CHAPTER 4

Output-Error Method

4.1 INTRODUCTION

Having dealt in the preceding chapter with the formulation of nonlinear state-space model postulates to define the cause-effect relationship purported to underlie the physical phenomenon, and having elaborated on the simulation (that is, computation of system responses) using numerical procedures, we now turn our attention to the first of the two central methods of aircraft parameter estimation, namely, the output-error method. The other method called filter-error method will be discussed in the next chapter. Both of these methods belong to a general class of output error, also called response-curve-fitting, methods. The class of estimation methods called the equation-error methods will be considered separately. In this class of output-error methods, model parameters are adjusted iteratively to minimize the error between the measured variables (system output) and the estimated (model predicted) responses. The method, however, leads to a nonlinear optimization problem, in which the computational burden is relatively high. The method of weighted least squares, the simplest among this class, accounts for measurement noise. However, it assumes a priori specification of the weighting matrix. Based on probability theory, a more profound formulation called the maximum-likelihood principle was provided by Fisher [1–3]. It can handle both process and measurement noise and has several desirable statistical properties of a "good estimator." This chapter considers the case where we assume that the process noise is negligible and that the measurements are corrupted by additive measurement noise only. It leads to the popularly called "output-error method" (OEM). The output-error as well as filter-error methods represent a natural formulation for dynamic systems, be it a linear or nonlinear. In the majority of practical applications, the dynamic process being investigated evolves continuously in time whereas the observations are recorded at discrete time points. Accordingly, we consider here the mixed continuous-/discrete-time formulation.

Since its introduction in the 1960s, the output-error method is the most widely applied time-domain method to estimate aircraft parameters from flight data.

This is quite evident from the numerous applications reported in the literature; Refs. [4–9] provide a good survey of the flight vehicle applications. Initially, during the 1960s to 1970s, the scope of the output-error method was limited to linear systems. The limitations were not related to the applicability of the maximum likelihood principle to nonlinear systems; rather, they were practical difficulties related to handling general nonlinear model structures and to compute the sensitivity matrix required in the optimization of the cost function. In the 1980s, these difficulties were solved through a numerical approach, which has paved the way to analyze general nonlinear systems of arbitrary complexity [6].

In this chapter, we begin with the basic formulation of the maximum likelihood principle, the assumptions that are made, and study briefly its statistical properties. This is followed by a detailed study of the cost function and various optimization methods. For each aspect, we indicate the pros and cons and attempt to make recommendations based on our experience of modeling different types of flight vehicles [6, 7]. We study the performance of the output-error method on two sample examples, using estimation software that we also supply with the book to follow the discussion and to gain more insight into the algorithmic steps and numerical aspects. The data analyzed are also supplied as sample data to check the results and possible adaptation of the software to individual needs.

4.2 PRINCIPLE OF MAXIMUM LIKELIHOOD ESTIMATION

In the general theory of estimation founded by Fisher [1–3], it is assumed as a working hypothesis that the parent population has a known determinate mathematical form but for the numerical value of some parameter vector Θ (a column vector). Parent population is a set of all possible measurements as the number of samples N goes to ∞. It is required to deduce the unknown value of the parameter vector Θ using a set of observations $(z_1, z_2, \ldots, z_N)$.

Considering the given samples of N random observations $(z_1, z_2, \ldots, z_N)$ to be independent and taken from the same population, the likelihood function can be defined as

$$p(z|\Theta) = p(z_1|\Theta) \cdot p(z_2|\Theta) \cdot \cdots \cdot p(z_N|\Theta) = \prod_{k=1}^{N} p(z_k|\Theta) \tag{4.1}$$

where $p(z|\Theta)$ is the probability of z given Θ. The maximum likelihood (ML) method, introduced by Fisher as a general estimation procedure, amounts to selecting that value of Θ within the admissible range such that $p(z|\Theta)$ is maximized. The likelihood function represents probability density, but the probability density of observed variables and not of the parameters. The unknown parameters are assumed not to depend on chance [10, 11].

Because of the exponential nature of many density functions [12], the logarithm of the likelihood function, which has the same optimal solution, is generally preferred. The maximum likelihood estimate is thus obtained as

$$\widehat{\Theta}_{\mathrm{ML}} = \arg\left\{ \min_{\Theta} \ell n\, p(z|\Theta) \right\} \tag{4.2}$$

Assuming that $p(z|\Theta)$ is a twice differentiable function, practical application of the maximum likelihood method requires solving the likelihood equation:

$$\frac{\partial\, \ell n\, p(z|\Theta)}{\partial \Theta} = 0 \tag{4.3}$$

Equation (4.3), a column vector, is a set of nonlinear equations that can be solved only by successive approximation. Linear expansion of the likelihood equation about a first approximation Θ_0 of Θ yields

$$\frac{\partial\, \ell n\, p(z|\Theta_1)}{\partial \Theta} \approx \frac{\partial\, \ell n\, p(z|\Theta_0)}{\partial \Theta} + \frac{\partial^2\, \ell n\, p(z|\Theta_0)}{\partial \Theta^2}\, \Delta\Theta \tag{4.4}$$

where $\Theta_1 = \Theta_0 + \Delta\Theta$ is the improved approximation to Θ. Equating Eq. (4.4) to zero yields a linear system of equations:

$$\frac{\partial^2\, \ell n\, p(z|\Theta_0)}{\partial \Theta^2}\, \Delta\Theta = -\frac{\partial\, \ell n\, p(z|\Theta_0)}{\partial \Theta} \tag{4.5}$$

which can be solved by any procedure of linear algebra to yield the parameter improvement vector $\Delta\Theta$. The expected value of the matrix of second gradient $[\partial^2 \ell n\ p(z|\Theta)/\partial \Theta^2]$ is called the Fisher information matrix.

To apply the maximum likelihood function to the dynamical system that we have considered in Chapter 3, it is now required to write the expression for $p(z|\Theta)$, that is, the conditional probability density function. Although the method is applicable to any form of the density function distribution, for mathematical tractability we consider the Gaussian (normal) distribution, which is completely determined by the first and second moments, that is, the mean and the covariance matrix. It is the most widely used assumption in practical cases. For such a case, and further assuming the error $v(t_k) = z(t_k) - y(t_k)$ at different time points t_k to be statistically independent, that is,

$$E\{v(t_k)\, v^T(t_\ell)\} = R\, \delta_{k\ell} \tag{4.6}$$

where $\delta_{k\ell}$ is the Kronecker delta symbol ($\delta_{k\ell} = 1$ for $k = \ell$ and $= 0$ for $k \neq \ell$), it is shown in Appendix C that the likelihood function $p(z(t_1), \ldots, z(t_N)|\Theta, R)$ of n_y-dimensional measurement vector at N discrete time points for a given

parameter vector Θ and given measurement error covariance matrix R is [13, 14]

$$p(z(t_1),\ldots,z(t_N)|\Theta,R)=\prod_{k=1}^{N}p(z(t_k)|\Theta,R)$$

$$=\{(2\pi)^{n_y}|R|\}^{-N/2}\exp\left[-\frac{1}{2}\sum_{k=1}^{N}[z(t_k)-y(t_k)]^T\,R^{-1}\,[z(t_k)-y(t_k)]\right] \quad (4.7)$$

The reader may note that we have implicitly extended here the basic definition of Eq. (4.1) to include the conditioning on R, the covariance matrix of the residuals. For notational simplicity we denote

$$p(z|\Theta,R)=p(z(t_1),\ldots,z(t_N)|\Theta,R) \quad (4.8)$$

We pay specific attention here to the commonly misunderstood concept and interpretation of the exact meaning of the maximum likelihood function. The likelihood function $p(z|\Theta,R)$ does not represent the probability distribution of the unknown parameters Θ, but of the measurements z. Although looking at the observations some parameters may appear more plausible and some less, the unknown parameters are not random variables and do not have probability density. Maximum likelihood estimation means that the Θ vector is searched, which maximizes the function $p(z|\Theta,R)$. Such a vector is "the most plausible" because it gives the highest probability to the measurements.

From Eq. (4.7), we can immediately compute the maximum likelihood estimates of the unknown parameters Θ and of the measurement noise covariance matrix R by setting the first derivatives with respect to each of them to zero. However, an equivalent and more practical solution to the optimization problem is the minimization of $L(z|\Theta,R)$, the negative logarithm of the likelihood function:

$$L(z|\Theta,R)=\frac{1}{2}\sum_{k=1}^{N}[z(t_k)-y(t_k)]^T\,R^{-1}[z(t_k)-y(t_k)]$$

$$+\frac{N}{2}\ell n(\det(R))+\frac{Nn_y}{2}\ell n(2\pi) \quad (4.9)$$

We prefer to deal with Eq. (4.9) instead of (4.7) because, as already mentioned, the natural logarithm of the normal density function is a simpler function than the density function itself. The differentiation result $\partial(\ell n\ p)/\partial\Theta=1/p\cdot\partial p/\partial\Theta$ guarantees that the derivative of log of p is zero, when $\partial p/\partial\Theta$ is zero, and thus yields the same results.

4.3 PROPERTIES OF MAXIMUM LIKELIHOOD ESTIMATES

Solving the likelihood equation yields parameter estimates $\widehat{\Theta}_{\mathrm{ML}}$ that lend the largest probability to the observations $(z_1, z_2, \ldots, z_N)$, which are assumed to be

statistically independent. Although the maximum likelihood principle is intuitively appealing, we will study here briefly the statistical properties of the estimates. For a more detailed treatment of these statistical properties, the reader can refer to Appendix D or any one of the standard books on statistics [10–17].

The concepts of efficiency and consistency of the maximum likelihood estimates were introduced by Fisher [3]. Based on the assumption of independent observations, Wald established the asymptotic consistency and unbiasedness of the maximum likelihood estimates [15]. The asymptotic normality was proved by Cramer [16]. We recapture here only the essentials of a "good" estimator [18].

1. Unbiased estimates:
   The maximum likelihood estimates are asymptotically unbiased, that is,

$$\lim_{N\to\infty} E(\widehat{\Theta}_{\mathrm{ML}}) = \Theta$$

   where Θ represents the true values of the parameters.

2. Asymptotic consistency:
   Maximum likelihood estimates $\widehat{\Theta}_{\mathrm{ML}}$ are asymptotically consistent, that is, $\widehat{\Theta}_{\mathrm{ML}}$ converges in probability to the true value Θ.

3. Asymptotic normality:
   The maximum likelihood estimates $\widehat{\Theta}_{\mathrm{ML}}$ obtained from different sets of data samples are normally distributed around the true value Θ, that is,

$$\sqrt{N}(\widehat{\Theta}_{\mathrm{ML}} - \Theta) \to r_1 \quad \sim \mathcal{N}(0, \overline{\mathcal{F}}^{-1})$$

   where r_1 the random variables, $\overline{\mathcal{F}}$ the average Fisher information matrix per sample, and $\mathcal{N}(0, \overline{\mathcal{F}}^{-1})$ the normal (Gaussian) distribution with zero mean and variance $\overline{\mathcal{F}}^{-1}$.

4. Asymptotic efficiency:
   The maximum likelihood estimates $\widehat{\Theta}_{\mathrm{ML}}$ are asymptotically efficient in the sense that they attain the Cramér–Rao lower bounds, which is a measure of achievable statistical accuracy.

The property of consistency implies that the maximum likelihood estimates converge in probability to the true values. The property of asymptotic normality implies that the estimates obtained from different sets of data samples corresponding to different experiments are clustered around the true value with a normal distribution. The property of asymptotic efficiency is of practical significance. It implies that the maximum likelihood estimator makes efficient use of the available data. The Cramér–Rao lower bound indicates the theoretically maximum achievable accuracy of the estimates; see Appendix D.

4.4 MAXIMUM LIKELIHOOD FUNCTION FOR ESTIMATION OF PARAMETERS IN DYNAMICAL SYSTEMS

The mathematical model of the dynamical system, whose parameters are to be estimated, is assumed to be described by the following general nonlinear system representation:

$$\dot{x}(t) = f[x(t), u(t), \beta], \quad x(t_0) = x_0 \tag{4.10}$$

$$y(t) = g[x(t), u(t), \beta] \tag{4.11}$$

$$z(t_k) = y(t_k) + G\,\nu(t_k) \tag{4.12}$$

This is the same set of equations that we had postulated in Chapter 3 to represent a dynamical system in state space. All of the variables have already been defined there. To recollect, the nonlinear functional relationships f and g are assumed to be real valued. They may be nonlinear in the state variables x and/or in the control variables u, as well as in the parameters to be estimated [18–21]. This class of nonlinear system description is in general quite adequate for various estimation problems encountered in flight vehicle modeling from flight data [4–6].

To apply the likelihood function to estimate the parameters of a dynamical system represented in Eqs. (4.10–4.12), we make the following assumptions:

1. The input sequence $(u(t_k), k = 1, 2, \ldots, N)$ is exogenous, that is, generated independent of the system output.
2. The measurement errors $\nu(t_k) = z(t_k) - y(t_k)$ at different discrete time points are statistically independent. They are assumed to be distributed with zero mean and covariance matrix R, that is,

$$E\{\nu(t_k)\} = 0$$

$$E\{\nu(t_k)\nu^T(t_\ell)\} = R\delta_{k\ell}$$

3. The system is corrupted by measurement noise only.
4. Control inputs $u(t_k)$ are sufficiently and adequately (that is, in magnitude and frequencies) varied to excite directly or indirectly the various modes of the dynamical system being analyzed.

By definition, exogenous inputs are those that affect the system, but are not affected by the system itself. It is the key to success of parameter estimation and also helps to ensure that the residuals are independent at different time points. For stable aircraft, it is usually possible to apply control inputs about all axes independent of the aircraft response. However, in the case of unstable aircraft, due to feedback, this assumption might not be met, and hence special treatment is necessary. We will discuss problems and solutions of unstable aircraft in a separate chapter later.

The second assumption allows applying directly the likelihood function that we already discussed. The third assumption implies that we treat in this chapter the dynamic system as deterministic. Note that we have dropped the process noise w and the process noise distribution matrix F in Eq. (4.10). For such cases, the state variables are obtained through simple integration of the state equations. The case of process noise will be dealt with in the next chapter. Through the fourth assumption, we ensure identifiability of the parameters because the basic rule of *"If it is not in the data, it can not be identified"* applies to all exercises that attempt parameter estimation from experimental data. This is true irrespective of the type of flight vehicle we might attempt to model.

In the preceding discussion of Secs. 4.2 and 4.3, the likelihood function $p(z|\Theta, R)$ included the arguments z, Θ, and R. Strictly speaking, mathematically it would be more appropriate to write the likelihood function for the given data as $p(z|\Theta, R, u)$, where we have now included the argument u for the system inputs. However, the argument u is dropped in the further discussions without loss of generality, because 1) identical control inputs are used to excite the system as well as the postulated model (see Fig. 4.1) and 2) the control inputs are assumed to be exogenous, noise free, and known a priori.

From Sec. 4.2, we know that maximum likelihood estimates are obtained by minimization of Eq. (4.9), the negative logarithm $L(z|\Theta, R)$ of the likelihood function. For convenience, we rename $L(z|\Theta, R)$ as J, the cost function to be minimized

$$J(\Theta, R) = L(z|\Theta, R) = \frac{1}{2}\sum_{k=1}^{N}[z(t_k) - y(t_k)]^T R^{-1}[z(t_k) - y(t_k)] + \frac{N}{2}\ell n(\det(R)) + \frac{Nn_y}{2}\ell n(2\pi) \tag{4.13}$$

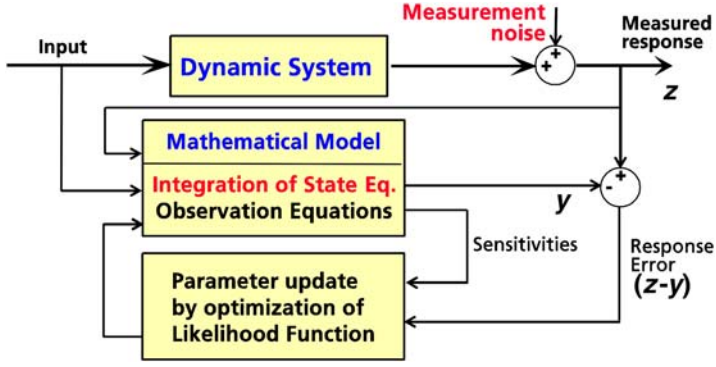

Fig. 4.1 Block schematic of output-error method.

subject to the system model postulated in Eqs. (4.10–4.12). Figure 4.1 shows a block schematic of the output-error method. The details of each of the blocks in this figure will be discussed hereafter. Note that the system responses y are functions of the parameters Θ, and hence the exact notation of the preceding equation should include $y(\Theta)$. However, for the sake of brevity, this dependence is not shown explicitly in any of the equations that we deal with.

4.5 BASICS OF COST FUNCTION OPTIMIZATION

Before we turn to actual methods of optimization, we further analyze the likelihood cost function $J(\Theta, R)$ to gain more insight [22, 23]. For a postulated model and data set being analyzed, the number of observation variables n_y and the number of data points N are fixed. Thus, the last term in Eq. (4.13) is a constant and hence neglected in the optimization without affecting the results. For the first two terms of Eq. (4.13), there are two cases that are of interest depending upon whether the measurement error covariance matrix R is known or not.

