最佳化方法與應用
MA5037-*

Chapter 10. Least-Squares Problems

Introduction

In least-squares problems, the objective function f takes the form

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x), \tag{1}$$

where each r_j is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$. We refer to each r_j as a residual, and we assume throughout this chapter that $m \geqslant n$.

Least-squares problems arise in many areas of applications. Many who formulate a parametrized model for real-world applications use a function of the form (1) to **measure the discrepancy (差異) between the model and the observed behavior** of the system. By minimizing this function, they select values for the parameters that best match the model to the data. In this chapter we show how to devise efficient, robust minimization algorithms by exploiting the special structure of the function f and its derivatives.

Introduction

In least-squares problems, the objective function f takes the form

$$f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j^2(x), \tag{1}$$

where each r_j is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$. We refer to each r_j as a residual, and we assume throughout this chapter that $m \geqslant n$.

Least-squares problems arise in many areas of applications. Many who formulate a parametrized model for real-world applications use a function of the form (1) to **measure the discrepancy (差異) between the model and the observed behavior** of the system. By minimizing this function, they select values for the parameters that best match the model to the data. In this chapter we show how to devise efficient, robust minimization algorithms by exploiting the special structure of the function f and its derivatives.

Introduction

In least-squares problems, the objective function f takes the form

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)\,, \tag{1}$$

where each r_j is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$. We refer to each r_j as a residual, and we assume throughout this chapter that $m \geqslant n$.

Least-squares problems arise in many areas of applications. Many who formulate a parametrized model for real-world applications use a function of the form (1) to **measure the discrepancy (差異) between the model and the observed behavior** of the system. By minimizing this function, they select values for the parameters that best match the model to the data. In this chapter we show how to devise efficient, robust minimization algorithms by exploiting the special structure of the function f and its derivatives.

Introduction

In least-squares problems, the objective function f takes the form

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x)\,, \tag{1}$$

where each r_j is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}$. We refer to each r_j as a residual, and we assume throughout this chapter that $m \geqslant n$.

Least-squares problems arise in many areas of applications. Many who formulate a parametrized model for real-world applications use a function of the form (1) to **measure the discrepancy (差異) between the model and the observed behavior** of the system. By minimizing this function, they select values for the parameters that best match the model to the data. In this chapter we show how to devise efficient, robust minimization algorithms by exploiting the special structure of the function f and its derivatives.

Introduction

By assembling the individual components r_j from (1) into a residual vector $r : \mathbb{R}^n \to \mathbb{R}^m$, as follows

$$r(x) = [r_1(x), r_2(x), \cdots, r_m(x)]^{\mathrm{T}}, \tag{2}$$

we can rewrite f as $f(x) = \frac{1}{2}\|r(x)\|_2^2$. The derivatives of $f(x)$ can be expressed in terms of the Jacobian matrix $J(x)$, which is the $m \times n$ matrix of first partial derivatives of the residuals, defined by

$$J(x) = \left[\nabla r(x)\right]_{m\times n} = \left[\frac{\partial r_j}{\partial x_i}\right]_{\substack{j=1,2,\cdots,m \\ i=1,2,\cdots,n}} = \begin{bmatrix} \nabla r_1(x)^{\mathrm{T}} \\ \nabla r_2(x)^{\mathrm{T}} \\ \vdots \\ \nabla r_m(x)^{\mathrm{T}} \end{bmatrix}, \tag{3}$$

where each $\nabla r_j(x)$, $j = 1, 2, \cdots, m$ is the gradient of r_j (represented by a column vector).

Introduction

The gradient and Hessian of f can then be expressed as follows:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^{\mathrm{T}} + \sum_{j=1}^{m} r_j(x) (\nabla^2 r_j)(x) \tag{5}$$

In many applications, the first partial derivatives of the residuals and hence the Jacobian matrix $J(x)$ are relatively easy or inexpensive to calculate. We can thus obtain the gradient $\nabla f(x)$ as written in formula (4). Using $J(x)$, we also can calculate the first term $J(x)^{\mathrm{T}} J(x)$ in the Hessian $\nabla^2 f(x)$ without evaluating any second derivatives of the functions r_j. This availability of part of $\nabla^2 f(x)$ "for free" is the distinctive feature of least-squares problems.

Introduction

The gradient and Hessian of f can then be expressed as follows:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x)\,. \tag{5}$$

In many applications, the first partial derivatives of the residuals and hence the Jacobian matrix $J(x)$ are relatively easy or inexpensive to calculate. We can thus obtain the gradient $\nabla f(x)$ as written in formula (4). Using $J(x)$, we also can calculate the first term $J(x)^{\mathrm{T}} J(x)$ in the Hessian $\nabla^2 f(x)$ without evaluating any second derivatives of the functions r_j. This availability of part of $\nabla^2 f(x)$ "for free" is the distinctive feature of least-squares problems.

Introduction

The gradient and Hessian of f can then be expressed as follows:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x) (\nabla^2 r_j)(x)\,. \tag{5}$$

In many applications, the first partial derivatives of the residuals and hence the Jacobian matrix $J(x)$ are relatively easy or inexpensive to calculate. We can thus obtain the gradient $\nabla f(x)$ as written in formula (4). Using $J(x)$, we also can calculate the first term $J(x)^{\mathrm{T}} J(x)$ in the Hessian $\nabla^2 f(x)$ without evaluating any second derivatives of the functions r_j. This availability of part of $\nabla^2 f(x)$ "for free" is the distinctive feature of least-squares problems.

Introduction

The gradient and Hessian of f can then be expressed as follows:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x) (\nabla^2 r_j)(x)\,. \tag{5}$$

In many applications, the first partial derivatives of the residuals and hence the Jacobian matrix $J(x)$ are relatively easy or inexpensive to calculate. We can thus obtain the gradient $\nabla f(x)$ as written in formula (4). Using $J(x)$, we also can calculate the first term $J(x)^{\mathrm{T}} J(x)$ in the Hessian $\nabla^2 f(x)$ without evaluating any second derivatives of the functions r_j. This availability of part of $\nabla^2 f(x)$ "for free" is the distinctive feature of least-squares problems.

Introduction

The gradient and Hessian of f can then be expressed as follows:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x)\,. \tag{5}$$

Moreover, this term $J(x)^{\mathrm{T}} J(x)$ is often more important than the second summation term in (5), either because the residuals r_j are close to affine near the solution (that is, the $\nabla^2 r_j(x)$ are relatively small) or because of small residuals (that is, the $r_j(x)$ are relatively small). Most algorithms for nonlinear least-squares exploit these structural properties of the Hessian.

Introduction

The most popular algorithms for minimizing (1) fit into the line search and trust-region frameworks described in earlier chapters. They are based on the Newton and quasi-Newton approaches described earlier, with modifications that exploit the particular structure of f.

Section 10.1 contains some background on applications. Section 10.2 discusses linear least-squares problems, which provide important motivation for algorithms for the nonlinear problem. Section 10.3 describes the major algorithms, while Section 10.4 briefly describes a variant of least squares known as orthogonal distance regression. Throughout this chapter, we use the notation $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$, unless a subscript indicates that some other norm is intended.

Introduction

The most popular algorithms for minimizing (1) fit into the line search and trust-region frameworks described in earlier chapters. They are based on the Newton and quasi-Newton approaches described earlier, with modifications that exploit the particular structure of f.

Section 10.1 contains some background on applications. Section 10.2 discusses linear least-squares problems, which provide important motivation for algorithms for the nonlinear problem. Section 10.3 describes the major algorithms, while Section 10.4 briefly describes a variant of least squares known as orthogonal distance regression. Throughout this chapter, we use the notation $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$, unless a subscript indicates that some other norm is intended.

§10.1 Background

We discuss a simple parametrized model and show how least-squares techniques can be used to choose the parameters that best fit the model to the observed data.

Example

We want to study the effect of a certain medication on a patient. We draw blood samples at certain times after the patient takes a dose, and measure the concentration of the medication in each sample, tabulating the time t_j and concentration y_j for each sample.

Based on our previous experience in such experiments, we find that the following function $\varphi(x;t)$ provides a good prediction of the concentration at time t, for appropriate values of the five-dimensional parameter vector $x \equiv (x_1, x_2, x_3, x_4, x_5)$:

$$\varphi(x;t) = x_1 + t x_2 + t^2 x_3 + x_4 e^{-x_5 t}. \tag{6}$$

§10.1 Background

We discuss a simple parametrized model and show how least-squares techniques can be used to choose the parameters that best fit the model to the observed data.

Example

We want to study the effect of a certain medication on a patient. We draw blood samples at certain times after the patient takes a dose, and measure the concentration of the medication in each sample, tabulating the time t_j and concentration y_j for each sample.

Based on our previous experience in such experiments, we find that the following function $\varphi(x;t)$ provides a good prediction of the concentration at time t, for appropriate values of the five-dimensional parameter vector $x \equiv (x_1, x_2, x_3, x_4, x_5)$:

$$\varphi(x;t) = x_1 + t x_2 + t^2 x_3 + x_4 e^{-x_5 t}. \tag{6}$$

§10.1 Background

Example (cont'd)

We choose the parameter vector x so that this model best agrees with our observation, in some sense. A good way to measure the difference between the predicted model values and the observations is the following least-squares function:

$$\frac{1}{2}\sum_{j=1}^{m}\left[\varphi(x;t_j)-y_j\right]^2, \tag{7}$$

which sums the squares of the discrepancies between predictions and observations at each t_j. This function has precisely the form (1) if we define

$$r_j(x)=\varphi(x;t_j)-y_j\,. \tag{8}$$

Graphically, each term in (7) represents the square of the vertical distance between the curve $\varphi(x;t)$ (plotted as a function of t) and the point (t_j,y_j), for a fixed choice of parameter x; see Figure 1.

