a set

of $k < p$ linearly independent member vectors, there are always $p - k$ additional vectors in P such that the concatenated set of p vectors represents a basis for P. If a set of vectors $\boldsymbol{h}_i$, $i = 1,2,\ldots, p$, constitutes a basis for the subspace P, then any vector $\boldsymbol{u} \in P$ can be represented as $\boldsymbol{u} = \sum_{i=1}^{p} \theta_i \boldsymbol{h}_i$. The concepts of vector norms, vector inner products, and orthogonal (and orthonormal) vectors, can be defined analogously to the same concepts described previously in the case of functions.

The subspace formed by the set of all linear combinations of the vectors $\boldsymbol{h}_i$, $i = 1, 2, \ldots, k$, is called the *span* of this vector set. The dimension of this subspace is less than or equal to k. Given an $N \times p$ matrix $\boldsymbol{H}$, the subspace spanned by its column vectors is called the *range* or the *column space*, and the subspace spanned by its row vectors is called the *row space*. For any matrix, the dimension of the row space is equal to the dimension of the column space, and this number is called the *rank* of the matrix. An $N \times p$ matrix $\boldsymbol{H}$ is of *full rank* if rank $(\boldsymbol{H}) = \min(N, p)$, and it is *rank-deficient* if rank $(\boldsymbol{H}) < \min(N, p)$. A square $N \times N$ matrix is *singular* if rank $(\boldsymbol{H}) < N$, and *nonsingular* if rank $(\boldsymbol{H}) = N$. A square matrix $\boldsymbol{Q}$ is *orthogonal* if $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Matrices with this property have extensive use in many approaches to solving least-squares problems. Lawson and Hanson (1) give a succinct and clear description of more linear-algebra concepts pertaining to the least-squares problem and its solution.

Least-Squares Curve Fitting

We continue our discussion of the least-squares problem by introducing some of its simplest and clearest manifestations as applied to the curve-fitting problem.

Straight-Line Fitting. Experiments in science and engineering produce data points that are subsequently used to derive relationships between variables in the observed models. In particular, a set of distinct data points $(t_1, y[1])$, $(t_2, y[2]),\ldots, (t_N, y[N])$ needs to be *fitted* with a function $f(t)$ that relates these two variables. Depending on our confidence in the exactness of the measured data points, we may approach this problem in two main ways. If the data points are known to be highly accurate (i.e., the measuring devices add little or no errors that are not accounted for in the model, or the level of external noise in the system is insignificant), then interpolation is the best way to go.

Interpolation will attempt to fit a curve that goes straight through all of the data points. If, however, the data points are known to be insufficiently accurate, interpolation will almost regularly give unsatisfactory results. Intuitively, attempting to fit a curve through data points that are likely to be randomly positioned around their "accurate" positions is bound to produce overly complex approximating functions that poorly describe the real underlying phenomenon. In these cases, the true value of $f(t_n)$ satisfies $f(t_n) = y[n] + \varepsilon[n]$, where $\varepsilon[n]$ denotes the measurement *error* $\varepsilon[n] = f(t_n) - y[n]$, which is also called the *residual*. Each of the norms listed previously under the discrete case can be associated with this residual to serve as a quality measure for the fit. Least-squares methods exploit the $\boldsymbol{L}_2$ norm.

First, we cover the simplest case of linear approximation. How do we find a best approximation of form $f(t) = \theta_1 t + \theta_2$ that goes near a set of data points $(t_1, y[1])$, $(t_2, y[2]),\ldots, (t_N, y[N])$ scattered in the (t, y) two-dimensional space? Our goal is to position this line "as close as possible" to all data points contained in the set. For convenience, our measuring stick for goodness will be the square of the L_2 norm of the residual,

$$\|\varepsilon\|^2 = \sum_{n=1}^{N} |\varepsilon[n]|^2 = \sum_{n=1}^{N} \big[f(t_n) - y[n]\big]^2 = \sum_{n=1}^{N} (\theta_1 t_n + \theta_2 - y[n])^2 = J(\theta_1, \theta_2). \quad (15)$$

The best linear approximation (i.e., the best straight line) is the one whose parameter pair (θ_1, θ_2) minimizes the error function $J(\theta_1, \theta_2)$; it is denoted by $(\hat{\theta}_1, \hat{\theta}_2)$. Hence, the approximation problem is transformed

into a minimization problem in the parameter space spanned by the parameters θ_1 and θ_2. At the point that minimizes the value of $J(\theta_1,\theta_2)$, the partial derivatives $\partial J/\partial\theta_1$ and $\partial J/\partial\theta_2$ are both zero:

$$\frac{\partial J(\theta_1,\theta_2)}{\partial\theta_1}=\sum_{n=1}^{N}2(\theta_1 t_n+\theta_2-y[t_n])t_n=2\sum_{n=1}^{N}\left(\theta_1 t_n^2+\theta_2 t_n-y[t_n]t_n\right)=0 \quad (16)$$

$$\frac{\partial J(\theta_1,\theta_2)}{\partial\theta_1}=\sum_{n=1}^{N}2(\theta_1 t_n+\theta_2-y[t_n])=2\sum_{n=1}^{N}\left(\theta_1 t_n+\theta_2-y[t_n]\right)=0 \quad (17)$$

The above equations can be arranged into a system of two equations with two unknowns, which—in the context of least-squares approximations—are referred to as the *normal equations*:

$$\left(\sum_{n=1}^{N}t_n^2\right)\theta_1+\left(\sum_{n=1}^{N}t_n\right)\theta_2=\sum_{n=1}^{N}t_n y[t_n] \quad (18)$$

$$\left(\sum_{n=1}^{N}t_n\right)\theta_1+N\theta_2=\sum_{n=1}^{N}y[t_n] \quad (19)$$

The solution to this system is given by

$$\hat{\theta}_1=\frac{N\sum t_n y[t_n]-\sum t_n\sum y[t_n]}{N\sum t_n^2-\left(\sum t_n\right)^2} \quad (20)$$

$$\hat{\theta}_2=\frac{\sum t_n^2\sum y[t_n]-\sum t_n\sum t_n y[t_n]}{N\sum t_n^2-\left(\sum t_n\right)^2} \quad (21)$$

where for simplicity $\Sigma(\cdot)$ denotes $\sum_{n=1}^{N}$.

Nonlinear Fitting Functions. The same least-squares method used to fit a straight line through a set of data points can be extended to nonlinear cases as well. For instance, consider the function $f(t)=\theta t^c$, where c is some known constant. Given a set of N data points and following the same least-squares method, we need to find a value of the parameter θ that minimizes the function

$$J(\theta)=\sum_{n=1}^{N}\left(\theta t_n^c-y[t_n]\right)^2 \quad (22)$$

Here we need to solve $\partial J/\partial\theta = 2\sum_{n=1}^{N}(\theta t^{2c}{}_n - t^c{}_n y[t_n]) = 0$, which yields the solution

$$\hat{\theta} = \frac{\sum_{n=1}^{N} t_n^c y[t_n]}{\sum_{n=1}^{N} t_n^{2c}} \tag{23}$$

One familiar example ($c = 2$) covered by this power fit is finding the acceleration of gravity from a set of time and distance measurements.