4.5.1 KNOWN MEASUREMENT NOISE COVARIANCE MATRIX

In the first case, we assume that the measurement noise covariance matrix R is known; at this stage it is immaterial as to how we might arrive at the knowledge of R. In such a case, because R is assumed to be known, the second term in Eq. (4.13), namely, $N\,\ell n(\det(R))/2$, is constant. The cost function then reduces to

$$J(\Theta) = \frac{1}{2}\sum_{k=1}^{N}[z(t_k) - y(t_k)]^T R^{-1}[z(t_k) - y(t_k)] \tag{4.14}$$

To understand the nature of Eq. (4.14), let us consider a simpler case of having just two observation variables and assume R to be a diagonal matrix. In such a case, multiplying out the terms in the summation on the right-hand side leads to $J(\Theta) = [r_{11}e_1^2(t_k) + r_{22}\,e_2^2(t_k)]$, where e_1 and e_2 denote the response errors in the two observation variables and r_{11} and r_{22} the diagonal elements of R representing the variances. Thus, the cost function $J(\Theta)$ in Eq. (4.14) is nothing but the weighted sum of squares of the response error and is quadratic in nature. We can apply any one of the optimization methods we will discuss in the following sections.

4.5.2 UNKNOWN MEASUREMENT NOISE COVARIANCE MATRIX

In the other case of unknown R, optimization of Eq. (4.13) is little more involved. The brute-force method would be to include the elements of the covariance matrix to the unknown parameters vector Θ and then apply one of the optimization methods. This approach is, however, never practiced because there is no closed-form solution to this minimization problem. During the optimization, the estimates of the system parameters depend on the measurement noise covariance

matrix and vice versa. Any attempt in this direction is bound to fail. To overcome this difficulty, we use the relaxation strategy in which optimization of the likelihood function, Eq. (4.13), is carried out in two steps.

In the first step, it can be shown that for any given value of the parameter vector Θ, the maximum likelihood estimate of R is given by

$$R = \frac{1}{N} \sum_{k=1}^{N} [z(t_k) - y(t_k)] \, [z(t_k) - y(t_k)]^T \tag{4.15}$$

Equation (4.15) is obtained by partially differentiating Eq. (4.13) with respect to R and setting it to zero. A detailed derivation of the preceding expression for R is provided in Appendix E.

Having obtained a maximum likelihood estimate of R, substitution of Eq. (4.15) in (4.13) yields

$$J(\Theta) = \frac{1}{2} n_y N + \frac{N}{2} \ell n(\det(R)) + \frac{N n_y}{2} \ell n(2\pi) \tag{4.16}$$

Once again, recall that n_y and N are fixed for a postulated model and a data set being analyzed. Hence, in addition to the last term, the first term of Eq. (4.16) also turns out to be a constant and can be neglected without affecting the minimization results. Thus, the cost function reduces to

$$J(\Theta) = \det(R) \tag{4.17}$$

Determination of the parameter vector Θ, which minimizes $\det(R)$, or equivalently the function $L(z|\Theta)$, is an optimization problem that can be solved by applying different methods.

Thus, the relaxation procedure can be summarized as follows:

1. Choose suitable initial values for Θ.
2. Compute system outputs y and the residuals $(z - y)$; estimate the measurement noise covariance matrix R.
3. Minimize $J(\Theta)$ with respect to Θ applying one of the nonlinear optimization methods.
4. Iterate on step 2, and check for convergence.

Mathematical proof of a global convergence of the relaxation procedures is difficult, but it is convenient to use and mostly works well in practice. From the abundant examples pertaining to flight vehicles of diversified complexity reported in the literature, it is apparent that this is the only pragmatic approach to nonlinear optimization.

In general, the covariance matrix of the output errors is unknown and, hence, has to be estimated. In such a case, as just shown, the cost function to be minimized reduces to the determinant of the covariance matrix of the output errors. Because the covariance matrix is usually assumed to be diagonal, a simplification

that has found widespread use in practice, the cost function then represents nothing but the product of the variances. Any optimization method, direct search or gradient based, can be applied to obtain the estimates of Θ. In any case, it is necessary to compute the system states x and responses y to obtain the cost function. Furthermore, depending upon the optimization algorithm, response gradients might be required.

4.6 GAUSS–NEWTON ALGORITHM

The Gauss–Newton method of nonlinear optimization belongs to a class of second-order algorithms and is one of the most widely applied method in flight vehicle system identification in the time domain. It is a modification of the Newton, also called Newton–Raphson, method. Therefore, we first study this basis method before discussing the modifications and its implications. A necessary condition for minimization of the likelihood function with respect to the unknown parameters is given by

$$\frac{\partial J(\Theta)}{\partial \Theta} = 0 \tag{4.18}$$

The Taylor-series expansion of $\partial J/\partial\Theta$ about the ith value of the parameter vector Θ, truncated after two terms, is given by

$$\left(\frac{\partial J}{\partial \Theta}\right)_{i+1} \approx \left(\frac{\partial J}{\partial \Theta}\right)_i + \left(\frac{\partial^2 J}{\partial \Theta^2}\right)_i \Delta\Theta \tag{4.19}$$

where $\Delta\Theta = \Theta_{i+1} - \Theta_i$ is the parameter change and $(\partial^2 J/\partial\Theta^2)_i$ is the second gradient of the cost function with respect to Θ at the ith iteration. In optimization theory, the matrix of second gradients is termed as Hessian. Now, using the necessary condition of Eq. (4.18) for the minimum, we set the right-hand side of Eq. (4.19) to zero and solve for $\Delta\Theta$:

$$\Delta\Theta = -\left[\left(\frac{\partial^2 J}{\partial \Theta^2}\right)_i\right]^{-1}\left(\frac{\partial J}{\partial \Theta}\right)_i \tag{4.20}$$

The preceding change in Θ on the $i+1$th iteration makes local $(\partial J/\partial\Theta)_{i+1}$ approximately zero. Thus, starting from an initial guess value Θ_0, $\Theta_{i+1} = \Theta_i + \Delta\Theta$ provides an iterative solution to find the minimum of a function. Such an algorithm is commonly known as the Newton–Raphson method [24, 25].

To demonstrate the preceding optimization procedure, we consider a simple one-dimensional case. First, we apply the preceding procedure to find θ that minimizes a quadratic cost function $J(\theta) = 8\theta^2$. Let the starting value of θ be 4. The first and second gradients, $\partial J/\partial\theta$ and $\partial^2 J/\partial\theta^2$, of this cost function evaluated at $\theta_0 = 4$ are given by $16\theta_0 = 16 \times 4$ and 16 respectively. Equation (4.20) then yields an incremental value of $\Delta\theta = -(16)^{-1} \times 16 \times 4 = -4$. Thus, after the

first iteration the iterative update leads to $\theta_1 = \theta_0 + \Delta\theta = 4 - 4 = 0$. For the quadratic cost function, this solution happens also to be the minimum and does not require any further iteration. This is always the case for any arbitrary quadratic function because the second gradient $\partial^2 J/\partial\theta^2$ is constant, which allows, as illustrated schematically in Fig. 4.2a, determination of the minimum in a single step irrespective of the starting value θ_0. This simple test case can be run using the program "/FVSysID2/chapter04/ExQuadFun.m."

In the second example, we consider optimization of a nonquadratic function $J(\theta) = 4\theta^2 + 0.4\theta^3$. The first and the second gradients are given by $\partial J/\partial\theta = 8\theta + 1.2\theta^2$ and $\partial^2 J/\partial\theta^2 = 8 + 2.4\theta$, respectively. The second gradient in this case is not a constant, but is a function of θ. The optimization procedure yields step by step an improved solution given by Eq. (4.20) based on the local quadratic approximation of the nonquadratic cost function. For example, starting from $\theta_0 = 4$, the first iteration yields the increment $\Delta\theta_1 = -(8 + 2.4 \times 4)^{-1}(8 \times 4 + 1.2 \times 4^2) = -2.9091$, and the updated parameter $\theta_1 = \theta_0 + \Delta\theta_1 = 4 - 2.9091 = 1.0909$. Now, starting from the updated value θ_1, the second iteration leads to $\Delta\theta_2 = -(8 + 2.4 \times 1.0909)^{-1}(8 \times 1.0909 + 1.2 \times 1.0909^2) = -0.9564$, yielding the updated parameter value of $\theta_2 = \theta_1 + \Delta\theta_2 = 1.0909 - 0.9564 = 0.1345$. The iterative procedure continues until the minimum is achieved after the fourth iteration. As already pointed out, in each iteration the first and second gradients change and have to be computed for the local value of θ. The optimization procedure is schematically illustrated in Fig. 4.2b. This test case is run using the program "/FVSysID2/chapter04/ExNonQuadFun.m." Thus, as just illustrated, geometrically interpreted, the basic idea behind the Newton–Raphson method is to construct, at each iteration, a quadratic approximation to the cost function and then take a step in the direction of the

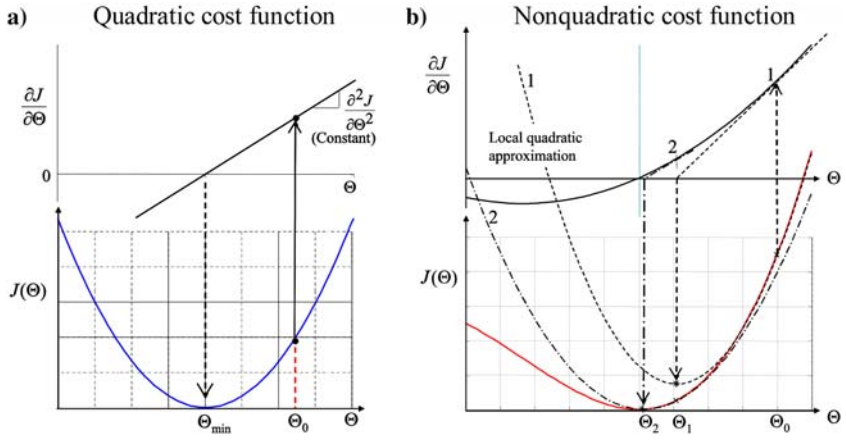

Fig. 4.2 Schematic of optimization procedure.

minimum of that approximated quadratic function. Because the step is directly towards the local minimum, the method is much more efficient than the gradient-based methods that merely step in the local downhill direction.

In general, away from the minimum of the cost function, the algorithm takes larger incremental steps $\Delta\theta$ corresponding to larger values of the gradient whereas they become smaller and smaller as the minimum is approached; the number of steps required finding the minimum in a specific case will depend upon the shape of the cost function. If the cost function has multiple minima depending upon the initial starting values, the search procedure might find a local extremum.

Now, in our case of parameter estimation from flight data, having determined the residual covariance matrix using Eq. (4.15), we now minimize the quadratic cost function $J(\Theta)$ given by Eq. (4.14) resulting from the likelihood function, Eq. (4.13). The partial differentiation of Eq. (4.14) results in

$$\frac{\partial J}{\partial \Theta} = -\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}[z(t_k) - y(t_k)] \tag{4.21}$$

and the partial differentiation of $\partial J/\partial\Theta$ in Eq. (4.21) with respect to Θ yields

$$\frac{\partial^2 J}{\partial \Theta^2} = \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \frac{\partial y(t_k)}{\partial \Theta} - \sum_{k=1}^{N} \left[\frac{\partial^2 y(t_k)}{\partial \Theta^2}\right]^T R^{-1}[z(t_k) - y(t_k)] \tag{4.22}$$

The computation of the first gradient $\partial J/\partial\Theta$ from Eq. (4.21) is relatively straightforward. It requires just the measured data z, the corresponding computed variables y, and the response gradient $\partial y/\partial\Theta$. Recall that the estimate of the weighting matrix R, given by Eq. (4.15), is already obtained for the current Θ.

Computation of the second gradient $\partial^2 J/\partial\Theta^2$ given by Eq. (4.22) is more complex and time consuming, as it requires the second gradient of the response $\partial^2 y/\partial\Theta^2$. However, the second term on the right-hand side of Eq. (4.22) includes a term $[z(t_k) - y(t_k)]$, and it is found that this contribution goes to zero as the process converges. In an ideal case, the residuals $[z(t_k) - y(t_k)]$ should just be the random measurement error at each time point. Based on our assumption of zero mean and independent noise, the second term tends to cancel out when summed over a sufficient length of data points. Based on this practical consideration, we neglect the second term in Eq. (4.22) and approximate the second gradient $\partial^2 J/\partial\Theta^2$ as

$$\frac{\partial^2 J}{\partial \Theta^2} \approx \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \left[\frac{\partial y(t_k)}{\partial \Theta}\right] \tag{4.23}$$

The main advantage of the preceding simplification is the saving in the computational burden without affecting much the rapid, quadratic convergence. It is also associated with another minor advantage, namely, the second gradient given by Eq. (4.23) always remains positive definite, because it is the sum of quadratic

terms. The preceding simplification was suggested by Balakrishnan [26] and leads to a more tractable algorithm called modified Newton–Raphson method, which is found to work well in most of the practical applications. In the literature this algorithm is also sometimes called the Newton–Balakrishnan algorithm or also the Gauss–Newton method, the terminology that we will follow in this book.

4.7 METHOD OF QUASI-LINEARIZATION

In this section we follow a slightly different approach to derive an expression to compute the parameter update. We already know from Eq. (4.18) that $\partial J/\partial\Theta$ must vanish at the minimum. Accordingly, setting Eq. (4.21) to zero yields

$$\frac{\partial J}{\partial \Theta} = -\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}[z(t_k) - y(t_k)] = 0 \tag{4.24}$$

Now, applying the quasi-linearization [27, 28], that is, a first-order approximation, to the system responses around some nominal value Θ_0, we obtain

$$y(\Theta) = y(\Theta_0 + \Delta\Theta) \approx y(\Theta_0) + \frac{\partial y}{\partial \Theta}\,\Delta\Theta \tag{4.25}$$

Substituting the linearized $y(\Theta)$ in Eq. (4.24) leads to

$$\frac{\partial J}{\partial \Theta} = -\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \left[z(t_k) - \left\{y(t_k) + \frac{\partial y(t_k)}{\partial \Theta}\,\Delta\Theta\right\}\right] = 0 \tag{4.26}$$

Note that for convenience, we have dropped the argument Θ_0 from y in the preceding equation. Equation (4.26) can be rewritten as

$$\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}\,[z(t_k) - y(t_k)] - \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \left[\frac{\partial y(t_k)}{\partial \Theta}\right] \Delta\Theta = 0 \tag{4.27}$$

which leads to a system of linear equations:

$$\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \left[\frac{\partial y(t_k)}{\partial \Theta}\right] \Delta\Theta = \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}[z(t_k) - y(t_k)] \tag{4.28}$$

Equation (4.28) derived applying quasi-linearization is exactly the same as Eqs. (4.21) and (4.23) derived for the Gauss–Newton method. Thus, the Gauss–Newton and quasi-linearization methods are equivalent.

The Gauss–Newton and quasi-linearization methods lead to a system of linear equations, which can be represented in a general form as

$$\Theta_{i+1} = \Theta_i + \Delta\Theta \quad \text{and} \quad \mathcal{F}\Delta\Theta = -\mathcal{G} \tag{4.29}$$

where i is the iteration index and $\mathcal{F}$ and $\mathcal{G}$ are given by

$$\mathcal{F} = \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]$$
$$\mathcal{G} = -\sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1} [z(t_k) - y(t_k)] \tag{4.30}$$

$\mathcal{F}$ is the information matrix (also called Hessian), $\mathcal{G}$ is the gradient vector, and $\Delta\Theta$ is the parameter change vector. Once the gradient vector $\mathcal{G}$ and the matrix of second gradients $\mathcal{F}$ are computed, the parameter updates are given by Eq. (4.29). Thus, we now have both of the steps of the relaxation algorithm, which can be carried out successively, namely, compute the measurement noise covariance matrix using Eq. (4.15) and then apply the Gauss–Newton method given by Eqs. (4.29) and (4.30) to update the parameters. Equation (4.29), $\mathcal{F}\Delta\Theta = -\mathcal{G}$, can be solved for $\Delta\Theta$ by any standard algorithm from linear algebra. Because of inaccurate numerical results, direct matrix inversion methods are rarely used. It is obvious from Eq. (4.30) that the information matrix $\mathcal{F}$ is symmetric, positive-definite. For such cases the Cholesky factorization is an efficient procedure. Singular value decomposition is yet another approach that is also efficient. Details of these techniques can be found in any standard text book on numerical methods.

4.8 SYSTEM RESPONSE AND SENSITIVITY COEFFICIENTS

The iterative update of the parameter vector Θ by Gauss–Newton or quasi-linearization method using Eqs. (4.29) and (4.30) requires the following:

1. Computation of the system responses $y(t_k)$
2. Computation of the response gradients $\partial y(t_k)/\partial\Theta$

Efficient implementation of these computational aspects, together with flexibility to handle conveniently different model structures, is important for parameter estimation in nonlinear systems [20, 21, 29].

The model predicted response $y(t)$, Eq. (4.11), is a function of the parameters being estimated and the state variables $x(t)$. According to our assumption of the system being corrupted by measurement noise only and that the measured control inputs contain no errors and noise, we are dealing in this chapter with deterministic systems. For such a case we can simply integrate the state equations, Eq. (4.10), using one of the numerical integration procedures elaborated in Sec. 3.8.