§10.1 Background

Example (cont'd)

We choose the parameter vector x so that this model best agrees with our observation, in some sense. A good way to measure the difference between the predicted model values and the observations is the following least-squares function:

$$\frac{1}{2}\sum_{j=1}^{m}\left[\varphi(x;t_j)-y_j\right]^2, \tag{7}$$

which sums the squares of the discrepancies between predictions and observations at each t_j. This function has precisely the form (1) if we define

$$r_j(x)=\varphi(x;t_j)-y_j\,. \tag{8}$$

Graphically, each term in (7) represents the square of the vertical distance between the curve $\varphi(x;t)$ (plotted as a function of t) and the point (t_j,y_j), for a fixed choice of parameter x; see Figure 1.

§10.1 Background

Example (cont'd)

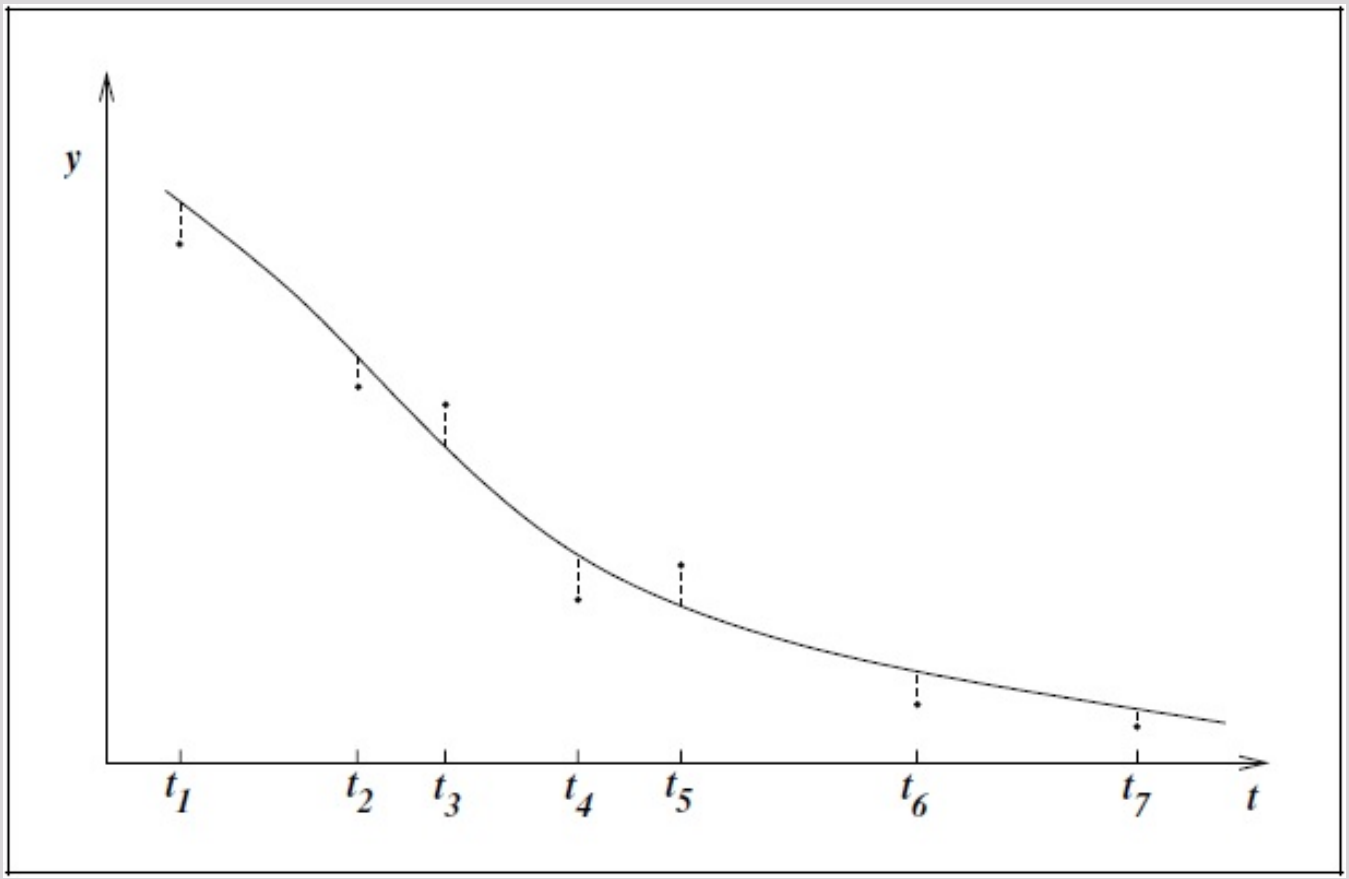

Figure 1: Model (7) (smooth curve) and the observed measurements, with deviations indicated by vertical dotted lines.

§10.1 Background

Example (cont'd)

The minimizer x_* of the least-squares problem is the parameter vector for which the sum of squares of the lengths of the dotted lines in Figure 1 is minimized. Having obtained x_*, we use $\varphi(x_*; t)$ to estimate the concentration of medication remaining in the patient's bloodstream at any time t.

This model is an example of what statisticians call a **fixed-regressor model**. It assumes that the times t_j at which the blood samples are drawn are known to high accuracy, while the observations y_j may contain more or less random errors due to the limitations of the equipment (or the lab technician!)

§10.1 Background

Example (cont'd)

The minimizer x_* of the least-squares problem is the parameter vector for which the sum of squares of the lengths of the dotted lines in Figure 1 is minimized. Having obtained x_*, we use $\varphi(x_*; t)$ to estimate the concentration of medication remaining in the patient's bloodstream at any time t.

This model is an example of what statisticians call a **fixed-regressor model**. It assumes that the times t_j at which the blood samples are drawn are known to high accuracy, while the observations y_j may contain more or less random errors due to the limitations of the equipment (or the lab technician!)

§10.1 Background

In general data-fitting problems of the type just described, the ordinate t in the model $\varphi(x; t)$ could be a vector instead of a scalar. In the example above, for instance, t could have two dimensions, with the first dimension representing the time since the drug was admistered and the second dimension representing the weight of the patient. We could then use observations for an entire population of patients, not just a single patient, to obtain the "best" parameters for this model.

§10.1 Background

The sum-of-squares function (7) is not the only way of measuring the discrepancy between the model and the observations. Other common measures include the maximum absolute value

$$\max_{1 \leqslant j \leqslant m} \left| \varphi(x; t_j) - y_j \right| \tag{9}$$

and the sum of absolute values

$$\sum_{j=1}^{m} \left| \varphi(x; t_j) - y_j \right| . \tag{10}$$

By using the definitions of the ℓ_∞ and ℓ_1 norms, we can rewrite these two measures as

$$f(x) = \| r(x) \|_\infty , \quad f(x) = \| r(x) \|_1 , \tag{11}$$

respectively. As we discuss in Chapter 17, the problem of minimizing the functions (11) can be reformulated a smooth constrained optimization problem.

§10.1 Background

In this chapter we focus only on the ℓ_2-norm formulation (1). In some situations, there are statistical motivations for choosing the least-squares criterion. Changing the notation slightly, we let the discrepancies between model and observation be denoted by ε_j:

$$\varepsilon_j = \varphi(x; t_j) - y_j \,.$$

It often is reasonable to assume that the ε_j's are i.i.d. with a certain variance σ^2 and probability density function $g_\sigma(\cdot)$. Under this assumption, the likelihood of a particular set of observations y_j, $j = 1, 2, \cdots, m$, given that the actual parameter vector is x, is given by

$$p(y; x, \sigma) = \prod_{j=1}^{m} g_\sigma(\varepsilon_j) = \prod_{j=1}^{m} g_\sigma(\varphi(x; t_j) - y_j) \,. \tag{12}$$

Given the observations $y_1, y_2, \cdots, y_m$, the "most likely" value of x is obtained by maximizing $p(y; x, \sigma)$ with respect to x. The resulting value of x is called the maximum likelihood estimate.

§10.1 Background

Assume that the discrepancies follow a normal distribution. Then

$$g_\sigma(\varepsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right).$$

Substitution in (12) yields

$$p(y; x, \sigma) = (2\pi\sigma^2)^{-m/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{j=1}^{m} \left[\varphi(x; t_j) - y_j\right]^2\right).$$

For any fixed value of the variance σ^2, it is obvious that p is maximized when the sum of squares (7) is minimized. To summarize: When the discrepancies are assumed to be i.i.d. with a normal distribution function, the maximum likelihood estimate is obtained by minimizing the sum of squares.