The General Linear Least-Squares Problem

As a generalization of the above curve-fitting discussions, the linear least-squares problem can be presented as follows. We need to model a function $y(t)$ over a given interval with an optimal (best in the least-squares sense) linear combination of p known basis functions $h_i(t)$, $i = 1, 2, \ldots, p$:

$$y(t) = \sum_{i=1}^{p} \theta_i h_i(t) + \varepsilon(t) \tag{24}$$

where $\varepsilon(t)$ is the modeling (fitting) error and θ_i are the unknown modeling coefficients. In the straight-line fitting discussion, we used two basis functions (a linear and a constant function), while in the nonlinear curve-fitting discussion we used a single (nonlinear) basis function. Regardless of whether the basis functions themselves are linear or nonlinear, the linearity of the least-squares procedure stems from the fact that the basis functions are elements of a linear space, i.e., the model function in Eq. (24) is linear in the unknown coefficients θ_i.

In the discrete case where the functions are only known at N distinct points, the linear least-squares problem is as follows:

$$\begin{aligned} y[1] &= \sum_{i=1}^{p} \theta_i h_i[1] + \varepsilon[1] \\ y[2] &= \sum_{i=1}^{p} \theta_i h_i[2] + \varepsilon[2] \\ &\vdots \\ y[N] &= \sum_{i=1}^{p} \theta_i h_i[N] + \varepsilon[N] \end{aligned} \tag{25}$$

that is, $y[n] = \Sigma^{p}{}_{i=1}\, \theta_i h_i[n] + \varepsilon[n]$ for $n = 1, 2, \ldots, N$. In matrix form this can be written as

$$\begin{bmatrix} y[1] \\ y[2] \\ \vdots \\ y[N] \end{bmatrix} = \begin{bmatrix} h_1[1] & h_2[1] & \cdots & h_p[1] \\ h_1[2] & h_2[2] & \cdots & h_p[2] \\ \vdots & \vdots & \ddots & \\ h_1[N] & h_2[N] & \cdots & h_p[N] \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix} + \begin{bmatrix} \varepsilon[1] \\ \varepsilon[2] \\ \vdots \\ \varepsilon[N] \end{bmatrix} \tag{26}$$

or

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{27}$$

where $\boldsymbol{H}$ is an $N \times p$ matrix and the compositions of the vectors and matrices involved are obvious from the above. This data-modeling problem corresponds to selecting the basis coefficients so that the data model best represents the measured data in a least-squares sense.

We seek the vector θ that minimizes the square of the L_2 norm of the criterion (modeling error)

$$\hat{\theta} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\theta}\|^2 \tag{28}$$

In nonmatrix form, the function to be minimized is

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} \left(y[n] - \sum_{i=1}^{p} \theta_i h_i[n] \right)^2 \tag{29}$$

After taking the partial derivatives $\partial J(\theta)/\partial\theta_i$, $i = 1, 2, \ldots, p$, and setting them equal to zero, we get the following system of equations:

$$\sum_{n=1}^{N} \left(y[n] - \sum_{i=1}^{p} \theta_i h_i[n] \right) h_l[n] = 0 \qquad \text{for} \quad l = 1, 2, \ldots, p \tag{30}$$

Notice the introduction of a new index l, which is not to be confused with the index i. After interchanging the order of summations, we get p (linear) normal equations with p unknown coefficients θ_i:

$$\sum_{i=1}^{p} \left(\sum_{n=1}^{N} h_l[n] h_i[n] \right) \theta_i = \sum_{n=1}^{N} h_l[n] y[n] \qquad \text{for} \quad l = 1, 2, \cdots, p \tag{31}$$

Equivalently, the matrix form of the normal equations can be presented as $(\boldsymbol{H}^{\mathrm{T}}\,\boldsymbol{H})\theta = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{y}$, where $\boldsymbol{H}^{\mathrm{T}}$ denotes the transpose of the matrix $\boldsymbol{H}$. The solution for the unknown coefficient vector θ is

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{y} \tag{32}$$

which is referred to as a least-squares solution. Without going into derivations, it should be noted here that in the case of complex data modeling (where we allow for complex-valued elements of θ, $\boldsymbol{H}$, and/or $\boldsymbol{y}$), the above formula becomes

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{H}^{*}\boldsymbol{H})^{-1}\boldsymbol{H}^{*}\boldsymbol{y} \tag{33}$$

where $\boldsymbol{H}^{*}$ denotes the *complex conjugate transpose* of $\boldsymbol{H}$.

Note that the solutions yielded by Eqs. (32) and (33) are by no means unique. (They are unique if and only if $\boldsymbol{H}$ is of full rank.) It is possible that there exists a whole set of least-squares solutions that are associated with the same (and unique) minimal (least-squares) value.

Solving the Linear Least-Squares Problem

Orthogonal Decomposition. The solutions presented in Eqs. (32) and (33) are oftentimes computationally costly (because of high-order matrix inversions) and/or very sensitive to small perturbations in the observed model. There are several ways to reach a least-squares solution more efficiently. Some of them are direct, and some of them are iterative. Most of them take advantage of the important property of orthogonal matrices that they preserve the L_2 norm under multiplication. In other words, given an $N \times N$ orthogonal matrix $\boldsymbol{Q}$ and the N-vector ε,

$$\|\boldsymbol{Q}\,\varepsilon\| = \|\varepsilon\|, \qquad \text{or equivalently,} \quad \|\boldsymbol{Q}\,(\boldsymbol{y} - \boldsymbol{H}\,\boldsymbol{\theta})\| = \|(\boldsymbol{y} - \boldsymbol{H}\,\boldsymbol{\theta})\| \tag{34}$$

Using this property, we can present the least-squares problem in a modified form. Following Lawson and Hanson (1), we first assume that the $N \times p$ matrix $\boldsymbol{H}$ ($N \geq p$) is of rank k and that it can be decomposed as

$$\boldsymbol{H} = \boldsymbol{A}\boldsymbol{R}\boldsymbol{B}^{\mathrm{T}} \tag{35}$$

where $\boldsymbol{A}$ is an orthogonal $N \times N$ matrix, $\boldsymbol{B}$ is an orthogonal $p \times p$ matrix, and $\boldsymbol{R}$ is of the form

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{36}$$

where $\boldsymbol{R}_{11}$ is a $k \times k$ matrix of rank k. Next, we introduce the new N-vector

$$\boldsymbol{u} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{bmatrix} \begin{matrix} \text{size } k \\ \text{size } N-k \end{matrix} \tag{37}$$

and the new p-vector

$$\lambda = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{\theta} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \begin{matrix} \text{size } k \\ \text{size } p-k \end{matrix} \tag{38}$$

We define $\hat{\lambda}_1$ to be the unique solution of

$$\boldsymbol{R}_{11}\lambda_1 = \boldsymbol{u}_1 \tag{39}$$

Then:

(1) All solutions to the least-squares problem of minimizing $\|\boldsymbol{y} - \boldsymbol{H}\theta\|$ can be presented as

$$\hat{\theta} = \boldsymbol{B}\begin{bmatrix} \hat{\lambda}_1 \\ \lambda_2 \end{bmatrix}, \qquad \text{where } \lambda_2 \text{ is any arbitrary } (p-k)\text{-vector} \tag{40}$$

(2) Any $\hat{\theta}$ so defined is associated with the same residual (error) vector

$$\varepsilon = \boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\theta}} = \boldsymbol{A}\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{u}_2\end{bmatrix} \tag{41}$$

(3) The norm of ε satisfies

$$\|\varepsilon\| = \|\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\theta}}\| = \|\boldsymbol{u}_2\| \tag{42}$$

(4) The unique *solution of minimum length* is

$$\hat{\boldsymbol{\theta}} = \boldsymbol{B}\begin{bmatrix}\hat{\lambda}_1\\ \boldsymbol{0}\end{bmatrix} \tag{43}$$

The proofs of the above four statements can be found in Lawson and Hanson (1). The decomposition $\boldsymbol{H} = \boldsymbol{ARB}^{\mathrm{T}}$ is called an *orthogonal decomposition* of $\boldsymbol{H}$, and it is by no means unique. Some of the most widely known and applied orthogonal decompositions are the *QR decomposition* and the *singular value decomposition* (*SVD*). The *discrete Fourier transform* can also be applied. In general, regardless of the orthogonal decomposition employed, every least-squares problem has a *unique solution of minimum length*, a *unique set of all solutions*, and a *unique minimum residual value*. In the special case when rank $(\boldsymbol{H}) \equiv k = p$, the solution to the least-squares problem is itself unique.

Weighted Least Squares

Oftentimes, in practice, it is desirable to attach separate weights to each of the terms in the norm summation for the modeling error. In the continuous case, a given continuous weighting function may be associated with the norm formula. Cases where this is needed are the ones where each data point is known to be associated with a different level of certainty, accuracy, or reliability. Intuitively, we would like the members of the error-norm sum stemming from more reliable data to make larger contributions to the total. This is where weighted least squares comes into play. In the more general complex data-modeling case, the error function to be minimized is

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\theta})^{*}\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{H}\boldsymbol{\theta}) \tag{44}$$

where the $N \times N$ weighting matrix $\boldsymbol{W}$ is positive definite and Hermitian (i.e., $\boldsymbol{W}* = \boldsymbol{W}$). If $\boldsymbol{W}$ is equal to the identity matrix, the model reduces to the classical (unweighted) least-squares case. Most often, $\boldsymbol{W}$ is diagonal with the weighting coefficients populating the main diagonal. Skipping the rather straightforward derivation (which is readily found in numerical analysis and optimization-related textbooks) the solution to the weighted least-squares problem is

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{H}^{*}\boldsymbol{W}\boldsymbol{H})^{-1}\boldsymbol{H}^{*}\boldsymbol{W}\boldsymbol{y} \tag{45}$$

Polynomial Fitting and Spline Interpolation

After analyzing the general least-squares problem and discussing some solution approaches, we return to some more complex manifestations of the curve-fitting problem such as polynomial fitting and spline interpolation.

One convenient way of modeling empirical data is by fitting a polynomial of degree p through N data points $(t_n, y[t_n])$, n = 1, 2, ..., N. We have already presented a special case when we addressed the fitting of a quadratic function through a set of data points. The linear least-squares approach described above can be implemented towards the solution of the generalized polynomial fitting problem by using the basis functions $h_i(t) = t^i$, $i = 0, 1, \ldots, p$. The approximating function is

$$\psi(t) = \sum_{i=0}^{p} \theta_i h_i(t) = \theta_0 + \theta_1 t + \theta_2 t^2 + \cdots + \theta_p t^p \qquad (46)$$

Note that there are $p + 1$ unknown coefficients. Of course, all the previous discussion for the general least-squares problem and its solution applies to this special case.

If the number of data points, N, is less than or equal to the number of unknown polynomial coefficients, $p + 1$, there will always exist a polynomial curve (of degree p) that can bring the modeling error down to zero (i.e., it will manage to go straight through the data points). This special subcase of polynomial fitting is called *polynomial interpolation*.

There are difficulties associated with polynomial fitting, and much attention needs to be paid to the nature of the modeled phenomenon and the empirical data. Unless we are sure that the data points actually lie on the polynomial curve (polynomial interpolation problem) and unless *a priori* knowledge is available for the polynomial degree of the phenomenon, polynomial fitting can often lead to unsatisfactory solutions. Note that a polynomial of degree p can have $p - 1$ local extrema. This means that the least-squares polynomial will have more oscillations as p increases. The modeling errors (measured at the data points) may still be zero or very small (or, at least, be minimal in a least-squares sense), but that fact itself will not guarantee "nice" behavior of the polynomial curve in the space between the data points.

Example: As an illustration of the above discussion, Fig. 1 shows a polynomial fitting ($p + 1 < N$) example, while Fig. 2 shows a polynomial interpolation ($p + 1 \geq N$) example. All polynomial curves shown are least-squares solutions for a specific case where the number of data points is $N = 5$. The data points are $\{(t_n, y[t_n])\} = \{(1,1), (2,3), (3,2), (4,5), (5,7)\}$. Figure 1 shows the $1 \leq p \leq 3$ (fitting) cases, and Fig. 2 the $4 \leq p \leq 6$ (interpolation) cases, where p is the degree of the polynomial. Note how the magnitude of the oscillations increases with p. A remedy for this so-called *polynomial wiggle* phenomenon can be found in *spline interpolation*.

The spline interpolation approach tries to piece together lower-degree polynomial curves $\psi_k(t)$, each of which interpolates the data over predetermined abscissa segments. The combined curve $\psi(t)$, defined over the whole relevant abscissa range, is called a *spline*. The connection points between the segments are called *knots*. The simplest and most trivial spline interpolation case is when the polynomials are linear (i.e., of degree 1). This amounts to simply connecting adjacent data points with straight lines. Most widely used, especially in the computer graphics industry, are the *cubic splines*. They are smooth interpolating curves without excessive oscillations. Intuitively, they are a good choice because they are the lowest-degree polynomials with nonzero first and second derivatives. A low polynomial degree minimizes unwanted oscillations (wiggles) while the (existing, hence controllable) first and second derivatives enforce the desirable behavior around the knots.