Computation of the response gradients $\partial y(t_k)/\partial\Theta$, also called the sensitivity matrix, involves some critical issues that directly affect the capability to handle

different nonlinear models. In the conventional approach, the sensitivity matrix $\partial y/\partial \Theta$ is obtained by solving the sensitivity equations, which are obtained analytically by partial differentiation of the system equations (4.10) and (4.11):

$$\frac{\partial \dot{x}}{\partial \Theta} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial \Theta} + \frac{\partial f}{\partial \Theta} \tag{4.31}$$

$$\frac{\partial y}{\partial \Theta} = \frac{\partial g}{\partial x}\frac{\partial x}{\partial \Theta} + \frac{\partial g}{\partial \Theta} \tag{4.32}$$

In the case of linear systems, the method of transition matrices for the solution of Eq. (4.31) and simple matrix multiplications provides an elegant procedure. However, if the system model is nonlinear, as in Eqs. (4.10) and (4.11), any changes in the model structure entail rederivation of the sensitivity equations of the form given in Eqs. (4.31) and (4.32). As a consequence, changes in the parameter estimation software are necessary. This additional programming effort becomes cumbersome especially when alternative model structures are to be tried out to fit the experimental data. Furthermore, in case of systems with discontinuous nonlinearities numerical problems might be encountered due to the difficulty of defining the derivatives analytically. These are the practical difficulties arising out of the model postulate, which is nonlinear, and the need to have flexibility of handling different model structures conveniently without software changes [29].

The need for explicit derivation of the sensitivity coefficients, Eq. (4.31) and (4.32), can be eliminated by approximating the sensitivity coefficients by numerical differences. The forward difference approximation yields each element of the response gradient matrix:

$$\left[\frac{\partial y(t_k)}{\partial \Theta}\right]_{ij} \approx \frac{y_i^p(t_k) - y_i(t_k)}{\delta \Theta_j}; \quad i = 1, \ldots, n_y; \quad j = 1, \ldots, n_q$$

$$\approx \frac{g_i(x^p(t_k), u(t_k), \Theta + \delta\Theta_j e^j) - g_i(x(t_k), u(t_k), \Theta)}{\delta \Theta_j} \tag{4.33}$$

where n_y is the number of output variables, n_q the total number of unknown parameters, superscript p the perturbed variables, $\delta\Theta_j$ a small perturbation in the jth component of Θ, e^j a column vector with one in the jth row and zeros elsewhere, $y^p(t_k)$ the perturbed response variables corresponding to the perturbation $(\Theta + \delta\Theta_j\, e^j)$, and $y(t_k)$ the unperturbed responses corresponding to Θ. The perturbed responses $y^p(t_k)$ at each time point are obtained from the perturbed system equations. These equations have a form similar to those of Eqs. (4.10) and (4.11), except that each time the parameter vector Θ is replaced through the perturbed parameter vector Θ_p. The reader can easily write these equations. The perturbed responses $y^p(t_k)$ are also functions of the perturbed states $x^p(t_k)$, which are once again obtained by numerically integrating the perturbed state equations, which are basically the same state equations, except for a different

set of parameter values. Thus, we use the same model and thereby avoid the changes to the estimation software.

Thus, the finite difference approximation of the response gradients provides a more flexible approach to handle nonlinear models. The sensitivity coefficients in Eq. (4.33) were approximated by forward differences. A more accurate method would be based on the two-sided, central difference formula. It, however, requires two evaluations of the perturbed states and of the response variables at each time point and needs twice the computational time. From our experience, the one-sided forward difference approximation works fairly well in most of the cases. For large-scale problems (that is, large numbers of outputs n_y, parameters n_q, and data points N), near the minimum of the cost function a better performance can be obtained through central differencing [30]. The other alternatives that also provided good results in such cases were forward difference approximation in conjunction with the line search that we will discuss in Sec. 4.10.2 to overcome stalling of the optimization or to apply the Levenberg–Marquardt method we will address in Sec. 4.13.

The choice of the parameter perturbation will affect the accuracy of the approximated sensitivity coefficients. Some care is necessary in choosing the perturbation size; it should be small enough to give a valid approximation of the true gradients, but too small a value might pose difficulties due to machine accuracy and roundoff errors. We have already discussed this aspect of tradeoff in Sec. 3.5.3 in connection with approximating system matrices. Because the parameters might vary in the order of their magnitudes, a relative parameter perturbation is more appropriate. Typically,

$$\delta\Theta_j = 10^{-6}\,\Theta_j \tag{4.34}$$

is found to be a reasonable choice. In some cases, larger perturbation might be required, for example, when the particular parameter appears as a discrete nonlinear term, or when the effect (changes) in the system response because that parameter is small. If a particular parameter happens to be zero, either as initially specified or during iteration, then deviating from Eq. (4.34) it is necessary to specify a small perturbation; otherwise, Eq. (4.34) would yield zero perturbation, and the computed gradient would be zero. A small absolute value, say, $\delta\Theta_j = 10^{-6}$, is usually adequate to overcome this numerical peculiarity.

The procedure elaborated already is a classical approach in which the parameters are varied one at a time [18, 20, 21, 29]. Thus, for n_q number of unknown parameters, at each time point we have to solve n_q times the perturbed state equations. More recently, in connection with the stochastic optimization a procedure called "simultaneous perturbations" has been investigated [31], requiring just two evaluations at each discrete time point. For large n_q, such a method will be much faster. We have not investigated such a concept of simultaneous perturbation for dynamic systems and in the framework of Gauss–Newton method, and hence do not go into any further discussion of it.

4.9 AUTOMATIC GRADIENT COMPUTATION

As elaborated in Sec. 4.8, gradients are usually approximated using either one-sided (forward or backward) or two-sided central differencing. The latter is more accurate, but requires twice the computational time. Basically, in both the cases proper choice of step size is critical to obtain valid and accurate approximations. If the step size is too small, the roundoff errors (resulting from subtraction of two almost equal numbers) may dominate; on the other hand, too large a step can lead to nonnegligible truncation errors (resulting from neglected higher-order terms of Taylor series) yielding an erroneous gradient. Some tradeoff is usually necessary between these errors. We had resorted to the finite difference approximation due to various reasons, for example:

1. Analytical differentiation of the postulated model can be very tedious for complex models.
2. Flexibility is needed to consider different nonlinear model structures without programming changes each time.
3. In some cases a program may be available just for computing the desired function.

However, if we can find a technique to do all of these jobs automatically and also generate program code equivalent to an analytic derivative without actually going through the steps of deriving the mathematical expressions, we will not only achieve the desired goal, but also improve the accuracy by eliminating the errors introduced by finite differencing. This becomes possible through an upcoming technology in the field of mathematical computations, called automatic differentiation (AD) [32, 33]. We cover here just the basic fundamentals of this recent approach to indicate new possibilities.

The concept of automatic differentiation is based on the fact that any function, no matter in which programming language, is executed as a sequence of elementary operations such as additions, multiplications, and intrinsic functions. Such basic operations and functions are limited in number, and their analytic derivatives are known. By repeated application of chain and product rules, any function can be differentiated. To illustrate the procedure, we quote here a simple example from [33] considering a function $y = (x + x^2)^2$. Figure 4.3a gives a possible way of programming the preceding function and Fig. 4.3b that for the automatically generated gradient, where the value of dx is arbitrary but not zero.

In simple terms, given a program code to evaluate a function in terms of the dependent and independent variables, the AD processor first synthesizes (breaks down) the code in terms of elementary operations. Based on this code synthesis, a program code is constructed to compute the derivative of the dependent variables with respect to the independent variables. Because the derivatives are computed now using elementary and intrinsic functions, the approximation errors vanish,

Fig. 4.3 Sample example of automatic differentiation by chain rule.

a) Function evaluation

```
Function y = f(x)
z=x*x
w=x+z
y=w*w
end
```

b) Function and gradient evaluation

```
Function (y, dy) = fdot(x, dx)
z=x*x
dz = 2*x*dx
w = x+ z
dw = dx + dz
y= w*w
dy = 2 * w * dw
end
```

giving derivatives to machine precision. The approach being analytical, it does not involve perturbations, and hence roundoff or truncation errors are eliminated.

Thus, automatic differentiation provides an option to overcome numerical difficulties that might arise from finite difference approximations. Automatic differentiation will perform better, particularly in those optimization cases where the algorithm is sensitive to gradients. Applications of this new approach have been reported in few fields based on static models (that is, systems not incorporating state equations). Preliminary investigations suggest that the approach can be applied to dynamic systems with state and observation equations. Utility in terms of estimating aerodynamic derivatives and catering to special cases where finite differences posed severe problems will have to be established through applying the new approach to a large number of cases. The only minor issue appears to be that of using a generated code that looks like a black box and thereby tends to loose engineer's insight into a program code he or she might use.

4.10 STEP-SIZE CONTROL

The unconstrained Gauss–Newton method, described in Sec. 4.6, takes a full step of parameter update at each iteration. Generally, the Gauss–Newton method performs very well. However, as presented in Secs. 4.6 and 4.7, the update is based on the assumption of local linearity and a quadratic cost function. Furthermore, an approximation to the second gradient is incorporated to reduce the numerical burden, and the response gradients are approximated through numerical approximations. Because of these limitations, the Gauss–Newton method might perform poorly, for example, due to initial values far away from optimum or due to numerical errors introduced through the finite difference approximation of gradients near the optimum, and it might show intermediate local divergence or stalling. In such cases it is necessary to control the step size carefully once the direction of the correction vector has been established. Three options are possible to overcome these typical numerical difficulties:

1. Heuristic approach of parameter halving
2. Line search
3. Dominant directions approach

4.10.1 HEURISTIC APPROACH

In the first approach based on heuristic considerations, if the full step of the Gauss–Newton algorithm, Eq. (4.29), diverges at any iteration, the parameter update $\Delta\Theta$ is successively reduced by halving each time until reduction in the cost function compared to the prior iteration is achieved. It amounts to a successive reduction of the full step through the factors of (2, 4, 8, . . .). During these halving steps, just the cost-function evaluation is necessary, without computing the gradient. Halving of the parameter increments is generally performed a maximum of 10 times, which amounts to a factor of ($2^{10} = 1024$), and thereby at the end of 10 halving steps we practically reach the parameters of the preceding iteration. It implies that under given conditions the current iteration does not lead to any further improvement in the optimization. In that case, the optimization procedure is terminated. Usually few halving steps are sufficient to overcome the intermediate divergence. This heuristic procedure is quite simple and in many cases found to be adequate.

4.10.2 LINE SEARCH

A more systematic approach and one that guarantees efficient performance is based on introducing a line search in the Gauss–Newton method. The update to Eq. (4.29) in such a case becomes

$$\Theta_{i+1} = \Theta_i + \alpha_i \Delta\Theta \quad \text{with} \quad \Delta\Theta = -\mathcal{F}^{-1}\mathcal{G} \tag{4.35}$$

where α_i is determined by a line search algorithm in each iteration. Line searches, also called one-dimensional searches, are the basic optimization procedures of finding a minimum of a cost function along the specified search direction. In the present case it amounts to reducing or increasing $\Delta\Theta$ of the basic Gauss–Newton method optimally to obtain maximum reduction in the cost function at each iteration.

Because just a few function evaluations are necessary for line search, the additional computational overhead is marginal. Such damping strategies help to widen the convergence region and to overcome other numerical problems [18, 34, 35]. Any standard line search algorithm can be applied here; a quadratic line search procedure has shown good performance in many practical problems. Line search algorithm is also necessary for the bounded-variable Gauss–Newton method, which will be discussed in the next section.

4.10.3 DOMINANT DIRECTIONS

The third approach to overcome intermediate divergence or stalling is based on evaluating the eigenvalues of the information matrix $\mathcal{F}$ and determining the dominant search directions by checking the smallest and the largest eigenvalues. To understand the basic philosophy behind this approach, we begin by analyzing

the various contributions to the parameter step $\Delta\Theta$. Recall that the parameter step is given by

$$\Delta\Theta = -\mathcal{F}^{-1}\mathcal{G} \tag{4.36}$$

where $\mathcal{F}$ is the information (Hessian) matrix and $\mathcal{G}$ is the gradient vector. If $\mathcal{F}$ is ill-conditioned, the Gauss–Newton method can result in an extremely large step and divergence. We now attempt to detect the ill-conditioning through singular value decomposition (SVD). We know that $\mathcal{F}$ is a $(n_q \times n_q)$ real, symmetric matrix. In this case we can write $\mathcal{F}$ as follows:

$$\mathcal{F} = T\Lambda T^{-1} \tag{4.37}$$

where Λ is the diagonal matrix of eigenvalues of $\mathcal{F}$ and T is the matrix of eigenvectors. By inverting the preceding equation and using the matrix algebra results $(AB)^{-1} = B^{-1}A^{-1}$ and $T^{-1} = T^T$ for orthogonal matrices, we obtain

$$\mathcal{F}^{-1} = \left(T\Lambda T^{-1}\right)^{-1} = \left(T^{-1}\right)^{-1}\Lambda^{-1}T^{-1} = T\Lambda^{-1}T^T \tag{4.38}$$

Thus, the parameter improvement $\Delta\Theta$ is given by

$$\Delta\Theta = -T\Lambda^{-1}T^T\mathcal{G} \tag{4.39}$$

Equation (4.38) can be rewritten in SVD form as

$$\mathcal{F}^{-1} = \frac{1}{\lambda_1}t_1t_1^T + \frac{1}{\lambda_2}t_2t_2^T + \cdots + \frac{1}{\lambda_q}t_qt_q^T \tag{4.40}$$

where the matrix of eigenvectors is partitioned into n_q column vectors as $T = [t_1, t_2, \ldots, t_q]$ corresponding to the $(\lambda_1, \lambda_2, \ldots, \lambda_q)$ eigenvalues.

Substituting Eq. (4.40) into Eq. (4.36) leads to

$$\Delta\Theta = -\frac{t_1^T\mathcal{G}}{\lambda_1}t_1 - \frac{t_2^T\mathcal{G}}{\lambda_2}t_2 - \cdots - \frac{t_q^T\mathcal{G}}{\lambda_q}t_q \tag{4.41}$$

which is a well-known result in the optimization theory, saying that the parameter step $\Delta\Theta$ is a sum of n_q steps of magnitude $[t_i^T\mathcal{G}/\lambda_i]$ in the directions of the eigenvectors t_i, $i = 1, 2, \ldots, n_q$ of the information matrix.

For an ill-conditioned information matrix $\mathcal{F}$, at least one of its eigenvalues is very small. Let us assume that there are r such eigenvalues that might be considered very small. Because the step sizes in Eq. (4.41) are inversely proportional to the value of the eigenvalues, it can lead to very large step sizes along these r eigenvectors and in worst cases to divergence. The intuitive idea is to neglect these so-called bad directions and consider only the remaining $(n_q - r)$ dominant

search directions. Mathematically, Λ and T are partitioned as

$$\Lambda = \begin{bmatrix} \Lambda_d & | & 0 \\ - & - & - \\ 0 & | & \Lambda_r \end{bmatrix} \quad \text{and} \quad T = [T_d \quad T_r] \tag{4.42}$$

such that the dominant eigenvalues greater than ε (a small positive number) are grouped together as Λ_d and the corresponding eigenvectors in the left partition T_d. Substituting Eq. (4.42) into (4.39) and simple manipulation leads to

$$\Delta\Theta = -T_d\Lambda_d^{-1}T_d^T\mathcal{G} - T_r\Lambda_r^{-1}T_r^T\mathcal{G} \tag{4.43}$$

and neglecting the r small eigenvalues gives the step in the dominant search direction as

$$\Delta\Theta \approx -T_d\Lambda_d^{-1}T_d^T\mathcal{G} \tag{4.44}$$

Let us discuss some practical issues and performance of the preceding modified form of the Gauss–Newton method. It requires specifying ε appropriately. This can depend upon the model size, modeling errors, and the noise in the data analyzed. In several practical cases of flight vehicle parameter estimation, it turned out that the preceding modification worked only in a third of the cases of an ill-conditioned matrix. This is not totally surprising because neglecting large $[t_i^T\mathcal{G}/\lambda_i]$ steps does not necessarily mean that new directions are used in the optimization. All of the remaining so-called dominant directions, in many cases a major number of them, are still the Gauss–Newton directions. A better way could be to generate a new set of search directions, as suggested in [35], possibly interpolating between Gauss–Newton and the Newton direction. The approach is again based on SVD and dominant directions. This is left to the reader to pursue, but with a remark that the best strategy would be one that is simple and straightforward like the heuristic approach or line search presented earlier.

4.11 BOUNDED-VARIABLE GAUSS–NEWTON METHOD

The Gauss–Newton method discussed in Sec. 4.6 provides an unconstrained solution to a nonlinear optimization problem. Parameter estimation subject to simple bounds might, however, be relevant in some cases. Typical applications related to flight vehicle system identification are the following [36]:

1. Parameters that describe the physical effects, in the present case aerodynamic effects, are often constrained to lie in a certain range, for example, the Oswald's factor [37] characterizing the increase in drag over ideal condition caused by nonelliptical lift distribution and interference is typically limited to less than one.

2. Time delays in the measurement variables are by definition always positive and hence greater than zero.
3. Estimation of highly nonlinear model parameters such as friction, which can lead to numerical difficulties due to different reasons like poor guess of initial values [38], is needed.
4. Attempt to estimate parameters with almost zero sensitivity, resulting either from the use of inappropriate maneuvers or observation variables in the estimation procedure can lead to divergence.

In this section we address the issues pertaining to extending the Gauss–Newton method to account for simple bounds.

The linearly constrained optimization problem in which the constraints are simple bounds on the variables is formulated as

$$\min_{\Theta} J(\Theta) \quad \text{subject to } \Theta_{\min} \leq \Theta \leq \Theta_{\max} \tag{4.45}$$

where $\Theta_{\min}$ and $\Theta_{\max}$ are the lower and upper bounds on the parameter respectively.