§10.2 Linear Least-Squares Problems

Many models $\varphi(x;t)$ in data-fitting problems are linear functions of x. In these cases, the residuals $r_j(x)$ defined by (8) also are linear, and the problem of minimizing (7) is called a **linear least-squares problem**. We can write the residual vector as $r(x) = Jx - y$ for some matrix J and vector y, both independent of x, so that the objective is

$$f(x) = \frac{1}{2}\|Jx - y\|^2, \tag{13}$$

where $y = -r(0)$. We also have

$$\nabla f(x) = J^{\mathrm{T}}(Jx - y), \quad (\nabla^2 f)(x) = J^{\mathrm{T}}J.$$

Note that the second term in

$$\nabla^2 f(x) = J(x)^{\mathrm{T}}J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x). \tag{5}$$

disappears, because $\nabla^2 r_j = 0$ for all $j = 1, 2, \cdots, m$.

§10.2 Linear Least-Squares Problems

Many models $\varphi(x;t)$ in data-fitting problems are linear functions of x. In these cases, the residuals $r_j(x)$ defined by (8) also are linear, and the problem of minimizing (7) is called a **linear least-squares problem**. We can write the residual vector as $r(x) = Jx - y$ for some matrix J and vector y, both independent of x, so that the objective is

$$f(x) = \frac{1}{2}\|Jx - y\|^2, \tag{13}$$

where $y = -r(0)$. We also have

$$\nabla f(x) = J^{\mathrm{T}}(Jx - y), \quad (\nabla^2 f)(x) = J^{\mathrm{T}}J.$$

Note that the second term in

$$\nabla^2 f(x) = J(x)^{\mathrm{T}}J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x). \tag{5}$$

disappears, because $\nabla^2 r_j = 0$ for all $j = 1, 2, \cdots, m$.

§10.2 Linear Least-Squares Problems

It is easy to see that the f in (13) is convex – a property that does not necessarily hold for the nonlinear problem (1). When f is convex, any point x_* for which $\nabla f(x_*) = 0$ is the global minimizer of f. Therefore, a minimizer x_* for problem

$$f(x) = \frac{1}{2}\|Jx - y\|^2, \tag{13}$$

must satisfy the following linear system of equations:

$$J^{\mathrm{T}}Jx_* = J^{\mathrm{T}}y. \tag{14}$$

These are known as the **normal equations** for (13).

In the following, we outline briefly three major algorithms for the unconstrained linear least-squares problem. We assume in most of our discussion that $m \geqslant n$ and that J has **full column rank**.

§10.2 Linear Least-Squares Problems

It is easy to see that the f in (13) is convex – a property that does not necessarily hold for the nonlinear problem (1). When f is convex, any point x_* for which $\nabla f(x_*) = 0$ is the global minimizer of f. Therefore, a minimizer x_* for problem

$$f(x) = \frac{1}{2}\|Jx - y\|^2, \tag{13}$$

must satisfy the following linear system of equations:

$$J^{\mathrm{T}}Jx_* = J^{\mathrm{T}}y. \tag{14}$$

These are known as the **normal equations** for (13).

In the following, we outline briefly three major algorithms for the unconstrained linear least-squares problem. We assume in most of our discussion that $m \geqslant n$ and that J has **full column rank**.

§10.2 Linear Least-Squares Problems

The first and most obvious algorithm is simply to form and solve the normal equation (14) by the following three-step procedure:

1. compute the coefficient matrix $J^{\mathrm{T}}J$ and the right-hand side $J^{\mathrm{T}}y$;
2. compute the Cholesky factorization of the matrix $J^{\mathrm{T}}J$;
3. perform two triangular substitutions with the Cholesky factors to recover the solution x_*.

The Cholesky factorization

$$J^{\mathrm{T}}J = \bar{R}^{\mathrm{T}}\bar{R}\,, \tag{15}$$

where $\bar{R}$ is an $n \times n$ upper triangular with positive diagonal elements, is guaranteed to exist when $m \geqslant n$ and J has rank n.

§10.2 Linear Least-Squares Problems

This method is frequently used in practice and is often effective, but it has one significant **disadvantage**, namely, that the condition number of $J^{\mathrm{T}}J$ is the square of the condition number of J. Since the relative error in the computed solution of a problem is usually proportional to the condition number, the Cholesky-based method may result in less accurate solutions than those obtained from methods that avoid this squaring of the condition number. When J is ill conditioned, the Cholesky factorization process may even break down, since roundoff errors may cause small negative elements to appear on the diagonal during the factorization process.

§10.2 Linear Least-Squares Problems

A second approach is based on a QR factorization of the matrix J. Since the Euclidean norm of any vector is not affected by orthogonal transformations, we have

$$\|Jx - y\| = \|Q^{\mathrm{T}}(Jx - y)\| \tag{16}$$

for any $m \times m$ orthogonal matrix Q. Suppose we perform a QR factorization with column pivoting on the matrix J to obtain

$$J\Pi = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R, \tag{17}$$

where

1. Π is an $n \times n$ permutation matrix (hence, orthogonal);
2. Q is $m \times m$ orthogonal;
3. Q_1 is the first n columns of Q, while Q_2 contains the last $m-n$ columns;
4. R is $n \times n$ upper triangular with positive diagonal elements.

§10.2 Linear Least-Squares Problems

By combining (16) and (17), we obtain

$$\|Jx-y\|^2 = \left\| \begin{bmatrix} Q_1^{\mathrm{T}} \\ Q_2^{\mathrm{T}} \end{bmatrix} (J\Pi\Pi^{\mathrm{T}}x - y) \right\|^2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \Pi^{\mathrm{T}}x - \begin{bmatrix} Q_1^{\mathrm{T}}y \\ Q_2^{\mathrm{T}}y \end{bmatrix} \right\|^2$$
$$= \|R(\Pi^{\mathrm{T}}x) - Q_1^{\mathrm{T}}y\|^2 + \|Q_2^{\mathrm{T}}y\|^2 . \qquad (18)$$

We can minimize $\|Jx-y\|$ by driving the first term to zero; that is, by setting

$$x_* = \Pi R^{-1} Q_1^{\mathrm{T}} y .$$

This QR-based approach does not degrade the conditioning of the problem unnecessarily. The relative error in the final computed solution x_* is usually proportional to the condition number of J, not its square, and this method is usually reliable. Some situations, however, call for greater robustness or more information about the sensitivity of the solution to perturbations in the data (J or y).

§10.2 Linear Least-Squares Problems

By combining (16) and (17), we obtain

$$\|Jx-y\|^2 = \left\| \begin{bmatrix} Q_1^{\mathrm{T}} \\ Q_2^{\mathrm{T}} \end{bmatrix} (J\Pi\Pi^{\mathrm{T}}x-y) \right\|^2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \Pi^{\mathrm{T}}x - \begin{bmatrix} Q_1^{\mathrm{T}}y \\ Q_2^{\mathrm{T}}y \end{bmatrix} \right\|^2$$
$$= \|R(\Pi^{\mathrm{T}}x) - Q_1^{\mathrm{T}}y\|^2 + \|Q_2^{\mathrm{T}}y\|^2 . \qquad (18)$$

We can minimize $\|Jx-y\|$ by driving the first term to zero; that is, by setting

$$x_* = \Pi R^{-1} Q_1^{\mathrm{T}} y .$$

This QR-based approach does not degrade the conditioning of the problem unnecessarily. The relative error in the final computed solution x_* is usually proportional to the condition number of J, not its square, and this method is usually reliable. Some situations, however, call for greater robustness or more information about the sensitivity of the solution to perturbations in the data (J or y).

§10.2 Linear Least-Squares Problems

A third approach, based on the singular-value decomposition (SVD) of J, can be used in these circumstances. Recall that the SVD of J is given by

$$J = U \begin{bmatrix} S \\ 0 \end{bmatrix} V^{\mathrm{T}} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S \\ 0 \end{bmatrix} V^{\mathrm{T}} = U_1 S V^{\mathrm{T}}, \qquad (19)$$

where

1. U is $m \times m$ orthogonal;
2. U_1 contains the first n columns of U, U_2 the last $m-n$ columns;
3. V is $n \times n$ orthogonal;
4. S is $n \times n$ diagonal, with diagonal elements $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n > 0$.

Note that $J^{\mathrm{T}}J = VS^2V^{\mathrm{T}}$, so that the columns of V are eigenvectors of $J^{\mathrm{T}}J$ with eigenvalues σ_j^2, $j = 1, 2, \cdots, n$.