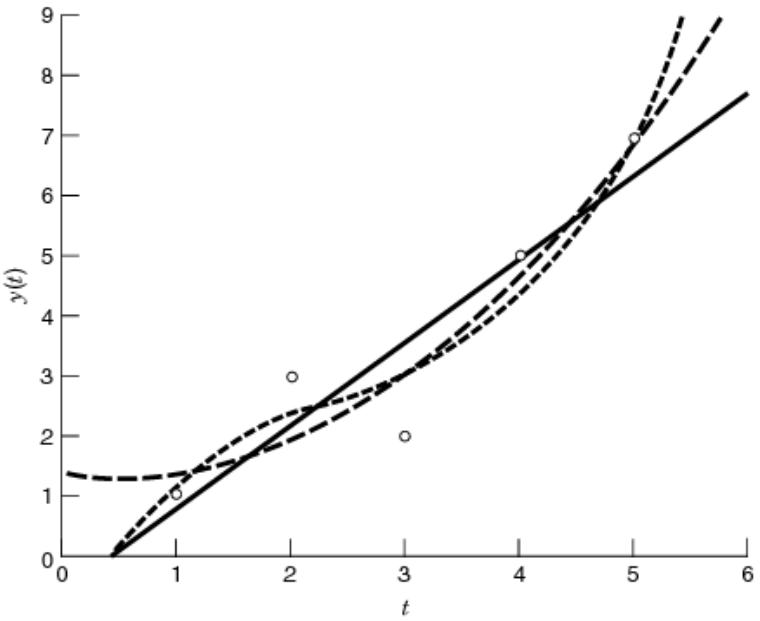

Fig. 1. Polynomial fitting example ($p + 1 < N$). The $N = 5$ data points are $\{(t_k, y_k)\} = \{(1,1), (2,3), (3,2), (4,5), (5,7)\}$. Full line ($p = 1$) represents $y(t) = 1.4000t - 0.6000$. Dashed line ($p = 2$) represents $y(t) = 0.2857t^2 - 0.3143t + 1.4000$. Dotted line ($p = 3$) represents $y(t) = 0.1667t^3 - 1.2143t^2 + 3.6190t + 1.4000$.

Consider $N + 1$ knots $(t_0, y[t_0])$, $(t_1, y[t_1])$,..., $(t_N, y[t_N])$ such that $t_0 < t_1 \cdots < t_N$. Also consider N cubic polynomials

$$\psi_n(t) = c_{n,0} + c_{n,1}(t - t_n) + c_{n,2}(t - t_n)^2 + c_{n,3}(t - t_n)^3 \qquad (47)$$

for $t \in [t_n, t_{n+1}]$ and $n = 0, 1, \ldots, N - 1$. A cubic spline $\psi(t)$ will be formed if the following four properties are satisfied:

$$\psi(t_n) = y[t_n] \qquad \text{for} \quad n = 0, 1, \ldots, N-1 \qquad (48)$$

$$\psi_n(t_{n+1}) = \psi_{n+1}(t_{n+1}) \qquad \text{for} \quad n = 0, 1, \ldots, N-2 \qquad (49)$$

$$\left.\frac{d\psi_n(t)}{dt}\right|_{t=t_{n+1}} = \left.\frac{d\psi_{n+1}(t)}{dt}\right|_{t=t_{n+1}} \qquad \text{for} \quad n = 0, 1, \ldots, N-2 \qquad (50)$$

$$\left.\frac{d^2\psi_n(t)}{dt^2}\right|_{t=t_{n+1}} = \left.\frac{d^2\psi_{n+1}(t)}{dt^2}\right|_{t=t_{n+1}} \qquad \text{for} \quad n = 0, 1, \ldots, N-2 \qquad (51)$$

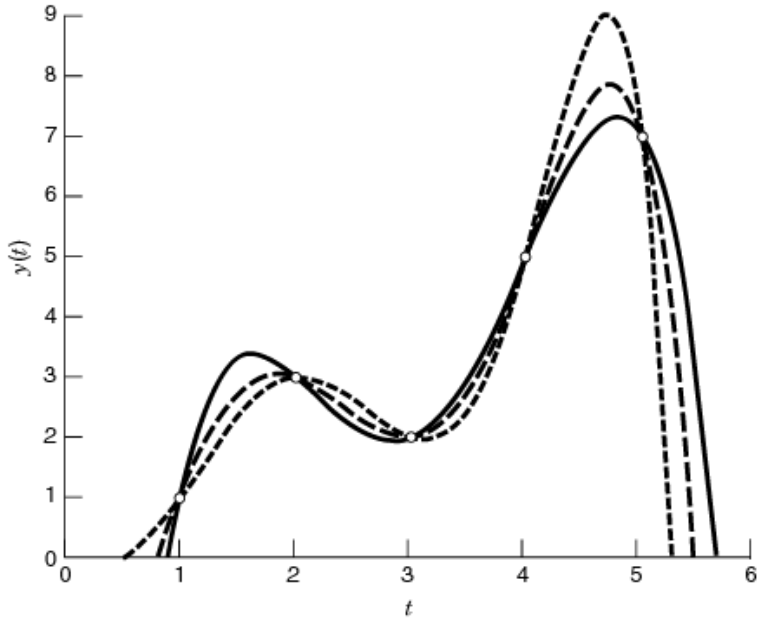

Fig. 2. Polynomial interpolation example ($p + 1 \geq N$). The data points are the same as in Fig. 1. Full line ($p = 4$) represents $y(t) = -0.5000t^4 + 6.1667t^3 - 26.0000t^2 + 44.3333t - 23.0000$. Dashed line ($p = 5$) represents $y(t) = -0.1618t^5 + 1.9270t^4 - 7.5864t^3 + 10.4051t^2 - 3.5839$. Dotted line ($p = 6$) represents $y(t) = -0.0625t^6 + 0.7458t^5 - 2.9375t^4 + 3.9375t^3 - 0.6833t$.

The property (48) ensures that the spline passes through the knots (data points). The property (49) ensures that the spline is a continuous function. The property (50) guarantees that the spline is smooth around the knots. The property (51) further limits the curvature of the spline around the knots.

The goal of the spline interpolation procedure is to find the set of cubic polynomials that satisfy the above properties. Each of the N cubic polynomials $\psi_n(t)$ has four coefficients, which results in a total of $4N$ unknowns. The property (48) provides $N + 1$ equations. Each of the properties (49), (50), and (51) provides $N - 1$ conditions, bringing the total to $N + 1 + 3(N - 1) = 4N - 2$ equations. Two more conditions can be added to control the spline's derivatives at the endpoints $(t_0, y[t_0])$ and $(t_N, y[t_N])$, thus bringing the number of equations to $4N$, which equals the number of unknown coefficients. Obviously, different endpoint conditions will produce correspondingly different end-segment polynomials. For an in-depth coverage of splines see Dierckx (2) and the bibliography therein.