Optimization theory provides several approaches to solve this problem, such as the following:

1. Transformation techniques
2. Penalty function methods
3. Barrier function or Lagrangian approach
4. Active set strategy

The transformation technique is based on transforming the independent variables such that the constraints are satisfied automatically. Once this is done, any standard unconstrained optimization method can be applied [39]. Although the concept is simple to understand, it is not always possible to transform all constraints, and the form of the transformation equation might not be simple to find. The penalty function and Lagrangian approaches are more general and suitable for complex nonlinear inequality constraints. They are also more complex. In the present case we are considering a simpler problem of linear constraints. Several algorithms and software programs, for example, limited memory BFGS [40], Extrem [41], quasi-Newton [42], and bounded-variable Least Squares [43], provide solutions to this problem. However, as already pointed out, the Gauss–Newton method is preferred here. The active set strategy is conceptually very appealing and can be readily extended to the Gauss–Newton method [36].

Starting from the initially specified parameter values Θ_0, an active set IA containing the indices of the variables hitting the bounds is formed and updated for every iteration. A variable is called a free variable if it is within

the permissible bounds and hence not in the active set. The Gauss–Newton search directions for the free variables are computed as follows:

$$\Delta\Theta_{\text{free}} = -\mathcal{F}_{\text{free}}^{-1}\,\mathcal{G}_{\text{free}} \tag{4.46}$$

where the information matrix $\mathcal{F}_{\text{free}}$ and the gradient vector $\mathcal{G}_{\text{free}}$ are computed using Eq. (4.30) for the free variables. The parameter updates resulting from Eq. (4.46) are checked for the specified bounds, and any violation of the constraints leads to modification of the active set IA. For such parameters the values are set to the respective bounds and the search directions of Eq. (4.46) to zero. For the remaining free parameters a new set of values is computed using a line search, which is necessary because changes in the number of free parameters change the search directions. As already addressed briefly in Sec. 4.10.2, we have used the quadratic line search procedure in our applications. It is interesting to point out that because the bounded-variable Gauss–Newton method includes the line search as an integral part of the algorithm, it automatically eliminates the problems of poor performance due to intermediate divergence.

An important aspect of the active set strategy is to develop a procedure of finding variables leaving or entering the feasible region, and accordingly to alter the active set IA in each iteration as the optimization progresses. The active set is changed whenever a free variable hits its bounds during iteration. Furthermore, if the Kuhn–Tucker optimality conditions [39]

$$\mathcal{G}_i < 0, \quad \text{for } \Theta_i = \Theta_{i\_\max}$$

or (4.47)

$$\mathcal{G}_i > 0, \quad \text{for } \Theta_i = \Theta_{i\_\min}$$

are not satisfied for any of the variables in the active set, then those variables are dropped from the active set and made free; $\mathcal{G}_i$, Θ_i, $\Theta_{i\_\min}$, and $\Theta_{i\_\min}$ are respectively the components of the gradient vector given by Eq. (4.30), the current parameter value, and its lower and upper bounds. In other words, conditions checked in Eq. (4.47) guarantee that the gradients for the variables hitting the bounds are such that they point outwards of the feasible region, implying that any further minimization of the cost function would be possible only when the particular parameters are not constrained within the specified limits.

The computational overhead to implement the active set strategy in an existing unconstrained Gauss–Newton method is minor; it is just required to check for the variables that hit the bounds and for the optimality conditions of Eq. (4.47) to enter or drop parameters from the active set. The advantages of this particular approach are twofold: 1) it provides an efficient solution retaining the desirable properties of the Gauss–Newton method, namely, the quadratic convergence property and 2) statistical information regarding the accuracy of the estimates, which we will address in Sec. 4.17, is readily available.

4.12 CONSTRAINED GAUSS–NEWTON METHOD USING INTERIOR-POINT ALGORITHM

Yet another approach to constrained optimization that has attracted quite some attention in the area of nonlinear programming is based on the interior-point algorithm [44–46]. Without going into the historical background on the development of such techniques, we straightaway address the applicability of such an algorithm within the framework of the maximum likelihood estimation that we have so far considered. To restate the problem, the maximum likelihood estimates of the unknown parameters and of the unknown noise covariance matrix are obtained by minimizing the cost function:

$$J(\Theta, R) = \frac{1}{2}\sum_{k=1}^{N}[z(t_k) - y(t_k)]^T R^{-1}[z(t_k) - y(t_k)] + \frac{N}{2}\ell n|R| \tag{4.48}$$

where Θ is the n_q-dimensional vector of unknown parameters. Minimization of Eq. (4.48) yields the unconstrained estimates of Θ. In the present case, the parameters to be estimated are subjected to inequality constraints:

$$\Theta_{\min} \leq c(\Theta) \leq \Theta_{\max} \tag{4.49}$$

where $c(\Theta)$ are the inequality constraints. Through simple manipulation, the two-sided constraints of Eq. (4.49) can be transformed into the form:

$$\begin{aligned} c(\Theta) - \Theta_{\max} &\leq 0 \\ \Theta_{\min} - c(\Theta) &\leq 0 \end{aligned} \tag{4.50}$$

For simplicity of notation and for derivation of the algorithm, we express the constraints in Eq. (4.50) in the general form given by

$$c_i(\Theta) \leq 0, \quad i = 1, \ldots, p \tag{4.51}$$

where p is the total number of constraints and c can be of any form, but is assumed to be twice continuously differentiable. Thus, it is required to optimize the cost function of Eq. (4.48) subject to the constraints specified in Eq. (4.51). To apply the interior-point approach, we first reformulate the problem using the slack variables s_i to each of the constraints of Eq. (4.51).

$$\underset{\Theta,R}{minimize}\, J(\Theta, R) \quad \text{subject to } c(\Theta) + s = 0, \quad \text{and} \quad s \geq 0 \tag{4.52}$$

where c and s are p-dimensional vectors. The inequality constraints in Eq. (4.52) are eliminated by introducing a barrier parameter $\mu(>0)$, which results in the following problem:

$$\underset{\Theta,R}{minimize}\left\{J(\Theta, R) - \mu\sum_{i=1}^{p}\log s_i\right\} \quad \text{subject to } c(\Theta) + s = 0 \tag{4.53}$$

where $J(\Theta, R)$ is given by Eq. (4.48). Adopting the Lagrangian approach, we can rewrite Eq. (4.53) as

$$\tilde{J}(\Theta, R, s, \lambda, \mu) = \frac{1}{2}\sum_{k=1}^{N}[z(t_k) - y(t_k)]^T R^{-1}[z(t_k) - y(t_k)] + \frac{N}{2}\ell n|R| - \mu\sum_{i=1}^{p}\log s_i + \lambda^T(c(\Theta) + s) \tag{4.54}$$

where λ is the vector of Lagrange multipliers.

The first-order condition for a minimum is that the first derivatives of $\tilde{J}$ with respect to the respective variables, namely, Θ, R, s, and λ, vanish. As in the case of the classical unconstrained output-error method described in Sec. 4.5, we apply the two-step relaxation strategy, estimating R first and then the other parameters. Following the development presented in Sec. 4.5.2 and Appendix E, it can be shown that the estimate of the covariance matrix of residuals R is given by

$$R = \frac{1}{N}\sum_{k=1}^{N}[z(t_k) - y(t_k)][z(t_k) - y(t_k)]^T \tag{4.55}$$

which is the same expression as given by Eq. (4.15).

For a known R given by Eq. (4.55) during the particular iteration, equating $\partial\tilde{J}/\partial\Theta$, the partial differentiation of Eq. (4.54) with respect to Θ, to zero yields

$$-\sum_{k=1}^{N}\left(\frac{\partial y(t_k)}{\partial\Theta}\right)^T R^{-1}[z(t_k) - y(t_k)] + \left(\frac{\partial c}{\partial\Theta}\right)^T\lambda = 0 \tag{4.56}$$

Similarly, partially differentiating Eq. (4.54) with respect to s and λ and equating to zero leads respectively to

$$-\mu e + S\lambda = 0 \tag{4.57}$$

and

$$c(\Theta) + s = 0 \tag{4.58}$$

where S is a diagonal matrix with elements s_i and e is a vector of ones. It can be shown that Eqs. (4.56–4.58) represent the so-called Karush–Kuhn–Tucker conditions for the constrained minimum.

As in the case of the standard unconstrained Gauss–Newton method, we start from a valid (interior) point Θ_0, s_0, λ_0. Without going into the detailed mathematical derivation of individual steps, we state here the final expression for $\Delta\Theta$,

Δs, $\Delta\lambda$ updates. For a prespecified μ, the Gauss–Newton step is computed according to

$$\left\{\sum_{k=1}^{N}\left(\frac{\partial y(t_k)}{\partial\Theta}\right)^T R^{-1}\left(\frac{\partial y(t_k)}{\partial\Theta}\right)+\left(\frac{\partial c}{\partial\Theta}\right)^T S^{-1}\Lambda\left(\frac{\partial c}{\partial\Theta}\right)\right\}\Delta\Theta$$
$$=\sum_{k=1}^{N}\left(\frac{\partial y(t_k)}{\partial\Theta}\right)^T R^{-1}[z(t_k)-y(t_k)]-\left(\frac{\partial c}{\partial\Theta}\right)^T S^{-1}\Lambda[s+c(\Theta)+\mu\Lambda^{-1}e] \quad (4.59)$$

$$\Delta s=-\left(\frac{\partial c}{\partial\Theta}\right)\Delta\Theta-c(\Theta)-s \quad (4.60)$$

$$\Delta\lambda=S^{-1}[\mu e-S\lambda-\Lambda\Delta s] \quad (4.61)$$

where Λ is a diagonal matrix with elements λ_i.

The matrix on the left-hand side of Eq. (4.59) is symmetric and positive definite. Any standard numerical procedure like Cholesky factorization can be used to solve Eq. (4.59) for the parameter update $\Delta\Theta$. The first term in the flower bracket on the left-hand side of Eq. (4.59) is the same as that appearing in Eq. (4.28) or (4.30) applying the unconstrained Gauss–Newton method. The full Gauss–Newton step, resulting from Eqs. (4.59–4.61), does not guarantee that the updated parameters satisfy all of the constraints. Therefore, a suitable damping strategy needs to be applied. The common approach is

$$\Theta_{i+1}=\Theta_i+\alpha_P\,\Delta\Theta_i \quad (4.62)$$
$$s_{i+1}=s_i+\alpha_P\,\Delta s_i \quad (4.63)$$
$$\lambda_{i+1}=\lambda_i+\alpha_D\,\Delta\lambda_i \quad (4.64)$$

where i is the iteration index and (α_P, α_D) are the damping coefficients. These coefficients guarantee that the nonnegative variables remain nonnegative and prevent the estimates from approaching the boundaries too closely [44, 45]. They are chosen based on the standard ratio test.

$$\alpha_P=\min\left\{0.9995\min_{(j:\Delta z_j<0)}\{-s_j/\Delta s_j\},\ 1\right\} \quad (4.65)$$

$$\alpha_D=\min\left\{0.9995\min_{(j:\Delta y_j<0)}\{-\Lambda_j/\Delta\lambda_j\},\ 1\right\} \quad (4.66)$$

The last aspect of the interior point algorithm pertains to the choice of the barrier parameter μ. In general, the choice is model and data dependent. However, the

following procedure was also found quite adequate for a number of examples of flight data analysis [45, 46].

$$\mu = \frac{\lambda^T s}{(n_q + p)^2} \tag{4.67}$$

It is fairly obvious that the interior point algorithm with μ equal to zero reduces to the unconstrained method. For $\mu > 0$, the algorithm ensures that the estimated parameters remain within the specified constraints.

For linear constraints specified as simple upper and lower bounds, the two approaches to account for the limits, namely, the bounded-variable Gauss–Newton method described in Sec. 4.11 and the interior-point algorithm described in this section, were found to be comparable in terms of convergence, numerical values, and accuracies. The more sophisticated formulation based on the interior-point will be efficient when more complex constraints have to be satisfied.

4.13 LEVENBERG–MARQUARDT METHOD

The Levenberg–Marquardt method is yet another approach to overcome the numerical problems discussed in the preceding sections [47, 48]. It combines the best features of the standard, unconstrained Gauss–Newton method and the steepest descent technique that moves in the gradient direction. Because of the optimal combination of two search directions, the Levenberg–Marquardt method has a wider convergence region. The update formula in such a case is given by

$$\Theta_{i+1} = \Theta_i + \Delta\Theta \quad \text{with } (\mathcal{F} + \lambda I)\Delta\Theta = -\mathcal{G} \tag{4.68}$$

where λ is the Levenberg–Marquardt (LM) parameter, which can be smoothly varied to control whether the update search direction is more like steepest descent or along the Gauss–Newton direction. It is easily seen from Eq. (4.68) that for $\lambda \to \infty$ the method becomes steepest descent, and as $\lambda \to 0$, it becomes like Gauss–Newton.

The computation of $\mathcal{F}$ and $\mathcal{G}$ is exactly the same as in the preceding case of the Gauss–Newton method, given by Eq. (4.30). It now remains to find proper value for λ that performs optimum interpolation between the two methods. Many methods exist for computing the LM parameter λ. The control strategy described in [47] is simple and found to work well in practical cases.

1. Choose a reduction factor $\nu > 1$, typically $\nu = 10$; choose $\lambda_0 = 0.001$ as initial starting value.
2. Perform one step of the Gauss–Newton method, and increment the iteration counter i by one; let λ^{i-1} denote the value from the preceding iteration.
3. Solve the system of linear equations, Eq. (4.68), for two values of the LM parameter λ^{i-1} and (λ^{i-1}/ν).

4. Now, for the two solutions of $\Delta\Theta$ from step 3, perform the parameter update, and compute the respective cost functions $L_i = L(\Theta(\lambda^{i-1}))$ and $L_i^{\nu} = L(\Theta(\lambda^{i-1}/\nu))$.
5. Comparison of L_i and L_i^{ν} leads to the following three possible cases:
   a. If $L_i^{(\nu)} \leq L_{i-1}$, then set $\lambda^{(i)} = \lambda^{(i-1)}/\nu$ and $\Delta\Theta = \Delta\Theta(\lambda^{i-1}/\nu)$. (That is, smaller λ results in reduced cost function; hence, accept the new point and reduce the LM parameter for the next iteration.)
   b. If $L_i^{(\nu)} > L_{i-1}$ and $L_i \leq L_{i-1}$, then set$\lambda^{(i)} = \lambda^{(i-1)}$ and $\Delta\Theta = \Delta\Theta(\lambda^{i-1})$. (That is, accept the new point, but retain the LM parameter from preceding iteration.)
   c. Otherwise, $L_i^{(\nu)} > L_{i-1}$ and $L_i > L_{i-1}$, then set $\lambda^{(i-1)} = \lambda^{(i-1)}\nu$, (i.e., increase λ), and repeat the steps 2 through 5, until an acceptable new point is found.

In simple words, the control strategy just elaborated evaluates the cost function values (among themselves and compared to the previous lowest cost) for two LM parameters. The one corresponding to the most reduction is accepted, and the program proceeds to the next iteration. In case both of the new updates have costs that are higher than the cost in the previous iteration, the factor λ is increased, and the cycle repeated. Adaptation of the LM parameter generally guarantees convergence to the optimum.

The choice of the reduction factor ν is arbitrary, and $\nu = 10$ has been found to be a good choice for many test cases pertaining to aircraft parameter estimation. Because the properties of the gradient optimization methods are not scale invariant, the starting value for the LM parameter λ depends on the order of magnitude of the system (information) matrix. A scaling, widely used in linear least-squares problems, is given by

$$\mathcal{F}^* = (f_{ij}^*) = \left(\frac{f_{ij}}{\sqrt{f_{ii}}\,\sqrt{f_{jj}}}\right) \tag{4.69}$$

$$\mathcal{G}^* = (g_i^*) = \left(\frac{g_i}{\sqrt{f_{ii}}}\right) \tag{4.70}$$

This leads to a scaled system of linear equations:

$$\mathcal{F}^*\Delta\Theta^* = -\mathcal{G}^* \tag{4.71}$$

which can be solved by any standard procedures of linear algebra. The solution for the transformed $\Delta\Theta^*$ is then scaled back as

$$\Delta\Theta_i = \frac{\Delta\Theta_i^*}{\sqrt{f_{ii}}} \tag{4.72}$$

The transformed matrix $\mathcal{F}^*$ is a simple correlation matrix, whose diagonal elements are one. For such a scaling, a starting value of $\lambda_0 = 0.001$ is recommended, which proved adequate for several test cases investigated [30, 49].

The Levenberg–Marquardt method requires solving Eq. (4.68) twice per iteration compared to once in the Gauss–Newton method. However, this overhead is just a minor part of the total overhead; as elaborated in Sec. 4.18, a large portion being spent on computing the information matrix, that is, the second gradient of the cost function, which is done only once in both the cases. The adaptation of the LM parameter ensures that the convergence is monotonic and guarantees convergence from iteration to iteration, and as such the procedures described in Sec. 4.10 to overcome intermediate divergence are not relevant here. The Levenberg–Marquardt method works well for starting points far from the optimum. It is also useful to overcome the numerical problems due to one-sided approximation of gradients near the optimum.

4.14 DIRECT SEARCH METHODS

Optimization methods that require just function evaluations and not gradients belong to the class of direct search methods. They are, in general, much slower compared to gradient-based methods [29, 50, 51] but are somewhat more robust, particularly starting from a point that is far from the optimum. Moreover, standard program codes are readily available. Because the literature on optimization theory is abundant with the details of these methods, only a very brief verbal description is provided here.

4.14.1 SIMPLEX AND SUBPLEX METHODS

The downhill simplex method developed by Nelder and Mead and the subspace searching method developed by Rowan are based on the principles of reflection, expansion, contraction, and shrinkage (massive contraction) of an initially specified q-dimensional convex hull, a geometrical figure with $n_q + 1$ vertices called simplex [52, 53].