§10.2 Linear Least-Squares Problems

By following the same logic that led to (18), we obtain

$$\begin{aligned}\|Jx-y\|^2 &= \left\| \begin{bmatrix} S \\ 0 \end{bmatrix} (V^{\mathrm{T}}x) - \begin{bmatrix} U_1^{\mathrm{T}}y \\ U_2^{\mathrm{T}}y \end{bmatrix} \right\|^2 \\ &= \|S(V^{\mathrm{T}}x) - U_1^{\mathrm{T}}y\|^2 + \|U_2^{\mathrm{T}}y\|^2 . \end{aligned} \tag{20}$$

Again, the optimum is found by choosing x to make the first term equal to zero; that is,

$$x_* = VS^{-1}U_1^{\mathrm{T}}y .$$

Denoting the i-th columns of U and V by $u_i \in \mathbb{R}^m$ and $v_i \in \mathbb{R}^n$, respectively, we have

$$x_* = \sum_{i=1}^{n} \frac{u_i^{\mathrm{T}}y}{\sigma_i} v_i . \tag{21}$$

§10.2 Linear Least-Squares Problems

(21) 式

$$x_* = \sum_{i=1}^{n} \frac{u_i^{\mathrm{T}} y}{\sigma_i} v_i \tag{21}$$

提供了有關 x_* 敏感性的有用信息。當 σ_i 較小時，x_* 對於影響 $u_i^{\mathrm{T}} y$ 值的 y 的擾動以及 J 的擾動特別敏感。這樣的信息在 J 幾乎是 rank-deficient 時（亦即 $\sigma_n/\sigma_1 \ll 1$ 時）尤其有用，因此有時使用需付出更多計算量的 SVD 演算法來獲取這種敏感性信息是相當值得的。

§10.2 Linear Least-Squares Problems

上述的三種方法都有其適用的情境。基於 Cholesky 的演算法在 $m \gg n$ 且存儲 $J^{\mathrm{T}}J$ 而非 J 本身實際可行時特別有用。當 $m \gg n$ 且 J 為稀疏矩陣時，這種方法可能也比替代方案更經濟。然而，當 J 為 rank-deficient 或 ill-conditioned 時，必須修改此方法，以允許對 $J^{\mathrm{T}}J$ 的對角元素進行 pivoting 操作。QR 方法避免了對條件數進行平方運算，因此在數值上可能更為穩健 (robust)。

§10.2 Linear Least-Squares Problems

演算法的穩健性 (robustness) 通常是指演算法對輸入的變化或擾動有良好的適應能力，且在面對不確定性或異常情況時能維持良好的性能。具體來說，演算法的穩健性表現在以下幾個方面：

1. 對 noise 的容忍度：一個穩健的演算法應該能夠處理輸入中的 noise 或隨機變動，而不至於產生過度的影響或錯誤。
2. 對參數變化的穩定性：如果演算法的性能不會過度受到輸入參數的小幅度變化的影響，則它被視為是穩健的。
3. 處理異常情況的能力：穩健的演算法應該能夠處理輸入中的異常情況或極端值，而不至於崩潰或產生不合理的輸出。
4. 數值穩定性：對於數值計算而言，演算法應該在面對浮點數誤差、數值不穩定性或數值爆炸的情況下保持穩定。

總的來說，演算法的穩健性是指它在各種不同條件下都能夠保持良好性能的特性，而不容易因為輸入的變化或不確定性而失效。

§10.2 Linear Least-Squares Problems

演算法的穩健性 (robustness) 通常是指演算法對輸入的變化或擾動有良好的適應能力，且在面對不確定性或異常情況時能維持良好的性能。具體來說，演算法的穩健性表現在以下幾個方面：

1. 對 noise 的容忍度：一個穩健的演算法應該能夠處理輸入中的 noise 或隨機變動，而不至於產生過度的影響或錯誤。
2. 對參數變化的穩定性：如果演算法的性能不會過度受到輸入參數的小幅度變化的影響，則它被視為是穩健的。
3. 處理異常情況的能力：穩健的演算法應該能夠處理輸入中的異常情況或極端值，而不至於崩潰或產生不合理的輸出。
4. 數值穩定性：對於數值計算而言，演算法應該在面對浮點數誤差、數值不穩定性或數值爆炸的情況下保持穩定。

總的來說，演算法的穩健性是指它在各種不同條件下都能夠保持良好性能的特性，而不容易因為輸入的變化或不確定性而失效。

§10.2 Linear Least-Squares Problems

演算法的穩健性 (robustness) 通常是指演算法對輸入的變化或擾動有良好的適應能力，且在面對不確定性或異常情況時能維持良好的性能。具體來說，演算法的穩健性表現在以下幾個方面：

1. 對 noise 的容忍度：一個穩健的演算法應該能夠處理輸入中的 noise 或隨機變動，而不至於產生過度的影響或錯誤。
2. 對參數變化的穩定性：如果演算法的性能不會過度受到輸入參數的小幅度變化的影響，則它被視為是穩健的。
3. 處理異常情況的能力：穩健的演算法應該能夠處理輸入中的異常情況或極端值，而不至於崩潰或產生不合理的輸出。
4. 數值穩定性：對於數值計算而言，演算法應該在面對浮點數誤差、數值不穩定性或數值爆炸的情況下保持穩定。

總的來說，演算法的穩健性是指它在各種不同條件下都能夠保持良好性能的特性，而不容易因為輸入的變化或不確定性而失效。

§10.2 Linear Least-Squares Problems

儘管可能是最花計算量的方法，奇異值分解 (SVD) 方法是所有方法中最穩健 (robust) 且可靠的。當矩陣 J 真正是 rank-deficient 時，一些奇異值 σ_i 恰好為零，而對於任意係數 τ_i，形如

$$x_* = \sum_{\sigma_i \neq 0} \frac{u_i^{\mathrm{T}} y}{\sigma_i} v_i + \sum_{\sigma_i = 0} \tau_i v_i \tag{22}$$

的任意向量 x_* 都是 (20) 的 minimizer。通常，具有最小範數的解是最理想的 – 我們可以通過在 (22) 中將每個 τ_i 設為零來取得最小範數的解。當 J 具有 full rank 但是是 ill-conditioned 時，最後幾個奇異值 $\sigma_n, \sigma_{n-1}, \cdots$ 相對於 σ_1 特別小。在 (22) 中的係數 $u_i^{\mathrm{T}} y / \sigma_i$ 在 σ_i 較小時對 $u_i^{\mathrm{T}} y$ 的擾動特別敏感，因此，通過從 (22) 式等號右邊的第一個和中省略這些項目，可以獲得對擾動不太敏感、比真實解更為穩健的近似解。

§10.2 Linear Least-Squares Problems

儘管可能是最花計算量的方法，奇異值分解 (SVD) 方法是所有方法中最穩健 (robust) 且可靠的。當矩陣 J 真正是 rank-deficient 時，一些奇異值 σ_i 恰好為零，而對於任意係數 τ_i，形如

$$x_* = \sum_{\sigma_i \neq 0} \frac{u_i^{\mathrm{T}} y}{\sigma_i} v_i + \sum_{\sigma_i = 0} \tau_i v_i \tag{22}$$

的任意向量 x_* 都是 (20) 的 minimizer。通常，具有最小範數的解是最理想的 – 我們可以通過在 (22) 中將每個 τ_i 設為零來取得最小範數的解。當 J 具有 full rank 但是是 ill-conditioned 時，最後幾個奇異值 $\sigma_n, \sigma_{n-1}, \cdots$ 相對於 σ_1 特別小。在 (22) 中的係數 $u_i^{\mathrm{T}} y / \sigma_i$ 在 σ_i 較小時對 $u_i^{\mathrm{T}} y$ 的擾動特別敏感，因此，通過從 (22) 式等號右邊的第一個和中省略這些項目，可以獲得對擾動不太敏感、比真實解更為穩健的近似解。

§10.2 Linear Least-Squares Problems

當問題的規模很大 ($n \gg 1$) 時，使用如共軛梯度法的迭代方法來解決 normal equation (14) 可能是有效的。直接實施共軛梯度法 (Algorithm 5.2) 需要在每次迭代中執行一次矩陣 $J^{\mathrm{T}}J$ 與向量的乘法。這個操作可以通過連續使用 J 和 J^{T} 進行乘法來完成；我們只需要能夠對這兩個矩陣執行矩陣–向量乘法來實現此算法。共軛梯度法的一些修改方案已經被提出，這些方案涉及每次迭代執行相似工作量的情況（分別與 J 和 J^{T} 進行一次矩陣–向量乘法），但具有更優越的數值特性。Paige 和 Saunders 在 [234] 中提出了一些替代方案，他們尤其提出了一種名為 LSQR 的算法，已成為非常成功的方法的基礎。

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **The Gauss-Newton method**

We now describe methods for minimizing the nonlinear objective function (1) that exploit the structure of the gradient ∇f and Hessian $\nabla^2 f$ in

$$\nabla f(x) = J(x)^{\mathrm{T}} r(x)\,, \tag{4}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x)\,. \tag{5}$$

The simplest of these methods – the Gauss-Newton method – can be viewed as a modified Newton's method with line search. Instead of solving the standard Newton equations $(\nabla^2 f)(x_k)p = -(\nabla f)(x_k)$, we solve instead the following system to obtain the search direction p_k^{GN}:

$$J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}} = -J_k^{\mathrm{T}} r_k\,. \tag{23}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

This simple modification gives a number of advantages over the plain Newton's method. First, the use of the approximation

$$\nabla^2 f_k \approx J_k^{\mathrm{T}} J_k \tag{24}$$

saves us the trouble of computing the individual residual Hessians $\nabla^2 r_j$, $j = 1, 2, \cdots, m$, which are needed in the second term in (5). In fact, if we calculated the Jacobian J_k in the course of evaluating the gradient $\nabla f_k = J_k^{\mathrm{T}} r_k$, the approximation (24) does not require any additional derivative evaluations, and the savings in computational time can be quite significant in some applications. Second, there are many interesting situations in which the first term $J^{\mathrm{T}} J$ in (5) dominates the second term (at least close to the solution x_*), so that $J_k^{\mathrm{T}} J_k$ is a close approximation to $\nabla^2 f_k$ and the convergence rate of Gauss-Newton is similar to that of Newton's method.