Nonlinear Least Squares

In many problems the data $\boldsymbol{y}$ are modeled as a nonlinear function of unknown parameters θ:

$$\boldsymbol{y} = \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{\varepsilon} \tag{52}$$

where $\boldsymbol{y}$ is an $N \times 1$ vector of observed samples, $\boldsymbol{g}(\cdot)$ is a nonlinear function in the parameters θ, and ε is an $N \times 1$ vector of errors. The parameter vector belongs to the parameter space $\Theta \subset \Re^p$; that is θ is a p-dimensional

vector. The expression in Eq. (52) can also be written as

$$y[n] = g(\boldsymbol{\theta}, n) + \varepsilon[n] \qquad \text{for} \quad n = 1, 2, \ldots, N \tag{53}$$

and the least-squares estimate of θ, $\hat{\theta}$, is obtained by minimizing the error sum of squares

$$J(\boldsymbol{\theta}) = (\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{\theta}))^{\mathrm{T}}(\boldsymbol{y} - \boldsymbol{g}(\boldsymbol{\theta})) \tag{54}$$

or

$$J(\boldsymbol{\theta}) = \sum_{n=1}^{N} (y[n] - g(\boldsymbol{\theta}, n))^2 \tag{55}$$

over $\theta \in \Theta$, i.e.,

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{56}$$

As opposed to the linear least-squares problem, the estimation of θ when $\boldsymbol{g}(\cdot)$ is a nonlinear function may be very difficult. The general methods for finding $\hat{\theta}$ are based on iterative numerical techniques, of which the best known are the Gauss–Newton method and the Newton–Raphson algorithm. Before we proceed with their description, it is important to comment on approaches that reduce the complexity of the problem and allow for easier implementation of the least-squares estimation.

Methods with Reduced Complexity. In some situations it is possible to transform the original nonlinear problem to a linear one by using a one-to-one transformation. In that case, the original parameters θ are transformed to α via

$$\boldsymbol{\alpha} = \boldsymbol{q}(\boldsymbol{\theta}) \tag{57}$$

where $\boldsymbol{q}(\cdot)$ is the transformation, so that

$$\boldsymbol{g}(\boldsymbol{\theta}(\boldsymbol{\alpha})) = \boldsymbol{g}(q^{-1}(\boldsymbol{\alpha})) = \boldsymbol{H}\,\boldsymbol{\alpha} \tag{58}$$

Then one can apply first the linear least-squares approach to estimate $\hat{\alpha}$ and follow it with the transformation

$$\hat{\boldsymbol{\theta}} = \boldsymbol{q}^{-1}(\hat{\boldsymbol{\alpha}}) \tag{59}$$

to obtain the desired estimates. Unfortunately, it is usually difficult to find a function $\boldsymbol{q}(\cdot)$ that converts the nonlinear problem to a linear one. To demonstrate the method, we provide an example that is of interest in many signal-processing applications, as presented in Kay (3).

Example: Let the data $y[n]$ be modeled as a sinusoid with known frequency f and unknown amplitude A and phase φ:

$$y[n] = A \sin(2\pi f n + \varphi) + \varepsilon[n] \qquad \text{for} \quad n = 1, 2, \ldots, N \tag{60}$$

and let the objective be to find the least-squares estimates of A and φ. Obviously, one of the parameters, φ, is nonlinear, which means that the straightforward approach would be to apply a nonlinear least-squares method. The alternative is to transform A and φ to a new set of parameters B_1 and B_2, which appear in the model of the data as linear parameters. The transformation is given by

$$B_1 = A\cos\varphi \tag{61}$$

$$B_2 = A\sin\varphi \tag{62}$$

and the model becomes

$$y[n] = B_1\,\sin(2\pi fn) + B_2\,\cos(2\pi fn) + \varepsilon[n] \qquad \text{for} \quad n = 1, 2, \ldots, N \tag{63}$$

The parameters B_1 and B_2 can now easily be estimated, and once they are obtained, the original parameters A and φ are found from

$$\hat{A} = \hat{B}_1^2 + \hat{B}_2^2 \tag{64}$$

$$\hat{\varphi} = \arctan\left(\frac{\hat{B}_2}{\hat{B}_1}\right) \tag{65}$$

Another approach that may reduce the complexity of the problem is based on the concept of separability, as described by Seber and Wild (4). Namely, in some problems the parameters θ can be partitioned, $\theta = [\beta, \gamma]$, so that the minimization of the criterion $J\,(\beta, \gamma)$ with respect to β is easy. The idea of the approach is to estimate β first and then proceed with estimating γ by minimizing $J\,(\,\hat{\gamma}, \gamma)$. For instance, when the dimension of θ is p and the dimensions of β and γ are p_β and p_γ, respectively, it is clear that the dimension of the parameter space over which a nonlinear least-squares procedure has to be applied is reduced from p to p_γ.

Example: The data represent a sinusoid as in Eq. (63). Let the unknowns be the amplitudes of the quadrature components B_1 and B_2 as well as the frequency f. The goal is to estimate the unknown parameters from the data $\boldsymbol{y}$. It is not difficult to see that Eq. (63) can be rewritten in vector–matrix form as

$$\boldsymbol{y} = \boldsymbol{H}\,(f)\beta + \boldsymbol{\varepsilon} \tag{66}$$

where $\boldsymbol{y}$ and ε are $\boldsymbol{N} \times 1$ vectors, $\boldsymbol{H}\,(f)$ is an $N \times 2$ matrix whose n-th row $\boldsymbol{h}^{\mathrm{T}}{}_n = [\sin(2\pi fn)\ \cos(2\pi fn)]$, and $\beta^{\mathrm{T}} = [B_1\ B_2]$. The unknown parameters $\theta^{\mathrm{T}} = [B_1 \quad B_2 \quad f]$ are thus split to β and $\gamma = [f]$. For given f, the parameters β can easily be estimated from

$$\hat{\beta} = \left(\boldsymbol{H}^{\mathrm{T}}(f)\boldsymbol{H}\,(f)\right)^{-1}\boldsymbol{H}^{\mathrm{T}}(f)\boldsymbol{y}. \tag{67}$$

If this result is plugged into the minimization of J, the estimate of f is obtained from

$$\hat{f} = \arg\min_{f} (\boldsymbol{y} - \boldsymbol{H}(f)\hat{\beta})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{H}(f)\hat{\beta}) \tag{68}$$

which, after simple algebra, can be rewritten as

$$\hat{f} = \arg\max_{f} \boldsymbol{y}^{\mathrm{T}} \boldsymbol{H}(f) \left(\boldsymbol{H}^{\mathrm{T}}(f)\boldsymbol{H}(f)\right)^{-1} \boldsymbol{H}^{\mathrm{T}}(f)\boldsymbol{y} \tag{69}$$

Thus, the estimation of the unknowns is implemented as follows: first the least-squares estimate of $\hat{f}$ is obtained from Eq. (69), and then the estimate of β according to

$$\hat{\beta} = \left(\boldsymbol{H}^{\mathrm{T}}(\hat{f})\boldsymbol{H}(\hat{f})\right)^{-1} \boldsymbol{H}^{\mathrm{T}}(\hat{f})\boldsymbol{y} \tag{70}$$

A rigorous treatment of problems where separability is possible can be found in Golub and Pereira (5).