In the Nelder and Mead simplex method, a simplex moves through the q-dimensional space of the cost function, changing shape and size, and automatically shrinks when the minimum is enveloped by the simplex [52]. The simplex method is widely used in the optimization of cost functions in several applications, but rarely for flight vehicle parameter estimation from flight data. Nevertheless, because of its robustness with respect to the initial guess of the unknown parameters, it could be useful in a few cases, for example, estimation of parameters with discrete nonlinearities, friction parameters, etc.

The subplex method is a generalization of the simplex method of Nelder and Mead. It decomposes the higher-dimensional problem into smaller-dimensional subspaces in which the simplex method can search efficiently [53]. The task is

twofold: first, to determine the improved set of subspaces, and second to apply the simplex method. The concept of orthogonality used for the subspaces is similar to the concept of primary and secondary search directions used in Jacob's heuristic minimum search method called "Extrem"; see Sec. 4.14.3.

When the number of parameters to be estimated is large, the classical simplex method of Nelder and Mead rapidly becomes inefficient. A restart procedure helps in some cases to improve the performance somewhat. On the other hand, the computational overhead for the subspace searching method increases roughly linearly with the problem dimension. From this viewpoint, it is considered as the most promising of the simplex-based methods.

4.14.2 POWELL'S METHOD

Powell's method is a widely used direct search method [54]. It can be shown to be a method of conjugate directions that minimizes the cost function in a finite number of cost function evaluations. The method is an iterative procedure and involves computing the search directions at each iteration and finding the minimum along these search directions applying a line search method.

4.14.3 JACOB'S METHOD

The Jacob's method is based on the choice of optimal search directions, determination of the optimum along a line and defining the step size for each search [41]. The Gram–Schmidt orthogonalization procedure is used to select the orthogonal search directions, in contrast to the conjugate directions used in Powell's method. The search directions are adapted to the local properties of the parameter space being searched. Basically, the procedure is very similar to the Gauss–Newton method. A program called "Extrem," which is available in the literature, caters to constrained optimization. For example, simple bounds on the parameters being estimated can be accounted for.

Having covered in Secs. 4.4–4.14 several methods to optimize the cost function, the choice of a particular method might appear to be a difficult one. However, based on comparative studies performed in the past [29, 50, 51], it can be concluded that the direct search methods covered in Sec. 4.14 are extremely slow compared to the gradient-based Gauss–Newton, its variation accounting for bounds or the Levenberg–Marquardt method. The last mentioned methods are far superior in performance, mainly because they make use of the first and second gradients to predict the minimum and directly step in the best local direction. Any one of these will perform equally well on the chosen problem. We have widely used the Gauss–Newton and bounded-variable Gauss–Newton method, incorporating either line search or the heuristic approach of parameter halving to overcome any intermediate divergence or stalling of the optimization procedure. Nevertheless, in certain rare cases, like estimation of friction parameters, the simplex or subplex may give better starting values for the parameters.

It is also possible to switch from one method to another as the optimization progresses.

4.15 REGRESSION STARTUP PROCEDURE

In Sec. 4.10, it has been pointed out that the Gauss–Newton method might perform poorly for initial values far away from optimum, and we had discussed some techniques to overcome this problem. Yet another way would be to generate better starting values using some other technique that does not need initial guess values. This is possible by using a procedure belonging to the broad class of least-squares methods that we call regression startup method. It represents a mixed least-squares and output-error approach. To illustrate the method, first consider the linearized system representation:

$$\dot{x}(t) = A(\beta)x(t) + B(\beta)u(t) \tag{4.73}$$

$$y(t) = C(\beta)x(t) + D(\beta)u(t) \tag{4.74}$$

Now, assuming that measurements of all of the state variables are available and denoting them as x_m, the system equations are reformulated as

$$\dot{x}(t) = A_F x(t) + [B(\beta); \quad A_E(\beta)] \begin{bmatrix} u(t) \\ x_m(t) \end{bmatrix} \tag{4.75}$$

$$y(t) = C_F x(t) + [D(\beta); \quad C_E(\beta)] \begin{bmatrix} u(t) \\ x_m(t) \end{bmatrix} \tag{4.76}$$

The matrices A and C are resolved each into two, namely, (A_F, A_E) and (C_F, C_E). The matrices with the subscript F contain the possibly fixed coefficients, if any, and those with the index E contain the parameters to be estimated.

In general, the fixed parameters appearing in matrices A_F and C_F correspond to those resulting from kinematic terms or derivatives fixed using the a priori values whereas those appearing in A_E and C_E correspond to stability and control derivatives, which are unknown. This reformulation makes the state variables x and thereby the observation variables y and hence the error $z(t_k) - y(t_k)$ linear in the parameters being estimated. For such cases, the cost function being quadratic, the least-squares method will yield estimates in a single iteration. A detailed treatment of the least-squares technique will be dealt with in Chapter 6. The preceding formulation has the advantage that the initial values of the unknown parameters are not required. The optimization procedure can be started with zero values of all of the parameters. Note that the preceding formulation retains state equations that have to be integrated, but it incorporates measured states as control inputs in conjunction with unknown parameters, thus leading to a combined output-error and least-squares approach. The optimization of the cost function is carried out using the same iterative algorithm that we have studied earlier (Sec. 4.7).

Applying regression analysis to the nonlinear model postulated in Eqs. (4.10) to (4.12) is not straightforward. If the model is linear in parameters, the optimization algorithm is a one-shot procedure whereas for a model nonlinear in parameters, the iterative optimization is necessary. In such a case some reasonable start values might be required. Treatment of unknown initial conditions is a little trickier. However, based on practical considerations, an acceptable solution becomes feasible. Assuming that the measurements of the state variables x_m are available; Eqs. (4.10) to (4.12) are reformulated as follows:

$$\dot{x}(t) = f[x(t), x_m(t), u(t) - \Delta u(b_{u,l}), (\beta_{xF}, \beta_{xE})], \quad x(t_{0,l}) = x_m(t_{0,l}) \tag{4.77}$$

$$y(t) = g[x(t), x_m(t), u(t) - \Delta u(b_{u,l}), (\beta_{xF}, \beta_{xE}), (\beta_{yC}, \beta_{yE})] + \Delta z(b_{y,l}) \tag{4.78}$$

where the unknown parameters β_x and β_y are subdivided into two components (β_{xF}, β_{xE}) and (β_{yF}, β_{yE}) corresponding to those parameters that are constants and those that are to be estimated. The parameters to be estimated are multiplied by the measured states x_m or by measured control inputs u whereas the constant parameters are multiplied by the integrated states. The initial values $x(t_0)$ are set to the measured values at time t_0 and held fixed. Furthermore, the zero shifts in the control variables Δu, which represent the systematic errors, are assumed to be small and are, hence, either set to zero or kept fixed at the value specified. Estimation of the zero shifts in the observation variables Δz poses no difficulties because the model is linear in them.

Thus, associating the unknown parameters with the measured states leads to a mixed output-error and least-squares approach, which can be used to generate good initial guess values for the unknown parameters, both for linear and nonlinear model postulates. As will be discussed in Sec. 9.5, such a mixed least-squares and output-error formulation is also useful in estimating parameters of unstable aircraft without facing the numerical problems otherwise encountered in applying the classical output-error method. In a few cases the preceding general approach has been used without subdividing the matrices and using the measured states throughout. Such a procedure is much simpler to implement and found to work equally well.

4.16 ESTIMATION ACCOUNTING FOR A PRIORI INFORMATION

In some cases a priori information about certain aerodynamic derivatives might be available from wind-tunnel tests or from estimation results of other flight-test investigations. The Bayesian estimator enables accounting for a priori information in a probabilistic way by treating the parameter vector Θ $(n_q \times 1)$ as a random vector [55]. In such a case, a priori knowledge about the probability density functions of the measurements and of the parameters is necessary. The Bayesian estimator represents the most general type of parameter estimation methodology, but it also needs the maximum amount of a priori information.

The conditional probability density function $p(\Theta|z)$, which represents the a posteriori probability density function of the parameters Θ, given the measurements z, can be expressed using Bayes' rule as [55]

$$p(\Theta|z) = \frac{p(\Theta)p(z|\Theta)}{p(z)} \tag{4.79}$$

For convenience, both $p(\Theta)$ and $p(z|\Theta)$ are assumed to be Gaussian. In this case, they can be expressed as

$$p(z|\Theta) = \{(2\pi)^{n_y}|R|\}^{-N/2}\exp\left(-\frac{1}{2}\sum_{k=1}^{N}[z(t_k)-y(t_k)]^T R^{-1}[z(t_k)-y(t_k)]\right) \tag{4.80}$$

and

$$p(\Theta) = \{(2\pi)^{n_q}|R_2|\}^{-1/2}\exp\left(-\frac{1}{2}(\Theta-\Theta^*)^T R_2^{-1}(\Theta-\Theta^*)\right) \tag{4.81}$$

where R is the measurement noise covariance matrix, Θ^* is the vector of a priori values of the derivatives, and R_2 is the error covariance matrix of the a priori values. It follows that the a priori values and the associated uncertainty are represented as

$$E\{\Theta\} = \Theta^* \tag{4.82}$$

and

$$E\left\{(\Theta-\Theta^*)(\Theta-\Theta^*)^T\right\} = R_2 \tag{4.83}$$

In the estimation procedure, for a given set of measured data, $p(z)$ can be treated as a constant. Thus, maximization of $p(\Theta|z)$ in Eq. (4.79) is equivalent to maximization of J_{ap} given by

$$J_{ap} = (2\pi)^{-(n_q+n_yN)/2}|R|^{-N/2}|R_2|^{-1/2} \times\exp\left(-\frac{1}{2}\sum_{k=1}^{N}[z(t_k)-y(t_k)]^T R^{-1}[z(t_k)-y(t_k)] - \frac{1}{2}(\Theta-\Theta^*)^T R_2^{-1}(\Theta-\Theta^*)\right) \tag{4.84}$$

Setting the gradient with respect to Θ to zero and solving yields

$$\left\{\sum_k\left[\frac{\partial y(t_k)}{\partial\Theta}\right]^T R^{-1}\left[\frac{\partial y(t_k)}{\partial\Theta}\right] + R_2^{-1}\right\}\Delta\Theta = \sum_k\left[\frac{\partial y(t_k)}{\partial\Theta}\right]^T R^{-1}[z(t_k)-y(t_k)] - R_2^{-1}(\Theta-\Theta^*) \tag{4.85}$$

The just-discussed estimator is commonly called in the literature as the maximum a posteriori probability estimator (MAP). Loosely speaking, it pulls the estimates toward the a priori values, depending upon the confidence in them. Although it represents a more general approach, the a priori information about the probability density function $p(\Theta)$ is often difficult to obtain. Relaxing the assumption of parameters being random variables leads to the maximum likelihood estimation that we had already discussed in some depth in Secs. 4.2 through 4.5. To recall once again, the maximum likelihood method assumes that Θ does not depend on chance and thereby requires a priori knowledge only about $p(z|\Theta)$. In such a case Eq. (4.85) simplifies to Eq. (4.28). In other words, maximum likelihood estimation is a special case of MAP, which, in turn, is an approximation of the Bayesian estimator.

Because it is difficult in practice to specify accurately the error covariance matrix of the a priori values, a weighting factor W is additionally introduced in conjunction with R_2. In such a case, it can be shown that Eq. (4.85) leads to

$$\left\{\sum_k \left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}\left[\frac{\partial y(t_k)}{\partial \Theta}\right] + W\,R_2^{-1}\right\}\Delta\Theta = \sum_k \left[\left[\frac{\partial y(t_k)}{\partial \Theta}\right]^T R^{-1}[z(t_k) - y(t_k)]\right] - W\,R_2^{-1}(\Theta - \Theta^*) \tag{4.86}$$

Introduction of the weighting factor W provides a convenient means to study the effect of a priori values on the estimates. It is obvious from Eq. (4.86) that $W = 0$ implies suppression of the a priori values and $W \to \infty$ implies suppression of the measurements. In the latter case, the algorithm will neglect the measurements completely, yielding the a priori values as the estimates. This is a hypothetical case because if we are 100% sure that the a priori values are perfect, then we do not have to worry about the estimation from experimental data. This is in contradiction to our hypothesis. We have gone through the whole exercise of statistically estimating the parameters because we want to improve upon the accuracy of the estimated parameters and the model structure from the experimental data. This approach could, however, help to avoid convergence problems encountered when the parameters are near linearly dependent.

In practice, if the a priori information is to be included in the estimation, it is usually done for system parameters (that is, aerodynamic derivatives) only. When this option is invoked, it is necessary to specify the a priori values and corresponding error covariances along with the values for parameters appearing in the state equations. In practice, specifying an appropriate error covariance matrix R_2 for the a priori values and of the weighting factor W is difficult. It is this aspect that restricts the utility of this option. In general, this option is rarely used in flight vehicle system identification.

4.17 STATISTICAL ACCURACY OF PARAMETER ESTIMATES

Having applied the statistical estimation methods, the first and the most natural question that comes to our mind is about the statistical accuracy of the estimates that we have obtained from the experimental data. In other words, we would like to gain some appreciation of the confidence that we might have in the estimates obtained by applying the maximum likelihood or Bayesian estimator. In some sense the issue of accuracy of the estimates belongs to the category of tests that are required to evaluate the model adequacy and validation. We will dwell on this aspect separately in Chapter 11 dealing with various procedures and practical issues of model validation. However, for the sake of completeness, we cover here only the mathematical basics of statistical evaluation of the estimates.

From Sec. 4.3 and Appendix D, we know that the maximum likelihood estimator is asymptotically efficient in the sense of achieving the Cramér–Rao lower bound. It follows from Eq. (D35) that the parameter error covariance matrix P is given by the inverse of the Fisher information matrix $\mathcal{F}$ [Eq. (4.30)].

$$P = \left\{ \sum_{k=1}^{N} \left[\frac{\partial y(t_k)}{\partial \Theta} \right]^T R^{-1} \left[\frac{\partial y(t_k)}{\partial \Theta} \right] \right\}^{-1} \tag{4.87}$$

If a priori information, as discussed in Sec. 4.16, is included in the estimation, it is required to use the system matrix, that is, terms within the flower bracket on the left-hand side, of Eq. (4.86) while computing the preceding parameter error covariance matrix. The standard deviations and correlation coefficients of the estimates are given by

$$\sigma_{\Theta i} = \sqrt{p_{ii}} \tag{4.88}$$

$$\rho_{\Theta i \Theta j} = \frac{p_{ij}}{\sqrt{p_{ii} p_{jj}}} \tag{4.89}$$

The property of asymptotic efficiency is of practical significance. It implies that the ML estimator makes efficient use of the available data and that the Cramér–Rao lower bound indicates the theoretically maximum achievable accuracy of the estimates. Although Eqs. (4.88) and (4.89) provide a simple way to obtain theoretical information about the accuracy of the estimates and correlation among them, unfortunately, in practical applications this issue is a little more complex and not without ambiguities. We will discuss these implications separately in Chapter 11 dealing with model validation.

Because the information matrix $\mathcal{F}$ is required in the optimization of the cost function, the parameter error covariance matrix is automatically available without any extra computations. The standard deviations and correlation coefficients can be readily obtained. Thus, information on the statistical accuracy of the estimates is a byproduct of the Gauss–Newton or Levenberg–Marquardt methods. Other optimization methods we had just briefly mentioned, namely, Powell's, Jacob's,

simplex, and subplex, do not provide this information directly, and this is one of the limitations of these methods, besides being very slow in convergence.

4.18 ALGORITHMIC IMPLEMENTATIONS

The computational demands for large-scale systems and huge amounts of measured data required for global model identification are significant and can become prohibitive; the job can run to hours of CPU time [56]. Therefore, special emphasis is placed on computational details to achieve reasonable turnaround time in such cases. It makes use of the special structure of the information matrix resulting from multiple time segments (maneuvers) being processed.

As already elaborated in the preceding sections, the iterative unconstrained or bounded-variable Gauss–Newton methods require solving a linear algebraic equation once and the Levenberg–Marquardt twice per iteration. However, depending upon the number of unknown parameters being estimated, the dimension of the information matrix could be large, and the main computational burden is not due to solving this algebraic equation required for matrix inversion; see Eq. (4.29). The major portion of the computational time is consumed in computing the $n_y \times n_q$ dimensional sensitivity coefficients $\partial y/\partial \Theta$ and the $n_q \times n_q$ dimensional information matrix $\mathcal{F}$ over N data points; see Eq. (4.30).

As a typical case, we consider an example pertaining to the six-degree-of-freedom (DOF) equations of motion, with number of states $n_x = 10$, number of inputs $n_u = 12$, number of outputs $n_y = 19$, and the number of parameters to be estimated $n_q = 30$. Throughout the following comparative investigations, we use the fourth-order Runge–Kutta integration method. We perform five different evaluations using different number of maneuvers analyzed simultaneously each time, namely, $n_E = 10$, 20, 30, 40, and 60, with $N = 13{,}424$; 25,453; 37,720; 51,144; and 75,724 data points, respectively. In each case the initial conditions are specified appropriately (as average of first few points of measured data, see Sec. 3.7) and held fixed, and no other bias parameters are considered. Thus, we have in each case the same number of unknown parameters $n_q = 30$, and the information matrix is 30×30. On a workstation with 3.4-GHz CPU, they required on an average 9.1, 17.4, 25.7, 34.8, and 52 s per iteration to compute the information matrix $\mathcal{F}$ and gradient vector $\mathcal{G}$ (time linearly increasing with the number of data points) whereas the matrix inversion took negligible time (less than 0.01 s).