§10.3 Algorithms for Nonlinear Least-Squares Problems

This simple modification gives a number of advantages over the plain Newton's method. First, the use of the approximation

$$\nabla^2 f_k \approx J_k^{\mathrm{T}} J_k \tag{24}$$

saves us the trouble of computing the individual residual Hessians $\nabla^2 r_j$, $j = 1, 2, \cdots, m$, which are needed in the second term in (5). In fact, if we calculated the Jacobian J_k in the course of evaluating the gradient $\nabla f_k = J_k^{\mathrm{T}} r_k$, the approximation (24) does not require any additional derivative evaluations, and the savings in computational time can be quite significant in some applications. Second, there are many interesting situations in which the first term $J^{\mathrm{T}} J$ in (5) dominates the second term (at least close to the solution x_*), so that $J_k^{\mathrm{T}} J_k$ is a close approximation to $\nabla^2 f_k$ and the convergence rate of Gauss-Newton is similar to that of Newton's method.

§10.3 Algorithms for Nonlinear Least-Squares Problems

This simple modification gives a number of advantages over the plain Newton's method. First, the use of the approximation

$$\nabla^2 f_k \approx J_k^{\mathrm{T}} J_k \tag{24}$$

saves us the trouble of computing the individual residual Hessians $\nabla^2 r_j$, $j = 1, 2, \cdots, m$, which are needed in the second term in (5). In fact, if we calculated the Jacobian J_k in the course of evaluating the gradient $\nabla f_k = J_k^{\mathrm{T}} r_k$, the approximation (24) does not require any additional derivative evaluations, and the savings in computational time can be quite significant in some applications. Second, there are many interesting situations in which the first term $J^{\mathrm{T}} J$ in (5) dominates the second term (at least close to the solution x_*), so that $J_k^{\mathrm{T}} J_k$ is a close approximation to $\nabla^2 f_k$ and the convergence rate of Gauss-Newton is similar to that of Newton's method.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The first term $J^{\mathrm{T}}J$ in (5) will be dominant when the norm of each second-order term (that is, $|r_j(x)|\|\nabla^2 r_j(x)\|$) is significantly smaller than the eigenvalues of $J^{\mathrm{T}}J$. As mentioned in the introduction, we tend to see this behavior when either the residuals r_j are small or when they are nearly affine (so that the $\|\nabla^2 r_j\|$ are small). In practice, many least-squares problems have small residuals at the solution, leading to rapid local convergence of Gauss-Newton.

§10.3 Algorithms for Nonlinear Least-Squares Problems

A third advantage of Gauss-Newton is that whenever J_k has full rank and the gradient ∇f_k is nonzero, the direction p_k^{GN} is a descent direction for f, and therefore a suitable direction for a line search. From (4) and the definition of p_k^{GN}

$$J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}} = -J_k^{\mathrm{T}} r_k \tag{23}$$

we have

$$\begin{aligned}(p_k^{\mathrm{GN}})^{\mathrm{T}} \nabla f_k &= (p_k^{\mathrm{GN}})^{\mathrm{T}} J_k^{\mathrm{T}} r_k = -(p_k^{\mathrm{GN}})^{\mathrm{T}} J_k^{\mathrm{T}} J_k \, p_k^{\mathrm{GN}} \\ &= -\|J_k p_k^{\mathrm{GN}}\|^2 \leqslant 0\,. \end{aligned} \tag{25}$$

The final inequality is strict unless $J_k p_k^{\mathrm{GN}} = 0$, in which case we have by (23) and full rank of J_k that $\nabla f_k = J_k^{\mathrm{T}} r_k = 0$; that is, x_k is a stationary point.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Finally, the fourth advantage of Gauss-Newton arises from the similarity between the equations (23) and the normal equations (14) for the linear least-squares problem. This connection tells us that p_k^{GN} is in fact the solution of the linear least-squares problem

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 . \tag{26}$$

Hence, we can find the search direction by applying linear least-squares algorithms to the sub-problem (26). In fact, if the QR or SVD-based algorithms are used, there is no need to calculate the Hessian approximation $J_k^{\mathrm{T}} J_k$ in (23) explicitly; we can work directly with the Jacobian J_k. The same is true if we use a conjugate-gradient technique to solve (26). For this method we need to perform matrix-vector multiplications with $J_k^{\mathrm{T}} J_k$, which can be done by first multiplying by J_k and then by J_k^{T}.

§10.3 Algorithms for Nonlinear Least-Squares Problems

If the number of residuals m is large while the number of variables n is relatively small, it may be unwise to store the Jacobian J explicitly. A preferable strategy may be to calculate the matrix $J^{\mathrm{T}}J$ and gradient vector $J^{\mathrm{T}}r$ by evaluating r_j and ∇r_j successively for $j = 1, 2, \cdots, m$ and performing the accumulations

$$J^{\mathrm{T}}J = \sum_{i=1}^{m} (\nabla r_j)(\nabla r_j)^{\mathrm{T}}, \quad J^{\mathrm{T}}r = \sum_{i=1}^{m} r_j(\nabla r_j). \tag{27}$$

The Gauss-Newton steps can then be computed by solving the system (23) of normal equations directly.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The sub-problem (26) suggests another motivation for the Gauss-Newton search direction. We can view this equation as being obtained from a linear model for the vector function $r(x_k + p) \approx r_k + J_k p$, substituted into the function $\frac{1}{2}\|\cdot\|^2$. In other words, we use the approximation

$$f(x_k + p) = \frac{1}{2}\|r(x_k + p)\|^2 \approx \frac{1}{2}\|J_k p + r_k\|^2,$$

and choose p_k^{GN} to be the minimizer of this approximation.

Implementations of the Gauss-Newton method usually perform a line search in the direction p_k^{GN}, requiring the step length α_k to satisfy conditions like those discussed in Chapter 3, such as the Armijo and Wolfe condition.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Convergence of the Gauss-Newton method**

The theory of Chapter 3 can applied to study the convergence properties of the Gauss-Newton method. We prove a global convergence result under the assumption that the Jacobian matrix J has its singular values uniformly bounded away from zero in the region of interest; that is, there is a constant $\gamma > 0$ such that

$$\|J(x)z\| \geqslant \gamma\|z\| \tag{28}$$

for all x in a neighborhood $\mathcal{N}$ of the level set

$$S = \left\{x \middle| f(x) \leqslant f(x_0)\right\}, \tag{29}$$

where x_0 is the starting point for the algorithm. We assume here and in the rest of the chapter that S is bounded. Our result is a consequence of Zoutendijk's Theorem.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Convergence of the Gauss-Newton method**

The theory of Chapter 3 can applied to study the convergence properties of the Gauss-Newton method. We prove a global convergence result under the assumption that the Jacobian matrix J has its singular values uniformly bounded away from zero in the region of interest; that is, there is a constant $\gamma > 0$ such that

$$\|J(x)z\| \geqslant \gamma\|z\| \tag{28}$$

for all x in a neighborhood $\mathcal{N}$ of the level set

$$S = \left\{x \middle| f(x) \leqslant f(x_0)\right\}, \tag{29}$$

where x_0 is the starting point for the algorithm. We assume here and in the rest of the chapter that S is bounded. Our result is a consequence of Zoutendijk's Theorem.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem

Suppose each residual function r_j is Lipschitz continuously differentiable in a neighborhood $\mathcal{N}$ of the bounded level set S given by (29), *and that the Jacobian matrix J satisfies the uniform full-rank condition* (28) *on $\mathcal{N}$. If the iterates x_k are generated by the Gauss-Newton method with step lengths α_k satisfying the Wolfe conditions*

$$f(x_k + \alpha_k p_k) \leqslant f(x_k) + c_1 \alpha_k \nabla f_k^{\mathrm{T}} p_k \,,$$
$$\nabla f(x_k + \alpha_k p_k)^{\mathrm{T}} p_k \geqslant c_2 \nabla f_k^{\mathrm{T}} p_k \,,$$

we have

$$\lim_{k \to \infty} \nabla f_k = \lim_{k \to \infty} J_k^{\mathrm{T}} r_k = 0 \,.$$

Proof.

We first check the validity of the sufficient conditions for applying Zoutendijk's Theorem. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem

Suppose each residual function r_j is Lipschitz continuously differentiable in a neighborhood $\mathcal{N}$ of the bounded level set S given by (29), *and that the Jacobian matrix J satisfies the uniform full-rank condition* (28) *on $\mathcal{N}$. If the iterates x_k are generated by the Gauss-Newton method with step lengths α_k satisfying the Wolfe conditions*

$$f(x_k + \alpha_k p_k) \leqslant f(x_k) + c_1 \alpha_k \nabla f_k^{\mathrm{T}} p_k \,,$$
$$\nabla f(x_k + \alpha_k p_k)^{\mathrm{T}} p_k \geqslant c_2 \nabla f_k^{\mathrm{T}} p_k \,,$$

we have

$$\lim_{k \to \infty} \nabla f_k = \lim_{k \to \infty} J_k^{\mathrm{T}} r_k = 0 \,.$$

Proof.