Numerical methods. When the above methods fail, one usually resorts to numerical iterative techniques. There are two general iterative approaches; one is known as the Gauss–Newton, and the other, as the Newton–Raphson method. Suppose that the process of estimating θ is iterative and that at iteration k, $\theta^{(k)}$ is the estimate of θ. If $\theta^{(k)}$ is close enough to θ, one can expand $\boldsymbol{g}(\theta)$ around $\theta^{(k)}$ using linear Taylor expansion. The result is

$$\boldsymbol{g}(\boldsymbol{\theta}) \approx \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{G}^{(k)}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\right) \tag{71}$$

where $\boldsymbol{G}^{(k)}$ is an $N \times p$ matrix whose elements are

$$\left[\boldsymbol{G}^{(k)}\right]_{i,j} = \frac{\partial g_i}{\partial \theta_j} \tag{72}$$

and where θ is substituted by $\theta^{(k)}$ to evaluate the partials, and g_i and θ_j denote the ith and jth elements of $\boldsymbol{g}$ and θ, respectively. Recall now that the least-squares estimate is the value of θ that minimizes $J = \varepsilon^{\mathrm{T}}\varepsilon$. If in

$$\varepsilon = \boldsymbol{y} - \boldsymbol{g}(\boldsymbol{\theta}) \tag{73}$$

we substitute the approximated $\boldsymbol{g}(\theta)$ in Eq. (71), and with it we form the approximation of the criterion J, its minimization with respect to θ becomes easy, because then θ appears as a set of linear parameters. This estimate of θ is denoted as $\theta^{(k+1)}$ and is given by

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \left(\boldsymbol{G}^{(k)\mathrm{T}}\boldsymbol{G}^{(k)}\right)^{-1} \boldsymbol{G}^{(k)\mathrm{T}}\left(\boldsymbol{y} - \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right)\right) \tag{74}$$

Once $\theta^{(k+1)}$ is evaluated, it is used to compute $\boldsymbol{G}^{(k+1)}$, and the equation (74) is applied to determine $\theta^{(k+1)}$. The iterations continue until a predefined criterion of convergence is satisfied. This algorithm is known as the Gauss–Newton algorithm, and as $k \to \infty$, under fairly general assumptions on $\boldsymbol{g}$, it converges to the least-squares estimate $\hat{\theta}$ [Seber and Wild (4)]. In summary, the Gauss–Newton method is obtained by using a

first-order Taylor expansion of ε in Eq. (73) around the most recent value $\theta^{(k)}$, substituting the approximated ε in $J(\theta) = \varepsilon^{\mathrm{T}}\varepsilon$, and estimating $\theta^{(k+1)}$ as the value which minimizes the approximated $J(\theta)$.

The Newton–Raphson method is also obtained by using an approximation of $J\ (\theta)$. This time $J\ (\theta)$ is expanded directly using a quadratic Taylor expansion around the most recent estimate $\theta^{(k)}$. If the gradient vector of $J\ (\theta)$ is denoted by

$$\boldsymbol{q}(\boldsymbol{\theta}) = \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{75}$$

and the Hessian matrix of $J(\theta)$ is written as

$$\boldsymbol{H}\,(\boldsymbol{\theta}) = \frac{\partial^2 J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}\ \partial \boldsymbol{\theta}^{\mathrm{T}}} \tag{76}$$

the quadrature approximation of $J(\theta)$ around $\theta^{(k)}$ can be expressed by

$$J(\boldsymbol{\theta}) \approx J\big(\boldsymbol{\theta}^{(k)}\big) + \boldsymbol{q}^{\mathrm{T}}\big(\boldsymbol{\theta}^{(k)}\big)\big(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\big) + \frac{1}{2}\big(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\big)^{\mathrm{T}}\boldsymbol{H}\,\big(\boldsymbol{\theta}^{(k)}\big)\big(\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\big) \tag{77}$$

The minimization of the approximated $J\ (\theta)$ is again a linear problem and thus is easily obtained. If the solution is $\theta^{(k+1)}$, we can write

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathrm{H}^{-1}\big(\boldsymbol{\theta}^{(k)}\big)\boldsymbol{q}\big(\boldsymbol{\theta}^{(k)}\big) \tag{78}$$

In summary, the Newton–Raphson method starts with an initial guess $\theta^{(0)}$ and proceeds by applying Eq. (78), where Eqs. (75) and (76) define the gradient $\boldsymbol{q}(\theta)$ and the Hessian $\boldsymbol{H}(\theta)$.

The Gauss–Newton and Newton–Raphson method must be applied with great care because they are iterative approaches and convergence to the least-squares estimate is a critical issue. Many adaptations and protective strategies have been developed to improve their reliability. For details, consult for example Seber and Wild (4). Convergence is assessed usually by criteria of the following forms:

$$0 \leq \frac{J\big(\boldsymbol{\theta}^{(k)}\big) - J\big(\boldsymbol{\theta}^{(k+1)}\big)}{J\big(\boldsymbol{\theta}^{(k)}\big)} < \tau \tag{79}$$

or

$$\max_{1 \leq j \leq p} \frac{\left|\theta_j^{(k)} - \theta_j^{(k+1)}\right|}{\left|\theta_j^{(k)}\right|} \leq \eta \tag{80}$$

where τ and η are some small predefined positive numbers. It should be kept in mind that these criteria do not guarantee convergence; they should, rather, be considered *termination* criteria.

Sequential Least Squares

In many applications the observed data are received sequentially in time. Quite often in such cases it is preferable to find and update the least-squares estimates of the unknowns as the data keep arriving. Procedures developed for processing the data in this fashion are known as sequential (or recursive) least-squares methods, as opposed to batch methods that use all the data at once. The concept is best described when the unknowns, which have to be estimated, are linear parameters.

Let the $N \times 1$ vector $\boldsymbol{y}$ be as in Eqs. (26) and (27), i.e.,

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{81}$$

where $\boldsymbol{H}$ is an $N \times p$ matrix with known elements, and θ is a $p \times 1$ vector of unknown parameters. Suppose that the last sample that has been received is $y[n]$, that is, $y[k]$ has been observed for $k = 1,2, \ldots, n$, where

$$y[k] = \boldsymbol{h}^{\mathrm{T}}[k]\boldsymbol{\theta} + \varepsilon[k] \tag{82}$$

Here, the $p \times 1$ vector $\boldsymbol{h}^{\mathrm{T}}[k]$ is the kth row of the matrix $\boldsymbol{H}$ in Eq. (81). If we denote the observed vector up to sample n by $\boldsymbol{y}[n]$, then

$$\boldsymbol{y}[n] = \boldsymbol{H}[n]\boldsymbol{\theta} + \boldsymbol{\varepsilon}[n] \tag{83}$$

where $\boldsymbol{y}^{\mathrm{T}}[n] = [y[1]\ y[2]\ \cdots\ y[n]]$, $\boldsymbol{\varepsilon}^{\mathrm{T}}[n] = [\varepsilon[1]\ \varepsilon[2]\ \cdots\ \varepsilon[n]]$, and $\boldsymbol{H}[n]$ is an $n \times p$ matix identical to the first n rows of $\boldsymbol{H}$ (it is assumed that $n \geq p$, and that $\boldsymbol{H}[n]$ is a full-rank matrix). Then the least-squares estimate of θ based on the first n samples, denoted by $\hat{\theta}[n]$, is

$$\hat{\boldsymbol{\theta}}[n] = (\boldsymbol{H}^{\mathrm{T}}[n]\boldsymbol{H}[n])^{-1}\boldsymbol{H}^{\mathrm{T}}[n]\boldsymbol{y}[n] \tag{84}$$