Now, let us consider estimating the initial conditions for each of the n_E time segments analyzed simultaneously. Then, the total number of parameters n_q is given by $30 + n_x \times n_E$, where in the present case the number of states is $n_x = 10$. Thus, in the five cases it amounts to estimating 130, 230, 330, 430, and 630 parameters, from which each time 30 are the aerodynamic derivatives. On the same 3.4-GHz workstation they required roughly 20, 78, 214, 485, and 1518 s per iteration to compute the information matrix $\mathcal{F}$ and gradients $\mathcal{G}$. The

information matrix $\mathcal{F}$ being symmetrical, we compute only the lower triangular part and then fill it up before inverting the same. Apart from this single consideration, no other special treatment is performed, implying that the buildup of $\mathcal{F}$ and $\mathcal{G}$ is done in a straightforward fashion using Eq. (4.30) for all n_q parameters. The matrix inversion required from 0.01 to 3 s is once again a negligible part of the overall computations even for the last case with 630×630 size matrix.

Thus, having substantiated our statement made earlier that the major portion of the computational time is consumed in computing the $n_q \times n_q$ information matrix, we now look for the ways to optimize computations to reduce the overall CPU time. If only aerodynamic parameters are estimated, keeping initial conditions and other bias parameters fixed, then no optimization is possible irrespective of how many n_E maneuvers are analyzed simultaneously. As just demonstrated, in such a case for a given model postulate the CPU time is linearly proportional to the number of data points. However, if initial conditions or bias parameters are also estimated, then it turns out that through a judicious implementation of the computations in the $\mathcal{F}$ matrix buildup pertaining to n_E time segments, significant reduction in the computational burden can be achieved. This becomes possible because concatenation of several time slices leads to a special block structure for the information matrix with respect to the initial conditions and other bias parameters. For illustration purposes, just three time segments and for each time segment the initial conditions and bias parameters of the observation equations are considered. The resulting matrix structure is shown schematically in Fig. 4.4. The nonzero elements appear in gray blocks

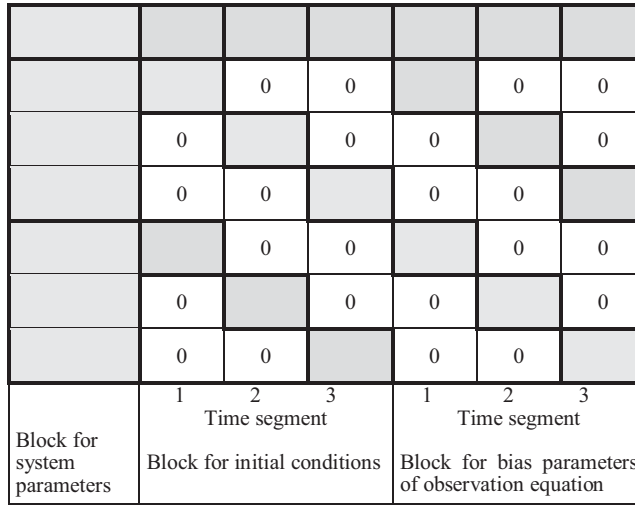

Fig. 4.4 Schematic of block structure of the information matrix for three time slices analyzed simultaneously.

with bold borders. The sparse matrix structure can be used advantageously to reduce the numerical computations while computing the $n_q \times n_q$ dimensional information matrix [57]. Furthermore, a part of the computation is done judiciously just once, namely, $[\partial y/\partial \Theta]^T R^{-1}$, which is common to both $\mathcal{F}$ and $\mathcal{G}$.

The use of this optimization led to CPU-time requirements of 12, 23, 35, 48, and 70 s per iteration for the five cases. Compared to the CPU times of 20, 78, 214, 485, and 1518 s using straightforward approach, the reduction is significant. The larger the number of time segments, the larger is the ratio of time saving. Because it is usually preferable to estimate the initial conditions and bias parameters, and the use of a large number of time segments is not uncommon, as well as 10 or more iterations are typically needed for the convergence, a significant amount of reduction in the computational overhead can be achieved by making optimum use of the special block structure for multiple maneuver evaluations. This issue is not so relevant for small-size problems, but it can become critical to ensure reasonable turnaround time during model development process for large-size problems encountered during global model identification addressed in Sec. 12.8.

Yet another special implementation feature pertains to avoiding large-size matrices through judicious implementation of computational sequence. Several estimation programs are based on computing system responses and those for the perturbed parameters required for approximating gradients, in a sequential way. In this direct approach, one parameter is perturbed at a time, and simulation is performed at a stretch over the complete set of N data points. The process is repeated n_q times, storing the responses in separate data arrays each time. This so-called sequential processing, shown on the left-hand side of Fig. 4.5, needs $(n_q + 1)$ arrays, each of size $(N \times n_y)$. At the end of such $(n_q + 1)$ simulations, the gradients are computed from these responses using Eq. (4.33) and summed over N data point according to Eq. (4.30). Thus, depending upon the number of data points N, the program might need very large size arrays; a program capable of analyzing up to 80,000 data points and 60 observations, 30 states, 60 inputs, 1200 parameters has been used in practice; in a rare case up to 150,000 data points or more have also been processed at a time.

For such large-size problems, the following alternative implementation, called pseudo-parallel processing, leads to substantial reduction in the storage needed. As depicted on the right-hand side of Fig. 4.5, instead of performing the simulations at a stretch over N data points for each perturbation, we handle each data point separately. At each data point, we perturb each of the n_q parameters, compute the responses, compute the gradients according to Eq. (4.30), and incrementally build up the matrices $\mathcal{F}$ and $\mathcal{G}$. Having done this for all n_q parameters, then we jump to the next data point, restore the perturbed states for continuation, and repeat the process to process the next point, until all of the data points are processed. Restoration of appropriate states at each point needs saving perturbed states for each parameter, requiring array of the size $n_x \times n_q$. This pseudoparallel approach characterized by horizontal processing at each point needs just two

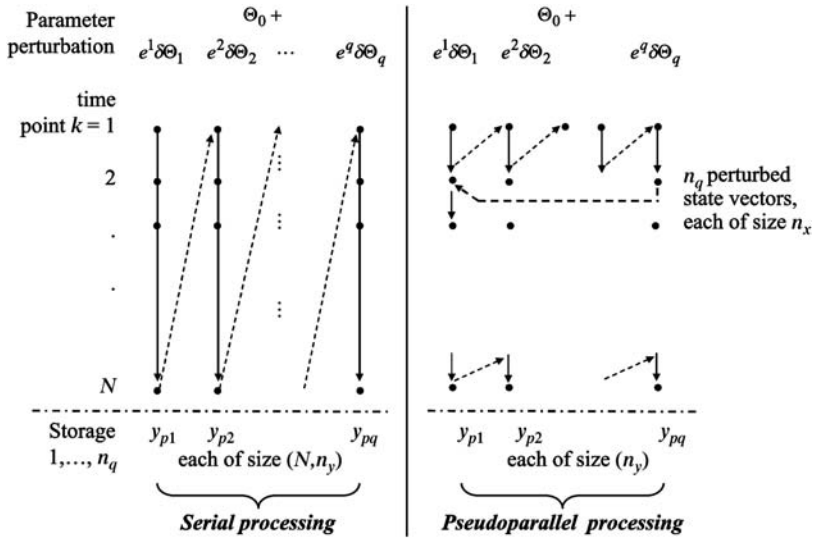

Fig. 4.5 Two different computational sequences for perturbed simulations.

arrays, one of size $(n_y \times n_q)$ and the other of $(n_x \times n_q)$. Compared to the direct implementation discussed earlier, requiring arrays of total size $(N \times n_y \times n_q)$, the storage requirements of the second approach are negligible. We have called this approach as pseudoparallel because, strictly speaking, the terminology of parallel processing has, in general, a different connotation, implying several simulations running at the same time whereas we still perform perturbations and simulations sequentially, but the order of performing the same over N data points and for n_q has been interchanged, leading to remarkable saving in the storage requirements.

4.19 OEM SOFTWARE

There are so very many ways to implement the output-error method; attempts to standardize the software program will be in vain because the model postulates vary from case to case. (Remember, we mostly deal with a general nonlinear system of arbitrary complexity.) Furthermore, the format of flight data to be analyzed will vary from organization to organization, more likely from project to project within the organization itself. Nevertheless, the basic parameter estimation algorithm remains the same. We provide in this book an extended version of software for general nonlinear systems. It caters to estimating not only the system parameters, time delays, and zero shifts, but also the initial conditions from multiple experiments. A more sophisticated estimation algorithm called

bounded-variable Gauss–Newton algorithm, discussed in Sec. 4.11, is also provided. In the case of linear systems, the bias parameters of the state and observation equations, as discussed in Sec. 3.5.2, can be included and estimated along with the system parameters. The software is quite adequate to analyze the test cases that are considered in this and in few other chapters, as well as for advanced and more complex applications. At the same time, it allows us to understand more closely the various algorithmic steps that we have so far studied in this chapter. To conform to the current trends, the provided software runs under Matlab® [58]. In general, the current trend shows that in the future more and more flight vehicle system identification work will be carried out under the Matlab® operating environment.

The extended version of the software caters to estimation of the following: 1) system parameters common to all time segments being analyzed, 2) initial conditions separately for multiple time segments being analyzed, 3) time delays, and 4) bias parameters separately for multiple time segments being analyzed simultaneously. These features have been achieved by treating the complete set of parameters into three different sets, namely, 1) system parameters (such as, for example, aerodynamic derivatives) common to all time segments, 2) initial conditions x_0 for each of the n_E experiments, and 3) separate bias parameters for each of the n_E experiments. If desired, the bias parameters can also be clubbed in the first category and treated as common to all time segments. This choice is left to the analyst. Note: because of the preceding extensions, which lead to highly flexible software and expand the scope significantly, the user interface to define the model has changed compared to that in the first edition of the book. As such, it is imperative that the model definition functions have to be adapted to the new structure and user interface.

Other estimation software packages are available as Matlab® Toolboxes based on restricted circulation or at a license fee [59, 60]. It is also possible to include models developed in Matlab/Simulink® using the block-diagram functionality [60]. Matlab® and Simulink provide an easy-to-use computing and modeling description environment. We do not go into the details of such software tools, except to say that they are based on the basic theory that we have covered in this book. More sophisticated software tools in FORTRAN language have been developed in the past and extensively used for flight vehicle system identification, for example, [57, 61, 62] among others. Here too, we do not go into any further details of these software tools.

Figure 4.6 shows a flowchart of the computational procedure for the output-error method. The function names for various parts are indicated at the bottom of each block. The source codes (Matlab m-files) for output-error method are provided in the directory /FVSysID2/chapter04/ and a README providing the details of various functions and test cases covered here. The starting point is the main program called "ml_oem2" and provides the user interface to define the model and to choose the optimization and integration methods and other information.

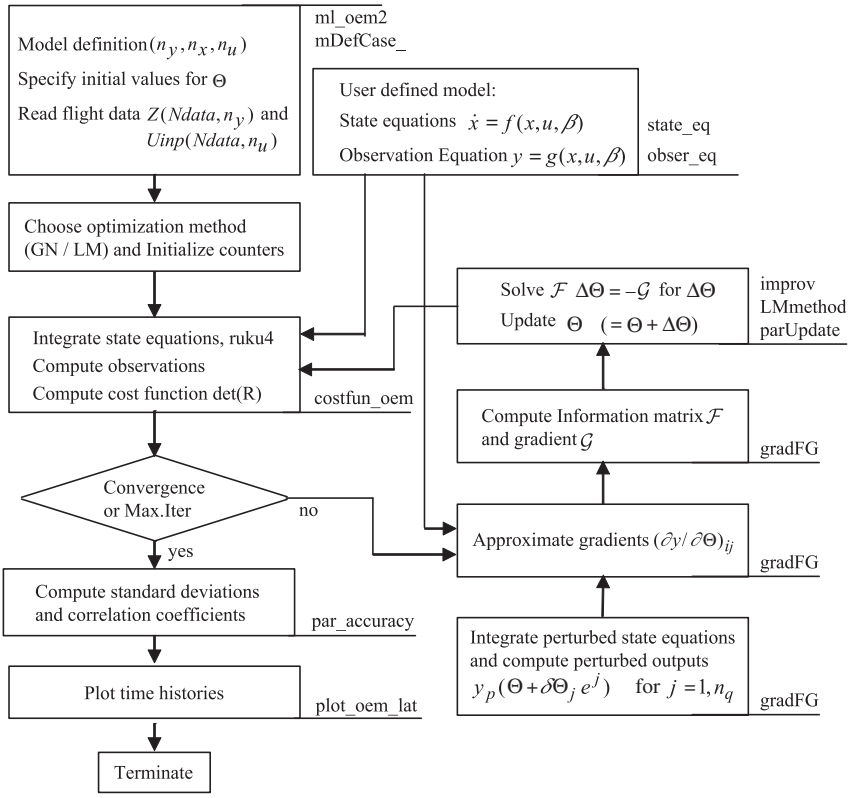

Fig. 4.6 Details of implemented output-error method.

test_case	index for the test case
iOptim	index for the optimization method 1: Gauss Newton; 2: Levenberg–Marquardt
integMethod	flag for integration method (ruku4, ruku3, ruku2)
niter_max	maximum number of iterations
tolR	relative change in the cost function for termination
par_step	parameter perturbation size for gradient approximations

Default values are defined for the maximum number of iterations niter_max, for the convergence criterion tolR and for the step size for parameter perturbations par_step. They can be suitably changed, if necessary.

A particular system model to be analyzed is denoted through the integer flag "test_case," which is to be uniquely associated with the state-space model and with the user-provided interface in the form of a function defining the model parameters, flight data, and other relevant details. Different state-space

models, to be coded by the user, are invoked by defining the flags and strings in a function mDefCase** called from "ml_oem2," where** denotes the test case index:

state_eq	function name: to code the right-hand side of state equations
obser_eq	function name: to code the right-hand side of observation equations
Nx	number of state variables (n_x)
Ny	number of output variables (n_y)
Nu	number of input variables (n_u)
NparSys	number of system parameters
Nparam	total number of unknown parameters
NparID	total number of parameters to be estimated (free parameters) (n_q)
param	starting values for unknown parameters (Θ)
parFlag	flags for free and fixed parameters = 1: free parameters (to be estimated) = 0: fixed parameters (not to be estimated)
param_min	lower bounds on the parameters; $= -\infty$ for unconstrained optimization
param_max	upper bounds on the parameters $= \infty$ for unconstrained optimization
Nminmax	number of lower and upper bounds
x0	starting values for initial conditions on state variables
parFlagX0	flags for free and fixed initial conditions = 1: free parameters (to be estimated) = 0: fixed parameters (not to be estimated)
NX0ID	total number of free (to be estimated) initial conditions
bXpar	starting values for bias parameters for each time segment
parFlagBX	flags for free and fixed bias parameters for each time segment = 1: free parameters (to be estimated) = 0: fixed parameters (not to be estimated)
NbX	total number of (fixed and free) bias parameters per time segment
NBXID	number of free (to be estimated) bias parameters
iArtifStab	flag for artificial stabilization (=0 for OEM)
StabMat	artificial stabilization matrix
LineSrch	flag for line search option in the Gauss–Newton method (=0, No line search, =1, Line search)

The variable names used in the program "ml_oem2" and calling functions are shown in the preceding list on the left-hand side, followed by the description and the notation used to denote these variables in the text. The option of artificial stabilization is not relevant for the present chapter, but it is required in Chapter 9,

which extends the basic OEM method to unstable aircraft. In all examples covered in this book, except for those in Chapter 9, the flag iArtifStab and the matrix StabMat are dummy variables, and we initialize them to zero and an empty matrix.

A couple of relevant indices such as: 1) total number of parameters to be estimated (free parameters) NparID, 2) the number of lower and upper bounds Nminmax, 3) total number of free initial conditions NX0ID, 4) total number of (fixed and free) bias parameters per time segment NbX, and 5) number of free (to be estimated) bias parameters NBXID can be readily computed from the size of the corresponding flags. The default values for lower and upper bounds on the parameters are $-\infty$ and ∞ respectively, implying unconstrained optimization. The flag LineSrch allows choosing between the two approaches of overcoming the intermediate divergence discussed in Secs. 4.10.1 and 4.10.2. Setting LineSrch equal to zero implies heuristic approach whereas setting it to one implies applying the line search option. The flag LineSrch is relevant only when the integer flag for optimization method iOptim is set to one, that is, when the Gauss–Newton method is to be applied.

It is obvious that model specification in terms of (Nx, Ny, Nu, Nparam) must match with those coded in the two user functions for the postulated model. It is also required to specify suitable starting values for the parameters Θ and the integer flags parFlag to indicate whether the particular parameter is free (to be estimated) or fixed (not to be estimated). Any arbitrary combination of free (to be estimated) and fixed system parameters is possible and can be suitably specified by parFlag. The same procedure of specifying stating values and the flags to keep them free or fixed is required for the initial conditions x_0 and the bias parameters bXpar. They are to be specified for n_E time segments being analyzed simultaneously. The corresponding flags parFlagX0 and parFlagBX can be specified in any arbitrary combination. This option to handle the three sets of parameters, as either unknowns or as known, or not identifiable from data analyzed, and hence to be kept fixed, makes the software flexible. For example, the function codes for the state and observation equations with larger number of parameters can be used without modifications to estimate a subset of them, depending upon the data available.

The flight data to be analyzed are also to be loaded in the function for model definition and require specification or assignment of the following information:

Ndata	number of data points
dt	sampling time
t	time vector [=0 to (Ndata-1)*dt]
Z(Ndata,Ny)	flight data for measured outputs (N, n_y)
Uinp(Ndata,Nu)	flight data for measured control inputs (N, n_u)

While loading the flight data in the arrays Z(Ndata,Ny) and Uinp(Ndata,Nu), it is obvious that the units of the variables must match with those used in the

postulated model and programmed in the state and observation functions "state_eq" and "obser_eq," respectively.