We first check the validity of the sufficient conditions for applying Zoutendijk's Theorem. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem

Suppose each residual function r_j is Lipschitz continuously differentiable in a neighborhood $\mathcal{N}$ of the bounded level set S given by (29)*, and that the Jacobian matrix J satisfies the uniform full-rank condition* (28) *on $\mathcal{N}$. If the iterates x_k are generated by the Gauss-Newton method with step lengths α_k satisfying the Wolfe conditions*

$$f(x_k + \alpha_k p_k) \leqslant f(x_k) + c_1 \alpha_k \nabla f_k^{\mathrm{T}} p_k \,,$$
$$\nabla f(x_k + \alpha_k p_k)^{\mathrm{T}} p_k \geqslant c_2 \nabla f_k^{\mathrm{T}} p_k \,,$$

we have

$$\lim_{k \to \infty} \nabla f_k = \lim_{k \to \infty} J_k^{\mathrm{T}} r_k = 0 \,.$$

Proof.

We first check the validity of the sufficient conditions for applying Zoutendijk's Theorem. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

By the fact that each r_j is Lipschitz continuously differentiable in $\mathcal{N}$, there exists $L_1 > 0$ such that

$$\|\nabla r_j(x) - \nabla r_j(\tilde{x})\| \leqslant L_1 \|x - \tilde{x}\| \quad \forall\, x, \tilde{x} \in \mathcal{N}, 1 \leqslant j \leqslant m\,.$$

Since $\mathcal{N}$ is an open set containing the compact set S, there exist $r > 0$ and $y_1, y_2, \cdots, y_K$ such that

1. $B(x, r) \subseteq \mathcal{N}$ for all $x \in S$.
2. $S \subseteq \bigcup_{k=1}^{K} B(y_k, r) \equiv \tilde{\mathcal{N}} \subseteq \mathcal{N}$.

Note that $\tilde{\mathcal{N}}$ is bounded; thus there exists $\beta_1 > 0$ such that

$$\|\nabla r_j(x)\| \leqslant \beta_1 \quad \forall\, x \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,.$$

Therefore, Taylor's Theorem implies that

$$\|r_j(x) - r_j(\tilde{x})\| \leqslant K\beta_1 \|x - \tilde{x}\| \quad \forall\, x, \tilde{x} \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,. \qquad \square$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

By the fact that each r_j is Lipschitz continuously differentiable in $\mathcal{N}$, there exists $L_1 > 0$ such that

$$\|\nabla r_j(x) - \nabla r_j(\tilde{x})\| \leqslant L_1 \|x - \tilde{x}\| \quad \forall\, x, \tilde{x} \in \mathcal{N}, 1 \leqslant j \leqslant m\,.$$

Since $\mathcal{N}$ is an open set containing the compact set S, there exist $r > 0$ and $y_1, y_2, \cdots, y_K$ such that

1. $B(x, r) \subseteq \mathcal{N}$ for all $x \in S$. 2. $S \subseteq \bigcup_{k=1}^{K} B(y_k, r) \equiv \tilde{\mathcal{N}} \subseteq \mathcal{N}$.

Note that $\tilde{\mathcal{N}}$ is bounded; thus there exists $\beta_1 > 0$ such that

$$\|\nabla r_j(x)\| \leqslant \beta_1 \quad \forall\, x \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,.$$

Therefore, Taylor's Theorem implies that

$$\|r_j(x) - r_j(\tilde{x})\| \leqslant K\beta_1 \|x - \tilde{x}\| \quad \forall\, x, \tilde{x} \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,. \qquad \square$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

In other words, r_j is Lipschitz continuous in $\tilde{\mathcal{N}}$ for $1 \leqslant j \leqslant m$, and the boundedness of $\mathcal{N}$ again provides $\beta_2 > 0$ such that

$$\|r_j(x)\| \leqslant \beta_2 \quad \forall\, x \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,.$$

From these upper bounds on r_j and ∇r_j and the fact that r_j and ∇r_j are Lipschitz continuous on $\tilde{\mathcal{N}}$, we find that ∇f is Lipschitz continuous in $\tilde{\mathcal{N}}$. Since f is bounded from below by zero, the assumptions of Zoutendijk's Theorem are satisfied; thus

$$\sum_{k=1}^{\infty} \cos^2\theta_k \|\nabla f_k\|^2 < \infty\,,$$

where θ_k is the angle between the search direction p_k^{GN} and the negative gradient $-\nabla f_k$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

In other words, r_j is Lipschitz continuous in $\tilde{\mathcal{N}}$ for $1 \leqslant j \leqslant m$, and the boundedness of $\mathcal{N}$ again provides $\beta_2 > 0$ such that

$$\|r_j(x)\| \leqslant \beta_2 \quad \forall\, x \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,.$$

From these upper bounds on r_j and ∇r_j and the fact that r_j and ∇r_j are Lipschitz continuous on $\tilde{\mathcal{N}}$, we find that ∇f is Lipschitz continuous in $\tilde{\mathcal{N}}$. Since f is bounded from below by zero, the assumptions of Zoutendijk's Theorem are satisfied; thus

$$\sum_{k=1}^{\infty} \cos^2\theta_k \|\nabla f_k\|^2 < \infty\,,$$

where θ_k is the angle between the search direction p_k^{GN} and the negative gradient $-\nabla f_k$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

In other words, r_j is Lipschitz continuous in $\tilde{\mathcal{N}}$ for $1 \leqslant j \leqslant m$, and the boundedness of $\mathcal{N}$ again provides $\beta_2 > 0$ such that

$$\|r_j(x)\| \leqslant \beta_2 \quad \forall\, x \in \tilde{\mathcal{N}}, 1 \leqslant j \leqslant m\,.$$

From these upper bounds on r_j and ∇r_j and the fact that r_j and ∇r_j are Lipschitz continuous on $\tilde{\mathcal{N}}$, we find that ∇f is Lipschitz continuous in $\tilde{\mathcal{N}}$. Since f is bounded from below by zero, the assumptions of Zoutendijk's Theorem are satisfied; thus

$$\sum_{k=1}^{\infty} \cos^2\theta_k \|\nabla f_k\|^2 < \infty\,,$$

where θ_k is the angle between the search direction p_k^{GN} and the negative gradient $-\nabla f_k$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

We now check that the angle θ_k between the search direction p_k^{GN} and the negative gradient $-\nabla f_k$ is uniformly bounded away from $\pi/2$. Before proceeding, note that the smoothness of r_j and the compactness of S shows that

$$\|J(x)^{\mathrm{T}}\| = \|J(x)\| \leqslant \beta \quad \forall\, x \in S$$

for some $\beta > 0$. Since

$$(p_k^{\mathrm{GN}})^{\mathrm{T}} \nabla f_k = -\|J_k p_k^{\mathrm{GN}}\|^2 \quad \text{and} \quad \|J(x)z\| \geqslant \gamma \|z\| \quad \forall\, x \in \tilde{\mathcal{N}}\,,$$

we have for $x_k \in S$ that

$$\cos\theta_k = -\frac{(\nabla f_k)^{\mathrm{T}} p_k^{\mathrm{GN}}}{\|p_k^{\mathrm{GN}}\| \|\nabla f_k\|} = \frac{\|J_k p_k^{\mathrm{GN}}\|^2}{\|p_k^{\mathrm{GN}}\| \|J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}}\|} \geqslant \frac{\gamma^2 \|p_k^{\mathrm{GN}}\|^2}{\beta^2 \|p_k^{\mathrm{GN}}\|^2} = \frac{\gamma^2}{\beta^2} > 0.$$

It then follows from Zoutendijk's condition that $\nabla f_k \to 0$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof (cont'd).

We now check that the angle θ_k between the search direction p_k^{GN} and the negative gradient $-\nabla f_k$ is uniformly bounded away from $\pi/2$. Before proceeding, note that the smoothness of r_j and the compactness of S shows that

$$\|J(x)^{\mathrm{T}}\| = \|J(x)\| \leqslant \beta \quad \forall\, x \in S$$

for some $\beta > 0$. Since

$$(p_k^{\mathrm{GN}})^{\mathrm{T}} \nabla f_k = -\|J_k p_k^{\mathrm{GN}}\|^2 \quad \text{and} \quad \|J(x)z\| \geqslant \gamma \|z\| \quad \forall\, x \in \tilde{\mathcal{N}}\,,$$

we have for $x_k \in S$ that

$$\cos\theta_k = -\frac{(\nabla f_k)^{\mathrm{T}} p_k^{\mathrm{GN}}}{\|p_k^{\mathrm{GN}}\|\|\nabla f_k\|} = \frac{\|J_k p_k^{\mathrm{GN}}\|^2}{\|p_k^{\mathrm{GN}}\|\|J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}}\|} \geqslant \frac{\gamma^2 \|p_k^{\mathrm{GN}}\|^2}{\beta^2 \|p_k^{\mathrm{GN}}\|^2} = \frac{\gamma^2}{\beta^2} > 0.$$

It then follows from Zoutendijk's condition that $\nabla f_k \to 0$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

If J_k is rank-deficient for some k (so that a condition like (28) is not satisfied), the coefficient matrix in

$$J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}} = -J_k^{\mathrm{T}} r_k \,. \tag{23}$$

is singular. Nevertheless, the system (23) still has a solution because of the equivalence between this linear system and the minimization problem

$$\min_p \frac{1}{2} \| J_k p + r_k \|^2 \,. \tag{26}$$

In fact, there are infinitely many solutions for p_k^{GN} in this case; each of them has the form of

$$x_* = \sum_{\sigma_i \neq 0} \frac{u_i^{\mathrm{T}} y}{\sigma_i} v_i + \sum_{\sigma_i = 0} \tau_i v_i \,. \tag{22}$$

However, there is no longer an assurance that $\cos\theta_k$ is uniformly bounded away from zero, so we cannot prove a result like the theorem above.