Now, a new sample is received, $y[n + 1]$, and the objective is to modify $\hat{\theta}[n]$ so that the new information in $y[n + 1]$ is included in the estimate of θ. Of course, a straightforward way of accomplishing it would be to estimate $\hat{\theta}[n + 1]$ by an analogous expression to Eq. (84). However, there is a better way of getting $\hat{\theta}[n + 1]$, and it saves a great deal of computation. We rewrite Eq. (83) with the new sample $y[n + 1]$ as follows:

$$\begin{bmatrix} \boldsymbol{y}[n] \\ y[n+1] \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}[n] \\ \boldsymbol{h}^{\mathrm{T}}[n+1] \end{bmatrix} \boldsymbol{\theta} + \boldsymbol{\varepsilon}[n+1] \tag{85}$$

The usual minimization yields

$$\hat{\boldsymbol{\theta}}[n+1] = (\boldsymbol{H}^{\mathrm{T}}[n+1]\,\boldsymbol{H}[n+1])^{-1}\boldsymbol{H}^{\mathrm{T}}[n+1]\,\boldsymbol{y}[n+1] \tag{86}$$

where

$$(\boldsymbol{H}^{\mathrm{T}}[n+1]\,\boldsymbol{H}[n+1])^{-1} = (\boldsymbol{H}^{\mathrm{T}}[n]\,\boldsymbol{H}[n] + \boldsymbol{h}[n+1]\boldsymbol{h}^{\mathrm{T}}[n+1])^{-1} \tag{87}$$

The expression on the right-hand side of Eq. (87) can be rewritten by using the matrix inversion lemma

$$(\boldsymbol{A}+\boldsymbol{BCD})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{C}^{-1}+\boldsymbol{DA}^{-1}\boldsymbol{B})^{-1}\boldsymbol{DA}^{-1} \tag{88}$$

and this allows us, after a few lines of derivation, to write

$$\hat{\boldsymbol{\theta}}[n+1]=\hat{\boldsymbol{\theta}}[n]+\boldsymbol{\kappa}[n+1](y[n+1]-\boldsymbol{h}^{\mathrm{T}}[n+1]\hat{\boldsymbol{\theta}}[n]) \tag{89}$$

where the $p \times 1$ vector $\kappa[n+1]$ is given by

$$\boldsymbol{\kappa}[n+1]=\boldsymbol{P}[n]\boldsymbol{h}[n+1](1+\boldsymbol{h}^{\mathrm{T}}[n+1]\boldsymbol{P}[n]\boldsymbol{h}[n+1])^{-1} \tag{90}$$

with $\boldsymbol{P}[n]$ being defined according to

$$\boldsymbol{P}[n]=(\boldsymbol{H}^{\mathrm{T}}[n]\boldsymbol{H}[n])^{-1} \tag{91}$$

From Eq. (89), it is important to note that the estimate of $\hat{\psi}[n+1]$ is given as a function of the previous estimate $\hat{\theta}[n]$. The term $\boldsymbol{h}^{\mathrm{T}}[n+1]\ \hat{\theta}[n]$ can be interpreted as the predicted value of $y[n+1]$ based on the past samples and the adopted linear model, and

$$e[n+1]=y[n+1]-\boldsymbol{h}^{\mathrm{T}}[n+1]\hat{\boldsymbol{\theta}}[n] \tag{92}$$

as the prediction error of the model. The vector $\kappa[n+1]$ is known as a gain vector, and thus, Eq. (89) has the predictor–corrector form. The updated estimate $\hat{\theta}[n+1]$ in Eq. (89) can also be written as

$$\hat{\boldsymbol{\theta}}[n+1]=(\boldsymbol{I}-\boldsymbol{\kappa}[n+1]\boldsymbol{h}^{\mathrm{T}}[n+1])\hat{\boldsymbol{\theta}}[n]+\boldsymbol{\kappa}[n+1]y[n+1] \tag{93}$$

which suggests a different interpretation of the updated estimate—it is a weighted sum of the previous estimate and the information provided by $y[n+1]$, where the gain $\kappa[n+1]$ is determined to allocate the weights optimally.

It seems that the computation of $\kappa[n+1]$ is rather demanding because of the need to compute $\boldsymbol{P}[n]$, which is obtained by inverting the $p \times p$ matrix $\boldsymbol{H}^{\mathrm{T}}[n]\boldsymbol{H}[n]$. In fact, this is not needed, because it can be shown that $\boldsymbol{P}[n]$ may be obtained from $\boldsymbol{P}[n-1]$ by

$$\boldsymbol{P}[n]=\boldsymbol{P}[n-1]-\boldsymbol{\kappa}[n]\boldsymbol{h}^{\mathrm{T}}[n]\boldsymbol{P}[n-1] \tag{94}$$

where, evidently, there is no inversion involved in the computation. In summary, given the most recent estimate $\hat{\theta}[n]$, gain $\kappa[n]$, and matrix $\boldsymbol{P}[n]$, upon receiving a new sample, $y[n+1]$, the sequential least-squares method updates them by applying Eqs. (90), (89), and (94).

It is important to stress that all sequential algorithms require initializations of θ, κ, and $\boldsymbol{P}$. This can be done in two ways: one is to use *a priori* knowledge and quantify it by assigning initial values to them; the other is to obtain the initial values by applying a batch method to a small portion of the data.

There is abundant literature on sequential least squares in many journals and books. What is described here is only the standard recursive least-squares algorithm. It has limitations in numerical robustness and (its recursive nature notwithstanding) excessive computational complexity. Some alternatives to the standard

recursive least-squares algorithm include the square-root recursive least-squares method, which is based on QR decomposition of matrices, and the fast recursive least-squares method, which is based on special implementations of linear least-squares prediction in both the forward and backward directions. For more information, refer to Haykin (6) and the references therein.

Predictive Least Squares

In many situations, observed data may have more than one possible model for their description. A typical example is when we want to fit the data with a polynomial and we do not know its degree. As was mentioned in the section on polynomial fitting and spline interpolation, a too high degree can easily overfit the data, whereas a too low degree may also be unsatisfactory. In most practical situations, the degree is unknown and some statistics must be used to determine it. In the case when we want to use the least-squares approach for estimation of unknown parameters, which precludes probabilistic assumptions about the observed data, there are not many reliable criteria available for selecting the right model for the data. An additional difficulty is that for different problems the roles of the models may be quite distinct, and the differences are then strongly reflected in the criteria for choosing the best model. The implication is clear—there can be no universal criterion for model selection.