Because we provide an option to analyze multiple experiments, it is required to define additionally the following time-segment related information:

Nzi	number of time segments to be analyzed simultaneously (n_E)
izhf	cumulative index at which the maneuvers end when concatenated.

For example, if we analyze three time segments, each having Nts1, Nts2, and Nts3 data points, respectively, then izhf is defined as [Nts1; Nts1 + Nts2; Nts1 + Nts2 + Nts3]. The total number of data points, Ndata, is just the sum of data points for each of the Nzi segments. In this chapter, we will be dealing with test cases analyzing a single maneuver at a time, but the option of multiple experiments will be invoked in examples covered in other chapters.

From Fig. 4.6, it can be seen that the function "costfun_oem" integrates the states and computes the observations and the cost function value; the function "gradFG" propagates the perturbed state equations, computes the perturbed system responses, and computes the gradient vector $\mathcal{G}$ and information matrix $\mathcal{F}$; the parameter updates $\Delta\Theta$ are computed in the function "improv" when the Gauss–Newton method is applied; otherwise, the function "LMmethod" for the Levenberg–Marquardt method is called. Depending upon whether lower and upper bounds are specified or not, as determined by Nminmax, either the unconstrained Gauss–Newton method or the bounded-variable Gauss–Newton method is applied. We prefer to use Cholesky factorization to solve for $\Delta\Theta$; other methods, if desired by users, can be easily invoked in these functions. The convergence checking is performed in "ml_oem2," so also the step-size control including halving of the parameter steps to overcome intermediate divergence. At the end, we compute the standard deviations and correlation coefficients in "par_accuracy" and finally make plots of the time histories and of estimates.

For each test case, a program is provided to plot time histories of the control variables and generate superimposed plots of measured and computed output variables that allow direct comparison. In a separate figure, the estimates plotted against iteration count show the convergence of the optimization method. The vertical bars are provided, showing the standard deviations. The plot programs are called "plots_TCnn_oem_ttt.m," where nn refers to the test case index and ttt to additional explanation usually giving details of model or aircraft analyzed.

4.20 EXAMPLES

Although we will cover some typical examples related to flight vehicles in a separate chapter, we consider here two simple examples to demonstrate the output-error method and the use of the software provided. The first example

pertains to analysis of flight data recorded during an aileron and rudder input maneuver, performed under seemingly steady atmospheric conditions. To this example, we will apply the Gauss–Newton, bounded-variable Gauss–Newton, and Levenberg–Marquardt methods. The second example also pertains to the lateral-directional motion, analyzing simulated data that have been generated with a moderate to high level of atmospheric turbulence. The test cases are selected through the flag "test_case" and run using the program "ml_oem2." The flight data analyzed in these cases are supplied to the readers as sample flight data for verification and to try out other options and possibly to verify software that they might modify or develop themselves. These data files are found under the directory /FVSysID2/flt_data/. More complex test cases are provided in Chapter 10, which provide details of analyzing multiple time segments and estimating initial conditions and bias parameters for each time segment separately.

4.20.1 LATERAL-DIRECTIONAL MOTION

The model pertaining to the lateral-directional motion is postulated as follows:

State equations:

$$\begin{aligned} \dot{p} &= L_p p + L_r r + L_{\delta a}\delta_a + L_{\delta r}\delta_r + L_\beta \beta + b_{x\dot{p}} \\ \dot{r} &= N_p p + N_r r + N_{\delta a}\delta_a + N_{\delta r}\delta_r + N_\beta \beta + b_{x\dot{r}} \end{aligned} \tag{4.90}$$

Observation equations:

$$\begin{aligned} \dot{p}_m &= L_p p + L_r r + L_{\delta a}\delta_a + L_{\delta r}\delta_r + L_\beta \beta + b_{y\dot{p}} \\ \dot{r}_m &= N_p p + N_r r + N_{\delta a}\delta_a + N_{\delta r}\delta_r + N_\beta \beta + b_{y\dot{r}} \\ a_{ym} &= Y_p p + Y_r r + Y_{\delta a}\delta_a + Y_{\delta r}\delta_r + Y_\beta \beta + b_{yay} \\ p_m &= p + b_{yp} \\ r_m &= r + b_{yr} \end{aligned} \tag{4.91}$$

where the subscript m denotes the measured variables; p the roll rate; r the yaw rate; a_y the lateral acceleration; $\dot{p}$ the roll acceleration; $\dot{r}$ the yaw acceleration; δ_a the aileron deflection; δ_r the rudder deflection; β the angle of sideslip; and $L_{()}$, $N_{()}$, and $Y_{()}$ the dimensional aerodynamic derivatives. As discussed in Sec. 3.5.2, the postulated model contains bias terms for the state and observation equations $b_{x()}$ and $b_{y()}$, respectively. The readers might recall that the lumped bias parameters provide a convenient means to appropriately account for the initial conditions and systematic errors in the measurements. Besides the aileron and rudder inputs, as discussed in Sec. 3.6, the angle of sideslip β is treated as a pseudo control input. The unknown parameter vector Θ consisting of the

dimensional derivatives and the bias parameters is given by

$$\Theta^T = [L_p \;\; L_r \;\; L_{\delta a} \;\; L_{\delta r} \;\; L_\beta \;\; N_p \;\; N_r \;\; N_{\delta a} \;\; N_{\delta r} \;\; N_\beta \;\; Y_p \;\; Y_r \;\; Y_{\delta a} \;\; Y_{\delta r} \;\; Y_\beta \;\; b_{x\dot{p}} \;\; b_{x\dot{r}} \;\; b_{y\dot{p}} \;\; b_{y\dot{r}} \;\; b_{yay} \;\; b_{yp} \;\; b_{yr}] \tag{4.92}$$

Thus, the postulated model is defined as follows:

	No. of Variables		Function Name
States	2	p, r	"xdot_TC01_attas_lat"
Outputs	5	$\dot{p}$, $\dot{r}$, a_y, p, r	"obs_TC01_attas_lat"
Inputs	3	δ_a, δ_r, β	

The right-hand sides of the state equation, Eq. (4.90), are programmed in the function "xdot_TC01_attas_lat" and those of the observation equation, Eq. (4.91), in "obs_TC01_attas_lat." Thus, the model definition for this example provided by the function "/FVSysID2/chapter04/mDefCase01.m," which is called from the main program "/FVSysID2/chapter04/ml_oem2" is as follows:

```
test_case = 1;                          % index for the test case
state_eq  = "xdot_TC01_attas_lat";      % function for state equations
obser_eq  = "obs_TC01_attas_lat";       % function for observation equations
Nx        = 2;                          % number of states
Ny        = 5;                          % number of observation variables
Nu        = 3;                          % number of input (control) variables
NparSys   = 15;                         % number of system parameters
Nparam    = NparSys + Nx + Ny;          % total number of parameters
dt        = 0.04;                       % sampling time
```

The flight data analyzed were gathered from flight tests carried out with the research aircraft ATTAS [63]. The rolling and yawing motions were excited in a sequence through pilot-applied aileron and rudder inputs. The flight maneuvers were carried out under seemingly steady atmospheric conditions when the turbulence was considered negligible. The data were recorded with a sampling time of 40 ms, and the duration of the maneuver is 60 s. The resulting lateral-directional motion was recorded and analyzed as a single maneuver. The data to be analyzed are loaded from the data file "/FVSysID2/flt_data/fAttasAilRud1.mat." The number of time segments is given by Nzi = 1, and the izhf is set to the total number of data points. The output and input variables in data arrays Z(Ndata,Ny) and Uinp(Ndata,Nu) are appropriately assigned; a list of channels recorded is provided in the README. Suitable starting values are specified for the unknown parameters; they are mostly 50% off the expected values. The integer flags parFlag are set to one, indicating that all parameters are to be estimated. The lower and upper bounds are set to $-\infty$ and ∞

respectively, leading to unconstrained optimization. All of these details can be easily traced from the model definition function "mDefCase01.m." In the main program, the flag integMethod is set to ruku4 for the fourth-order Runge–Kutta integration method.

4.20.1.1 GAUSS–NEWTON AND LEVENBERG–MARQUARDT METHODS

The results of parameter estimation applying the output-error method are summarized in Table 4.1, showing convergence obtained from the Gauss–Newton and Levenberg–Marquardt methods. By setting the flag iOptim = 1 (in ml_oem2.m) and LineSrch = 0 (in mDefCase01.m), we choose first the Gauss–Newton optimization and heuristic approach to cater to intermediate divergence. It is observed that the Gauss–Newton method shows intermediate divergence during the first iteration. As pointed out in Sec. 4.10.1, the heuristic approach of halving the parameter update proved quite sufficient to reduce the cost function through a single step. The subsequent convergence was smooth, and the process was terminated when the relative change in the cost function, that is, determinant of the covariance matrix R was less than tolR = 0.0001. The convergence is achieved in six iterations, and the cost function values are provided in the second column of Table 4.1.

In the second attempt, we continue applying the Gauss–Newton method, but choose the line search strategy to correct local intermediate divergence (selected by setting the integer flag LineSrch = 1 in mDefCase01.m). We start with the same initial parameter values and the cost function. It is observed that the line

TABLE 4.1 CONVERGENCE OF GAUSS–NEWTON WITH HEURISTIC APPROACH AND LINE SEARCH AND LEVENBERG–MARQUARDT METHODS FOR TEST CASE 1

Iteration	Gauss–Newton: Cost Function	Gauss–Newton with Line Search: Cost Function	Levenberg–Marquardt: Cost Function (LM Parameter)
0	1.3568×10^{-11}	1.3568×10^{-11}	1.3568×10^{-11}
1	3.6039×10^{-10} 6.7975×10^{-15}	2.2964×10^{-15}	3.5663×10^{-10} (λ: 1.0×10^{-2}) 3.2505×10^{-10} (λ: 1.0×10^{-1}) 4.0265×10^{-13} (λ: 1.0×10^{-1})
2	2.437×10^{-19}	2.1453×10^{-19}	8.5981×10^{-18} (λ: 1.0×10^{-2})
3	9.3597×10^{-20}	9.3548×10^{-20}	1.6038×10^{-19} (λ: 1.0×10^{-3})
4	9.1717×10^{-20}	9.1706×10^{-20}	9.2428×10^{-20} (λ: 1.0×10^{-4})
5	9.1702×10^{-20}	9.1702×10^{-20}	9.1707×10^{-20} (λ: 1.0×10^{-5})
6	9.1702×10^{-20}		9.1702×10^{-20} (λ: 1.0×10^{-6})

search option automatically overcomes the intermediate divergence during the first iteration. For this simple case the convergence is achieved in five iterations. The cost function values during the optimization in this case are tabulated in the third column of Table 4.1.

In the case of the Levenberg–Marquardt method (selected by setting the integer flag iOptim = 2 in ml_oem2.m), we start once again with the same initial parameter values and cost function and the default value for the LM parameter $\lambda = 0.001$, as suggested in Sec. 4.13. During the first iteration, intermediate divergence is encountered; at this stage the algorithm adapts the LM parameter λ until a reduction in the cost function is achieved. In the present case, this required three steps, changing λ from 0.001 to 0.01; note that during the third step of LM-parameter adjustment, λ is not reduced as a different condition is satisfied here; see Sec. 4.13. The λ values are shown in brackets in the fourth column of Table 4.1. During the subsequent iterations, it turns out that the Gauss–Newton directions are the best choice and the optimization progresses smoothly; hence, λ is reduced continuously from iteration to iteration.

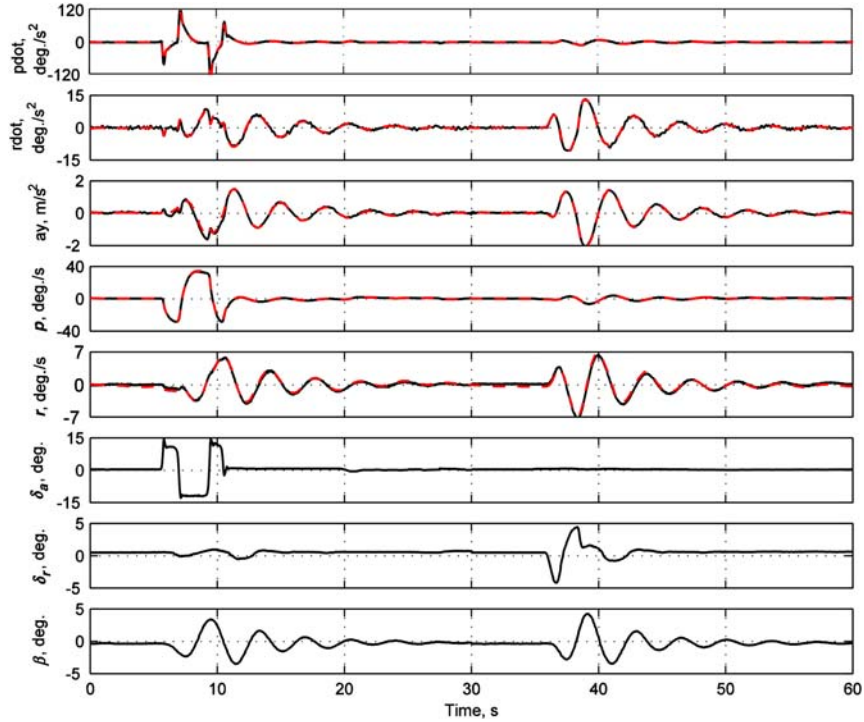

Fig. 4.7 Time histories of control inputs and a comparison of flight measured and model predicted outputs for test case 1: ——, flight measured; -----, model predicted.

The time history match is shown in Fig. 4.7 only once because there was no qualitative difference between the three methods. The three plots from the bottom show the control inputs, namely, the angle of sideslip and the rudder and aileron deflections, respectively. The five plots from the top show the match between the flight measured responses (shown in continuous lines) and the model estimated responses (shown in dashed). The match between the two is excellent. The computations in the postulated model are carried out in radians and radians/s whereas the time histories are plotted in degrees and degrees/s.

It is observed that the Gauss–Newton method with line search option requires one iteration less for convergence. But in general, for more complex cases, it is not necessary that the Gauss–Newton method with line search option will perform better, requiring less iterations for convergence. In a few cases, we have also observed that the line search option requires the same number of iterations and rarely one or two iterations more. Thus, the choice of heuristic approach or line search is not quite obvious. In most of the cases, we have been using the heuristic approach, which is found to be quite adequate, even for much more complex problems. Convergence applying the Levenberg–Marquardt method is achieved in the same number of iterations as that by the Gauss–Newton method with heuristic approach for intermediate divergence, which is more or less a coincidence. Depending upon the starting values and the nature of the cost function, the two methods may progress towards the minimum differently and may require different number of iterations.

The convergence plots for the derivatives $L_{()}$ and $N_{()}$ obtained from the Gauss–Newton method are provided in Fig 4.8, showing the error bars, that is, standard deviations, given by Eq. (4.88). It is observed that all of the parameters are estimated with very low standard deviations and there was no correlation between the aerodynamic derivatives. Numerical values and the standard deviations are not provided here, but it can be easily verified that the three methods yield the same results within the numerical accuracy and roundoff errors.

4.20.1.2 BOUNDED-VARIABLE GAUSS–NEWTON METHOD

We continue with the same example to demonstrate the use of bounded-variable Gauss–Newton method discussed in Sec. 4.11. For this purpose we arbitrarily define lower and upper bounds on two parameters appearing in the model postulated in Eqs. (4.90) and (4.91); they are $(-6.0, -3.0)$ for the third parameter $L_{\delta a}$ and $(-5.0, -3.5)$ for the fifth parameter L_{β}. These limits serve the demonstration purpose only. Accordingly, in "mDefCase01.m," we set param_min(3) $= -6.0$, param_max(3) $= -3.0$, and param_min(5) $= -5.0$, param_max(5) $= -3.5$. The starting values of parameters for which lower and upper bounds are specified should lie within the range defined by the bounds. The rest of the settings in model definition function remain unchanged. Now, we rerun the case by starting ml_oem2.m. It is observed that starting from the same cost function value

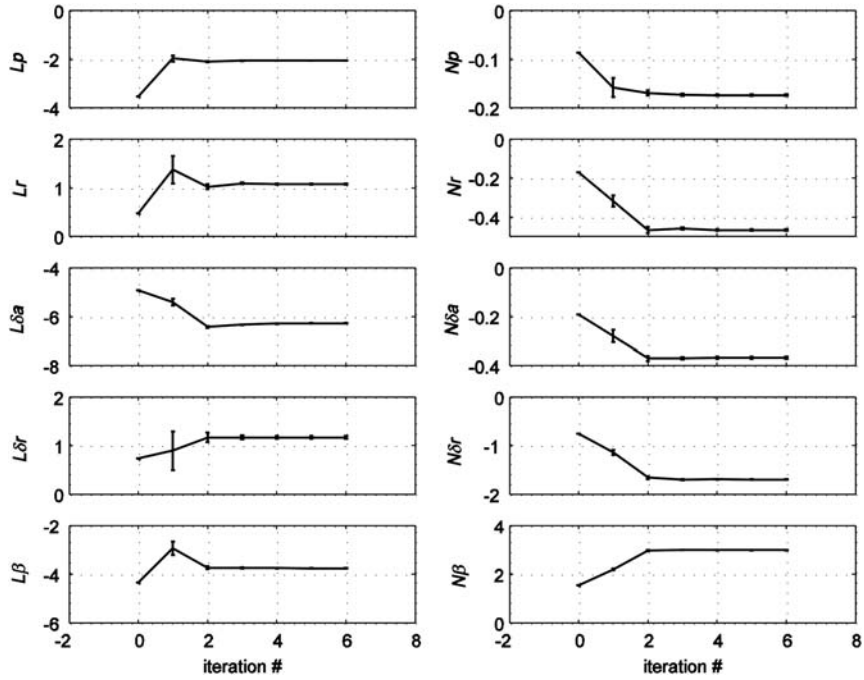

Fig. 4.8 Convergence plot of dimensional derivatives estimated from flight data for test case 1.