§10.3 Algorithms for Nonlinear Least-Squares Problems

If J_k is rank-deficient for some k (so that a condition like (28) is not satisfied), the coefficient matrix in

$$J_k^{\mathrm{T}} J_k p_k^{\mathrm{GN}} = -J_k^{\mathrm{T}} r_k \,. \tag{23}$$

is singular. Nevertheless, the system (23) still has a solution because of the equivalence between this linear system and the minimization problem

$$\min_p \frac{1}{2} \| J_k p + r_k \|^2 \,. \tag{26}$$

In fact, there are infinitely many solutions for p_k^{GN} in this case; each of them has the form of

$$x_* = \sum_{\sigma_i \neq 0} \frac{u_i^{\mathrm{T}} y}{\sigma_i} v_i + \sum_{\sigma_i = 0} \tau_i v_i \,. \tag{22}$$

However, there is no longer an assurance that $\cos\theta_k$ is uniformly bounded away from zero, so we cannot prove a result like the theorem above.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The convergence of Gauss-Newton to a solution x_* can be rapid if the leading term $J_k^{\mathrm{T}} J_k$ dominates the second-order term in the Hessian (5). Suppose that x_k is close to x_* and that assumption

$$\exists\, \gamma > 0 \ni \|J(x)z\| \geqslant \gamma \|z\| \quad \text{for all } z \in \mathbb{R}^n \tag{28}$$

is satisfied. Then, applying an argument like the Newton's method analysis in Chapter 3, we have for a unit step in the Gauss-Newton direction that

$$\begin{aligned} x_k + p_k^{\mathrm{GN}} - x_* &= x_k - x_* - \left(J_k^{\mathrm{T}} J_k\right)^{-1} \nabla f_k \\ &= \left(J_k^{\mathrm{T}} J_k\right)^{-1} \left[\left(J_k^{\mathrm{T}} J_k\right)(x_k - x_*) + \nabla f_* - \nabla f_k\right]. \end{aligned}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Let H to denote the second-order term in

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x)\,. \tag{5}$$

By the Fundamental Theorem of Calculus we have

$$\begin{aligned}\nabla f_k - \nabla f_* = &\int_0^1 \big(J^{\mathrm{T}} J\big)\big(x_* + t(x_k - x_*)\big)(x_k - x_*)\,dt \\ &+ \int_0^1 H\big(x_* + t\big(x_k - x_*\big)\big)(x_k - x_*)\,dt\,.\end{aligned}$$

Assuming Lipschitz continuity of H near x_*, we have

$$\begin{aligned}&\|x_k + p_k^{\mathrm{GN}} - x_*\| \\ &\quad\leqslant \int_0^1 \big\|\big(J_k^{\mathrm{T}} J_k\big)^{-1} H\big(x_* + t(x_k - x_*)\big)\big\|\|x_k - x_*\|dt + \mathcal{O}(\|x_k - x_*\|^2) \\ &\quad\approx \big\|\big[J^{\mathrm{T}} J(x_*)\big]^{-1} H(x_*)\big\|\|x_k - x_*\| + \mathcal{O}(\|x_k - x_*\|^2)\,. \qquad (30)\end{aligned}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Hence, if $\left\|[J^{\mathrm{T}}J(x_*)]^{-1}H(x_*)\right\| \ll 1$, we can expect a unit step of Gauss-Newton to move us much closer to the solution x_*, giving rapid local convergence. When $H(x_*) = 0$, the convergence is actually quadratic.

When n and m are both large and the Jacobian matrix J is sparse, the cost of computing steps exactly by factoring either J_k or $J_k^{\mathrm{T}}J_k$ at each iteration may become quite expensive relative to the cost of function and gradient evaluations. In this case, we can design inexact variants of the Gauss-Newton algorithm that are analogous to the inexact Newton algorithms discussed in Chapter 7. We simply replace the Hessian $\nabla^2 f_k$ in these methods by its approximation $J_k^{\mathrm{T}}J_k$. The positive semi-definiteness of this approximation simplifies the resulting algorithms in several places.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Hence, if $\left\|[J^{\mathrm{T}}J(x_*)]^{-1}H(x_*)\right\| \ll 1$, we can expect a unit step of Gauss-Newton to move us much closer to the solution x_*, giving rapid local convergence. When $H(x_*) = 0$, the convergence is actually quadratic.

When n and m are both large and the Jacobian matrix J is sparse, the cost of computing steps exactly by factoring either J_k or $J_k^{\mathrm{T}}J_k$ at each iteration may become quite expensive relative to the cost of function and gradient evaluations. In this case, we can design inexact variants of the Gauss-Newton algorithm that are analogous to the inexact Newton algorithms discussed in Chapter 7. We simply replace the Hessian $\nabla^2 f_k$ in these methods by its approximation $J_k^{\mathrm{T}}J_k$. The positive semi-definiteness of this approximation simplifies the resulting algorithms in several places.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **The Levenberg-Marquardt method**

Recall that the Gauss-Newton method is like Newton's method with line search, except that we use the convenient and often effective approximation

$$\nabla^2 f_k \approx J_k^{\mathrm{T}} J_k \tag{24}$$

for the Hessian. The Levenberg-Marquardt method can be obtained by using the same Hessian approximation, but replacing the line search with a trust-region strategy. The use of a trust region avoids one of the weaknesses of Gauss-Newton, namely, its behavior when the Jacobian matrix J is rank-deficient, or nearly so. Since the same Hessian approximations are used in each case, the local convergence properties of the two methods are similar.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **The Levenberg-Marquardt method**

Recall that the Gauss-Newton method is like Newton's method with line search, except that we use the convenient and often effective approximation

$$\nabla^2 f_k \approx J_k^{\mathrm{T}} J_k \tag{24}$$

for the Hessian. The Levenberg-Marquardt method can be obtained by using the same Hessian approximation, but replacing the line search with a trust-region strategy. The use of a trust region avoids one of the weaknesses of Gauss-Newton, namely, its behavior when the Jacobian matrix J is rank-deficient, or nearly so. Since the same Hessian approximations are used in each case, the local convergence properties of the two methods are similar.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The Levenberg-Marquardt method can be described and analyzed using the trust region framework of Chapter 4. In fact, the Levenberg-Marquardt method is sometimes considered to be the progenitor (前身) of the trust-region approach for general unconstrained optimization discussed in Chapter 4. For a spherical trust region, the subproblem to be solved at each iteration is

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \Delta_k\,, \tag{31}$$

where $\Delta_k > 0$ is the trust-region radius. In effect, we are choosing the model function $m_k(\cdot)$ to be

$$m_k(p) = \frac{1}{2}\|r_k\|^2 + (J_k^{\mathrm{T}} r_k)^{\mathrm{T}} p + \frac{1}{2} p^{\mathrm{T}} J_k^{\mathrm{T}} J_k\, p\,. \tag{32}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

The Levenberg-Marquardt method can be described and analyzed using the trust region framework of Chapter 4. In fact, the Levenberg-Marquardt method is sometimes considered to be the progenitor (前身) of the trust-region approach for general unconstrained optimization discussed in Chapter 4. For a spherical trust region, the subproblem to be solved at each iteration is

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \Delta_k\,, \tag{31}$$

where $\Delta_k > 0$ is the trust-region radius. In effect, we are choosing the model function $m_k(\cdot)$ to be

$$m_k(p) = \frac{1}{2}\|r_k\|^2 + (J_k^{\mathrm{T}} r_k)^{\mathrm{T}} p + \frac{1}{2} p^{\mathrm{T}} J_k^{\mathrm{T}} J_k\, p\,. \tag{32}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

We drop the iteration counter k during the rest of this section and concern ourselves with the sub-problem (31). The results of Chapter 4 allow us to characterize the solution of (31) in the following way: When the solution p^{GN} of the Gauss-Newton equations (23) lies strictly inside the trust region (that is, $\|p^{\mathrm{GN}}\| < \Delta$), then this step p^{GN} also solves the sub-problem (31). Otherwise, there is a $\lambda > 0$ such that the solution $p = p^{\mathrm{LM}}$ of (31) satisfies $\|p\| = \Delta$ and

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p = -J^{\mathrm{T}}r. \tag{33}$$

This claim is verified in the following lemma, which is a straightforward consequence of the key theorem in Section 4.3 that we recall/state in the next slide.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem (Key theorem in Section 4.3)

The vector p_* is a **global** *solution of the trust-region problem*

$$\min_{p\in\mathbb{R}^n} m(p) \equiv f + g^{\mathrm{T}}p + \frac{1}{2}p^{\mathrm{T}}Bp \quad \text{s.t.} \quad \|p\| \leqslant \Delta\,. \tag{5}_4$$

if and only if p_* is feasible and there is a scalar $\lambda \geqslant 0$ such that the following conditions are satisfied:

$$(B+\lambda \mathrm{I})p_* = -g\,, \tag{6a}_4$$

$$\lambda(\Delta - \|p_*\|) = 0\,, \tag{6b}_4$$

$$(B+\lambda \mathrm{I}) \text{ is positive semi-definite.} \tag{6c}_4$$

Recall that the model function m under disccusion now is

$$m(p) = \frac{1}{2}\|r\|^2 + (J^{\mathrm{T}}r)^{\mathrm{T}}p + \frac{1}{2}p^{\mathrm{T}}J^{\mathrm{T}}Jp\,. \tag{32}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem (Key theorem in Section 4.3)