One approach for selecting a model, which has been shown to work very well and is generally applicable, is known as the predictive least-squares method [Rissanen (7)]. It is very simple to use, and is based on the principle that good models predict the future from the past better than poor models. It is well known that a polynomial of high degree fits data better than a polynomial of low degree; in fact, if the degree is high enough, the fitting can be perfect. It should be noted that in usual situations, the criterion for measuring the goodness of fit is given by the total sum of squared residuals. It is very important to observe that the same data are used for estimation of the unknown polynomial coefficients and the computation of the residual. In many cases, this is not good philosophy: it goes against the principle of parsimony in science and engineering, which states that use of unnecessary parameters in modeling should be avoided. Here, we present the predictive least-squares method in the context of choosing the best degree of a polynomial.

Suppose that the observed data $\boldsymbol{y}$ can be modeled by a polynomial whose degree m comes from the set $\{0, 1, 2, \ldots, p\}$. The idea is to compare all the polynomials by using the same yardstick, which is the accumulated prediction error of each polynomial. The estimation of the polynomial coefficients is carried out in the usual way by applying one of the sequential least-squares algorithms. However, the validation of the polynomials is implemented by data that have not been used for parameter estimation. For example, if the next sample is $y[n + 1]$, once it is received, it is compared with the predicted value of $y[n + 1]$ given by

$$\hat{y}[n+1] = \boldsymbol{h}^{\mathrm{T}}[n+1]\hat{\boldsymbol{\theta}}[n] \tag{95}$$

and the prediction error $e[n + 1]$ computed as in Eq. (92). The squared value of the error is added to the accumulated sum of the previous prediction errors. Next, the parameter estimates are updated according to Eq. (86), and $\hat{\theta}[n + 1]$ is used for prediction of $y[n + 2]$. The degree of the polynomial is then selected as the one that minimizes the accumulated prediction error, that is,

$$\hat{m} = \arg\min_{m} \sum_{n} (y[n] - \hat{y}_m[n])^2 \tag{96}$$

where $\hat{y}[n]$ is the sample of $y[n]$ predicted by the polynomial with degree m.

In summary, the best polynomial is the one that minimizes the accumulated prediction errors as given by Eq. (96). The coefficients of every polynomial are estimated sequentially as presented in the previous section. As the unknown parameters of each polynomial are updated, the corresponding squared prediction errors are accumulated. Once all the data are processed and all the polynomials are examined, the polynomial whose total sum of squared prediction errors is minimal is the winner.

The Bootstrap Method

Least-squares estimation is used in problems where probabilistic assumptions about the data are not made. It seems then that it would be difficult to make claims about the accuracy of the obtained estimates, unless ubiquitous normal assumptions are made or large-sample theory invoked. Indeed, take the simple case of linear least-squares estimation where unknown parameters θ are estimated according to Eq. (32). Can we say anything about the accuracy of the estimates θ without making suppositions about the error vectors ε? The answer is yes, and a powerful statistical procedure for providing such assessments is known as the bootstrap method. Although the bootstrap method can be applied to various tasks, including confidence-interval estimation and hypothesis testing, the underlying principle in all the applications is the same.

The bootstrap method imitates a situation that a practitioner would like to have in order to assess the quality of estimates. In the case of the linear least-squares estimation, it would be nice to have many sets of data $\boldsymbol{y}_i$, $i = 1, 2, \ldots, M$, for which we could write

$$\boldsymbol{y}_i = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\varepsilon}_i \tag{97}$$

where the ε_i's have the same statistical distribution. In that case, one would normally estimate the unknown parameters from each data set to obtain $\hat{\theta}_i$, from which statistics can be constructed that provide information about the accuracy of the estimates. Clearly, in most practical situations, multiple data sets are simply not available. In the late seventies, Efron proposed the bootstrap method to generate such data sets by repeatedly drawing random samples from the original data sets. To illustrate the procedure we proceed by way of example.

Suppose that the data set $\boldsymbol{y}$ is modeled as in Eq. (32) and the least-squares estimate of $\hat{\theta}$ is obtained in the usual way. Then we compute the residual data $\hat{\varepsilon}$ by

$$\hat{\boldsymbol{\varepsilon}} = \boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{\theta}} \tag{98}$$

The bootstrap is now applied by sampling randomly (with substitution) from the samples $\hat{\varepsilon}[n]$, $n = 1, 2, \ldots, N$, and constructing new data sets of the form

$$\boldsymbol{y}_i^* = \boldsymbol{H}\hat{\boldsymbol{\theta}} + \boldsymbol{\varepsilon}_i{}^* \tag{99}$$

where i stands for the ith data set, and

$$\boldsymbol{\varepsilon}_i^{*\mathrm{T}} = \begin{bmatrix} \varepsilon_i^*[1] & \varepsilon_i^*[2] & \cdots & \varepsilon_i^*[N] \end{bmatrix} \tag{100}$$

When sampling from $\hat{\varepsilon}$, some of the samples may not appear in $\varepsilon^*{}_i$, at all, and some may show up more than once. With the so constructed data sets, we proceed as if they were observed. Each $\boldsymbol{y}^*{}_i$ is processed to find

the least-squares estimate $\hat{\theta}_i$, and from all the estimates $\hat{\theta}_i$, $i = 1,2,\ldots,M$, various statistics for assessing the accuracy of $\hat{\theta}$ can easily be constructed.

This procedure can also be applied to any nonlinear least-squares method with practically no modifications. For further details about the bootstrap method, see Efron and Tibshirani (8).

In conclusion, the bootstrap is a simple method for statistical inference, especially in cases where a few statistical assumptions are made about the observed data, as in problems where least-squares estimation is employed. The drawback of the method is that it is computationally rather intensive.

BIBLIOGRAPHY

1. C. L. Lawson R. J. Hanson *Solving Least Squares Problems*, Englewood Cliffs, NJ: Prentice-Hall, 1974.
2. P. Dierckx *Curve and Surface Fitting with Splines*, Oxford, England: Oxford University Press, 1993.
3. S. M. Kay *Fundamentals of Statistical Signal Processing: Estimation Theory*, Englewood Cliffs, NJ: PTR Prentice-Hall, 1993.
4. G. A. F. Seber C. J. Wild *Nonlinear Regression*, New York: Wiley, 1989.
5. G. H. Golub V. Pereira The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, *SIAM J. Numer. Anal.*, **10**: 413–432, 1973.
6. S. Haykin *Adaptive Filter Theory*, Upper Saddle River, NJ: Prentice-Hall, 1996.
7. J. Rissanen Order estimation by accumulated prediction errors, *J. Appl. Probab.*, **23A**: 55–61, 1986.
8. B. Efron R. J. Tibshirani *An Introduction to the Bootstrap*, New York: Chapman & Hall, 1993.

READING LIST

G. Evans *Practical Numerical Analysis*, West Sussex, England: Wiley, 1995.
R. P. Feinerman D. J. Newman *Polynomial Approximation*, Baltimore: Williams & Wilkins, 1974.
W. Gautschi *Numerical Analysis: An Introduction*, Boston: Birkhäuser, 1997.
J. H. Mathews *Numerical Methods for Computer Science, Engineering and Mathematics*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

ZORAN I. MITROVSKI
Morgan Stanley Dean Witter
PETAR M. DJURIC
State University of New York