1.3568×10^{-11}, as in the preceding cases, the convergence is achieved within five iterations. The optimized cost function in this case is 1.3452×10^{-19}. At the end of the optimization, the log of final results shows the estimates of free parameters, their standard deviations, and the lower and upper bounds, if any are specified. If any parameter remains at any one of the bounds, this is indicated by "*" after the estimate in the printout. It helps to pinpoint that the results are obtained though constrained optimization. For parameters remaining at its bounds, one cannot compute the standard deviations.

The convergence plot of the various parameters is shown in Fig. 4.9. The lower and upper bounds are marked for the two parameters (see third and fifth plot on the left-hand side of Fig. 4.9). It is observed that parameter L_β hits the upper bound of -3.5 during the first iteration, but gets freed during the subsequent iterations. On the other hand, the parameter $L_{\delta a}$ hits the lower bound of -6.0 during the second iteration and remains at its bound until the convergence. As discussed in Sec. 4.11, the algorithm automatically caters to either keep the parameters fixed at its bounds or free them as the optimization progresses.

Comparing the results of optimization with those given in Table 4.1, we notice that the cost function value (1.3452×10^{-19}) in this case is somewhat larger than

that obtained through unconstrained optimization (9.1702×10^{-20}). This behavior is typical of the constrained optimization. Any further reduction in the cost function would be possible only when the particular parameters are not constrained. Further, the number of iterations required for the convergence in the present case is the same as that for the unconstrained Gauss–Newton method with line search (third column, Table 4.1). In a couple of other more complex cases too, it was observed that the bounded-variable method required the same number of iterations. In a few problems, however, the constrained optimization by bounded-variable Gauss–Newton method required one or two additional iterations, particularly when the bounds were inappropriate. Thus, in general, some care is necessary while specifying the parameter bounds. In some cases cycling of active set, that is, parameters entering the active set on reaching the bounds followed immediately by an indication to leave the active set, can result, particularly as the minimum of the cost function is approached. The reason for this is twofold: 1) with the roundoff errors becoming dominant, the gradients computed using the numerical approximation are not sufficiently accurate; and 2) some parameters appearing in the model are approximately linearly dependent. This

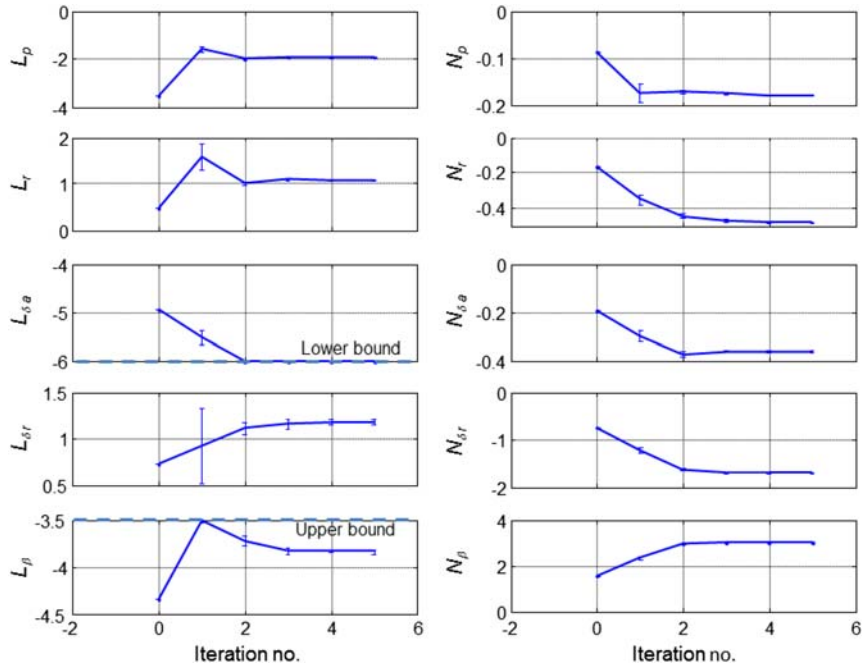

Fig. 4.9 Convergence plot of dimensional derivatives estimated applying bounded-variable Gauss–Newton method for test case 1.

phenomenon was, however, not encountered in several examples of estimating nonlinear aerodynamic parameters with various degrees of complexity.

4.20.1.3 ESTIMATION OF TIME DELAY

To demonstrate the use of software to estimate time delay, we continue with the same example discussed hitherto. The model postulate for this case remains the same as in Eqs. (4.90–4.91). For demonstration purposes, we choose to estimate a time delay in the measured yaw rate. We define an additional parameter called τ_r, the time delay in yaw rate r, and append the parameter vector Θ in Eq. (4.92) with τ_r. We call this test case as test_case = 31. Thus, now the number of system parameters NparSys is set to 16 in the model definition mDefCase_TC31.m (instead of 15 in the first case). Because we wish to account for the time delay in the measured data, shifting of the integrated variable is to be carried out in the function for the observation variables, in the present case "obs_TC31_attas_lat.m." Herein, we call the utility function "timeDelay.m" to shift the computed yaw rate r, which is based on the delay matrix discussed in Sec. 3.4 as follows:

$$[rTz, xWS, tNewX, iNewX] = timeDelay(rTz, tDelay\_r, xWS, tNewX, iNewX, nTdMx)$$

where rTz is the computed yaw rate to be time shifted by a parameter $tDelay\_r$ being estimated and xWS, $tNewX$, $iNewX$ are the work space arrays associated with time delay. We assume that the maximum time delay is 40 times the sampling time and accordingly define $nTDMX = 40$. If this is not sufficient, the computations are terminated in the function timeDelay, with a warning message for the analyst. The exact details of calling the time delay function discussed in Sec. 3.4 can be traced in the function for the observations "obs_TC31_attas_lat.m." The time-delayed computed yaw rate is then matched with the measured data.

The state equations for test_case = 31 are provided in "xdot_TC31_attas_lat.m," which are exactly the same as those in "xdot_TC01_attas_lat.m" for test_case = 1. The observation equations in "obs_TC31_attas_lat.m" differ only slightly compared with "obs_TC01_attas_lat.m"; it includes changes to account for the call of the function "timeDelay.m." A major part of the model definition mDefCase_TC33.m remains the same, except for increasing number of system parameters NparSys by one, and setting starting value for τ_r to zero in the parameter vector param and setting the corresponding flag parFlag to one. Now choosing the index Test_case = 31 in the main program ml_oem2, we run this case. Because we start with the same values for the aerodynamic derivatives and the starting value for time delay is set to zero, the starting cost function values of 1.3568×10^{-11} is the same as before found in Table 4.1. We apply the Gauss–Newton method and note that convergence is very similar to that

observed for the first example (test_case = 1), once again encountering intermediate divergence in the first iteration, which is corrected by the heuristic approach. The convergence is achieved in the same number of iterations. The cots function value is 9.0444×10^{-20}, which is slightly lower than 9.1702×10^{-20} for the first case. This is typical of parameter estimation; the more the number of parameters, the lower is the cost function and thereby better is the fit.

In this test case, it is also possible to set the time delay to zero and keep it fixed by setting the corresponding parFlag to zero in the model definition function "mDefCase_TC31.m." This yields exactly the same results given in Table 3.1 and nicely confirms that the procedure adopted to generate time delays does not affect the algorithm provided. The preceding simple example serves the only purpose of demonstrating estimation of time delay. In the present case, only one set of the global work spaces and indices *xWS*, *tNewX*, *iNewX*, *nTdMx* is necessary because we considered time delay in just one signal. However, if more than one signal is to be time shifted, recall from Sec. 3.4 that the calling procedure would be similar for each variable, and the work space variables have to be defined uniquely for each of them. A little more complex problem based on multiple time segments is presented later in Chapter 10.

4.20.2 APPLICATION OF OUTPUT-ERROR METHOD TO DATA WITH PROCESS NOISE

To evaluate the performance of more advanced estimation algorithms accounting for process noise that we will be addressing in the next chapter, aircraft responses pertaining to lateral-directional motion with an appreciable level of turbulence are generated through simulation [64]. The nominal values of the aerodynamic derivatives used correspond to those obtained from flight tests carried out in steady atmospheric conditions with the research aircraft de Havilland DHC-2. Equations (4.90) and (4.91), incorporating additional state and measurement noise, are used to generate the data. The angle of sideslip has been replaced through the lateral-velocity component v and the corresponding derivatives too. To provide realistic control inputs, the rudder and aileron excitations actually applied in a particular flight test are used to excite the postulated model. Independent process and measurement noise vectors are generated using pseudorandom noise generators. The state noise matrix was assumed to be diagonal. Note that state variables p and r only affected by the state noise and not the other control inputs δ_a, δ_r, and v. A total of 16 s of data with a sampling time of 0.05 s has been generated. These simulated responses are analyzed applying parameter estimation methods; see Fig. 4.10. Time histories of these data are provided as /FVSysID2/flt_data/y13aus_da1.asc.

To these data with process noise, we apply the output-error method. We use the model postulated in Eqs. (4.90) and (4.91) to estimate the unknown parameter vector of Eq. (4.91), except that, as already pointed out, the pseudo control

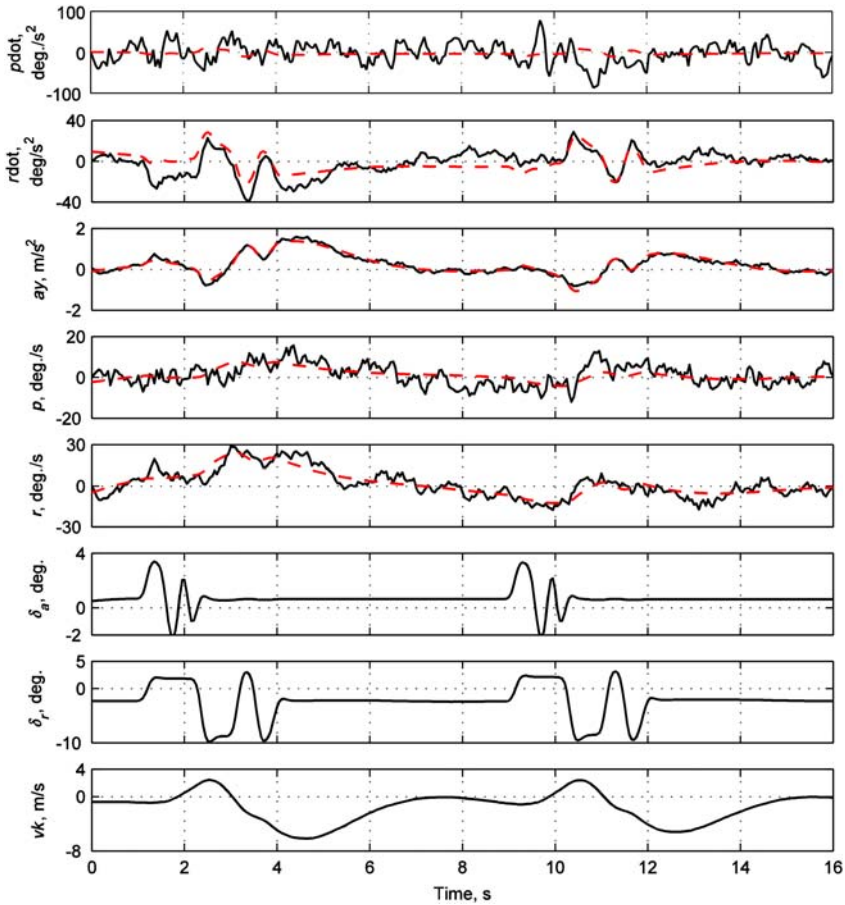

Fig. 4.10 Output-error method applied to data with moderate to high level of turbulence — test case 2: ——, flight measured; -----, model predicted.

input β has been replaced through v. This test case is designated as test_case = 2. The model definition is very similar to that of test_case = 1. The data to be analyzed, that is, the arrays Z(Ndata,Ny) and Uinp(Ndata,Nu), are loaded from the file /FVSysID2/flt_data/y13aus_da1.asc. The model definition for this example is provided by the function "/FVSysID2/chapter04/mDefCase02.m," which is called from the main program "/FVSysID2/chapter04/ml_oem2" for test_case = 2. To apply the Gauss–Newton, we set the integer flag iOptim = 1.

In this case, it is observed that the output-error method has severe convergence problems. Almost at every iteration, intermediate divergence is encountered. The process terminates after 42 iterations, yielding parameter values

that are not anywhere near the expected values. The estimated parameters are not provided here because we will get back to this case in Chapter 5, when we apply the filter-error method and make a comparison with the output-error method. The time history plots are provided in Fig. 4.10, showing the control inputs and the match for the output variables. The discrepancies in the match are clearly evident. Although this example is a simple one, it helps to demonstrate the limitations of the output-error method when we encounter process noise, that is, for flight data gathered under turbulent atmospheric conditions.

4.21 CONCLUDING REMARKS

In this chapter we presented the general principle of maximum likelihood estimation, highlighting the exact physical interpretation, and showed that it can be applied to nonlinear dynamical systems of arbitrary complexity. It was shown that the maximum likelihood estimator has several desirable statistical properties, namely, unbiased estimates, asymptotic consistency, normality, and efficiency. The advantages of the method are that it makes optimal use of the measured data and as a byproduct provides, without extra computations, statistical information on the accuracy of the parameter estimates in terms of the standard deviations and correlation coefficients. The subtle issues associated with evaluation of these statistical accuracies are deferred to Chapter 11 dealing with model validation.

The output-error method is one of the most widely applied algorithms to estimate aircraft parameters. It considers measurement noise only, but this assumption simplifies the algorithm significantly. It is also considered acceptable from a practical viewpoint because we generally carry out flight tests specifically for parameter estimation purposes, and as such these flights can be carried out as well when the atmospheric conditions are stable. The relaxation strategy provides the best approach to optimize the likelihood function and overcomes convergence problems through the closed-form solution for the estimation of the measurement noise covariance matrix. We studied in some depth the three most commonly applied optimization methods, namely, unconstrained Gauss–Newton, bounded-variable Gauss–Newton, and Levenberg–Marquardt. Various options to overcome the problems of intermediate divergence in the unconstrained Gauss–Newton have been presented; these finer practical details are very important to enlarge the scope of application of these methods. Several other numerical aspects, like approximation of sensitivity coefficients, nonlinear model structures and structure of the sensitivity matrix for large-scale systems were discussed.

We also covered briefly the aspects of Bayesian estimation, the most general form of an estimator, and showed that a priori information about the aerodynamic derivatives can be accounted for in the output-error method. We further

argued that due to uncertainties in the error covariance matrix for such a priori values, the approach is rarely used in practice.

Finally, samples of flight data and extended software implemented under Matlab® have been provided to follow the various computational steps of the algorithm. It has been applied to the two examples that were presented to demonstrate the use of the software and to evaluate the performance of the output-error method. The extended software provides options to estimate system parameters and initial conditions and bias parameters of multiple time segments being analyzed simultaneously, thereby extending the scope of the application significantly and even to more complex problems.

REFERENCES

[1] Fisher R. A., "On an Absolute Criterion for Fitting Frequency Curves," *Messenger of Mathematics*, Vol. 41, Macmillan, London, 1912, pp. 155–160.

[2] Fisher R. A., "On the Mathematical Foundations of Theoretical Statistics," *Philosophical Transactions of the Royal Society of London*, Vol. 222, 1922, pp. 309–368.

[3] Fisher R. A., "Theory of Statistical Estimations," *Proceedings of the Cambridge Philosophical Society*, Vol. 22, 1925, pp. 700–725.

[4] Iliff K. W., "Parameter Estimation for Flight Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 5, 1989, pp. 609–622.

[5] Klein V., "Estimation of Aircraft Aerodynamic Parameters from Flight Data," *Progress in Aerospace Sciences*, Vol. 26, No. 1, 1989, pp. 1–77.

[6] Hamel P. G., and Jategaonkar R. V., "Evolution of Flight Vehicle System Identification," *Journal of Aircraft*, Vol. 33, No. 1, 1996, pp. 9–28.

[7] Jategaonkar R. V., Fischenberg D., and von Gruenhagen W., "Aerodynamic Modeling and System Identification from Flight Data - Recent Applications at DLR," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 681–691.

[8] Wang K. C., and Iliff K. W., "Retrospective and Recent Examples of Aircraft Parameter Identification at NASA Dryden Flight Research Center," *Journal of Aircraft*, Vol. 41, No. 4, 2004, pp. 752–764.

[9] Morelli E. A., and Klein V., "Application of System Identification to Aircraft at NASA Langley Research Center," *Journal of Aircraft*, Vol. 42, No. 1, 2005, pp. 12–25.

[10] Kendall M. G., and Stuart A., *The Advanced Theory of Statistics*, 3rd ed., Griffin, London, 1969.

[11] Waerden B. L., *Mathematical Statistics*, Springer, New York, 1969.

[12] Sorenson H. W., *Parameter Estimation – Principles and Problems*, Marcel Dekker, New York, 1980.

[13] Davenport W. B., and Root W. L., *Random Signals and Noise*, McGraw–Hill, New York, 1958.

[14] Goodwin G. C., and Payne R. L., *Dynamic System Identification*, Academic Press, New York, 1977.

[15] Wald A., "Note on the Consistency of the Maximum Likelihood Estimates," *Annals of Mathematical Statistics*, Vol. 20, 1949, pp. 595–601.