The vector p_* is a **global** *solution of the trust-region problem*

$$\min_{p\in\mathbb{R}^n} m(p) \equiv f + g^{\mathrm{T}}p + \frac{1}{2}p^{\mathrm{T}}Bp \quad \text{s.t.} \quad \|p\| \leqslant \Delta\,. \tag{5}_4$$

if and only if p_* is feasible and there is a scalar $\lambda \geqslant 0$ such that the following conditions are satisfied:

$$(B+\lambda \mathrm{I})p_* = -g\,, \tag{6a}_4$$

$$\lambda(\Delta - \|p_*\|) = 0\,, \tag{6b}_4$$

$$(B+\lambda \mathrm{I}) \text{ is positive semi-definite.} \tag{6c}_4$$

Recall that the model function m under disccusion now is

$$m(p) = \frac{1}{2}\|r\|^2 + (J^{\mathrm{T}}r)^{\mathrm{T}}p + \frac{1}{2}p^{\mathrm{T}}J^{\mathrm{T}}Jp\,. \tag{32}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Lemma

The vector p^{LM} *is a solution of the trust-region sub-problem*

$$\min_{p} \|Jp + r\|^2 \quad \textit{subject to} \quad \|p\| \leqslant \Delta,$$

if and only if p^{LM} *is feasible and there is a scalar* $\lambda \geqslant 0$ *such that*

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p^{\mathrm{LM}} = -J^{\mathrm{T}}r, \tag{34a}$$

$$\lambda(\Delta - \|p^{\mathrm{LM}}\|) = 0\,. \tag{34b}$$

Proof.

Since $J^{\mathrm{T}}J$ is positive semi-definite, it suffices to establish

$$p^{\mathrm{LM}} \text{ is a feasible solution} \quad \Leftrightarrow \quad (\exists \lambda \geqslant 0)\big((34a) \wedge (34b)\big).$$

Nevertheless, (34a) and (34b) are simply $(6a)_4$ and $(6b)_4$ in the key theorem in Section 4.3, respectively, for the case $B = J^{\mathrm{T}}J$ and $g = p^{\mathrm{LM}}$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Lemma

The vector p^{LM} is a solution of the trust-region sub-problem

$$\min_p \|Jp + r\|^2 \quad \textit{subject to} \quad \|p\| \leqslant \Delta,$$

if and only if p^{LM} is feasible and there is a scalar $\lambda \geqslant 0$ such that

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p^{\mathrm{LM}} = -J^{\mathrm{T}}r, \tag{34a}$$

$$\lambda(\Delta - \|p^{\mathrm{LM}}\|) = 0\,. \tag{34b}$$

Proof.

Since $J^{\mathrm{T}}J$ is positive semi-definite, it suffices to establish

$$p^{\mathrm{LM}} \text{ is a feasible solution} \quad \Leftrightarrow \quad (\exists \lambda \geqslant 0)\big((34a) \wedge (34b)\big).$$

Nevertheless, (34a) and (34b) are simply $(6a)_4$ and $(6b)_4$ in the key theorem in Section 4.3, respectively, for the case $B = J^{\mathrm{T}}J$ and $g = p^{\mathrm{LM}}$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Lemma

The vector p^{LM} *is a solution of the trust-region sub-problem*

$$\min_{p} \|Jp + r\|^2 \quad \textit{subject to} \quad \|p\| \leqslant \Delta,$$

if and only if p^{LM} *is feasible and there is a scalar* $\lambda \geqslant 0$ *such that*

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p^{\mathrm{LM}} = -J^{\mathrm{T}}r, \tag{34a}$$

$$\lambda(\Delta - \|p^{\mathrm{LM}}\|) = 0\,. \tag{34b}$$

Proof.

Since $J^{\mathrm{T}}J$ is positive semi-definite, it suffices to establish

$$p^{\mathrm{LM}} \text{ is a feasible solution} \quad \Leftrightarrow \quad (\exists \lambda \geqslant 0)\big((34a) \wedge (34b)\big).$$

Nevertheless, (34a) and (34b) are simply $(6a)_4$ and $(6b)_4$ in the key theorem in Section 4.3, respectively, for the case $B = J^{\mathrm{T}}J$ and $g = p^{\mathrm{LM}}$. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

Note that the equations

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p = -J^{\mathrm{T}}r \tag{33}$$

are just the normal equations for the following linear least-squares problem:

$$\min_{p} \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix} p + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|^2 . \tag{35}$$

Just as in the Gauss-Newton case, the equivalence between (33) and (35) gives us a way of solving the sub-problem without computing the matrix-matrix product $J^{\mathrm{T}}J$ and its Cholesky factorization.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Implementation of the Levenberg-Marquardt method**

To find a λ that approximately matches the given Δ in the lemma, we can use the root-finding algorithm described in Chapter 4. It is easy to safeguard this procedure: The Cholesky factor R is guaranteed to exist whenever the current estimate $\lambda^{(\ell)}$ is positive, since the approximate Hessian $B = J^{\mathrm{T}}J$ is already positive semi-definite. Because of the special structure of B, we do not need to compute the Cholesky factorization of $B+\lambda \mathrm{I}$ from scratch in each iteration of Algorithm 4.1. Rather, we present an efficient technique for finding the following QR factorization of the coefficient matrix in (35):

$$\begin{bmatrix} R_\lambda \\ 0 \end{bmatrix} = Q_\lambda^{\mathrm{T}} \begin{bmatrix} J \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix}, \tag{36}$$

where Q_λ is orthogonal, R_λ is upper triangular. The upper triangular factor R_λ satisfies $R_\lambda^{\mathrm{T}} R_\lambda = J^{\mathrm{T}}J + \lambda \mathrm{I}$.

§10.3 Algorithms for Nonlinear Least-Squares Problems

We can save computer time in the calculation of the factorization (36) by using a combination of Householder and Givens transformations. Suppose we use Householder transformations to calculate the QR factorization of J alone as

$$J = Q \begin{bmatrix} R \\ 0 \end{bmatrix}. \tag{37}$$

We then have

$$\begin{bmatrix} R \\ 0 \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix} = \begin{bmatrix} Q^{\mathrm{T}} & \\ & \mathrm{I} \end{bmatrix} \begin{bmatrix} J \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix}. \tag{38}$$

The leftmost matrix in this formula is upper triangular except for the n nonzero terms of the matrix $\lambda\mathrm{I}$. These can be eliminated by a sequence of $n(n+1)/2$ Givens rotations, in which the diagonal elements of the upper triangular part are used to eliminate the nonzeros of $\lambda\mathrm{I}$ and the fill-in terms that arise in the process.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The first few steps of this process are as follows:

1. rotate row n of R with row n of $\sqrt{\lambda}\mathrm{I}$, to eliminate the (n, n) element of $\sqrt{\lambda}\mathrm{I}$;
2. rotate row $(n-1)$ of R with row $(n-1)$ of $\sqrt{\lambda}\mathrm{I}$ to eliminate the $(n-1, n-1)$ element of the latter matrix. This step introduces fill-in in position $(n-1, n)$ of $\sqrt{\lambda}\mathrm{I}$, which is eliminated by rotating row n of R with row $(n-1)$ of $\sqrt{\lambda}\mathrm{I}$, to eliminate the fill-in element at position $(n-1, n)$;
3. rotate row $(n-2)$ of R with row $(n-2)$ of $\sqrt{\lambda}\mathrm{I}$, to eliminate the $(n-2)$ diagonal in the latter matrix. This step introduces fill-in in the $(n-2, n-1)$ and $(n-2, n)$ positions, which we eliminate by $\cdots$

and so on.

§10.3 Algorithms for Nonlinear Least-Squares Problems

If we gather all the Givens rotations into a matrix $\bar{Q}_\lambda$, we obtain from (38) that

$$\bar{Q}_\lambda^{\mathrm{T}} \begin{bmatrix} R \\ 0 \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix} = \begin{bmatrix} R_\lambda \\ 0 \\ 0 \end{bmatrix},$$

and hence (36) holds with

$$Q_\lambda = \begin{bmatrix} Q & \\ & \mathrm{I} \end{bmatrix} \bar{Q}_\lambda .$$

The advantage of this combined approach is that when the value of λ is changed in the root-finding algorithm, we need only recalculate $\bar{Q}_\lambda$ and not the Householder part of the factorization (38). This feature can save a lot of computation in the case of $m \gg n$, since just $\mathcal{O}(n^3)$ operations are required to recalculate $\bar{Q}_\lambda$ and R_λ for each value of λ, after the initial cost of $\mathcal{O}(mn^2)$ operations needed to calculate Q in (37).

§10.3 Algorithms for Nonlinear Least-Squares Problems

Least-squares problems are often poorly scaled. Some of the variables could have values of about 104, while other variables could be of order 10^{-6}. If such wide variations are ignored, the algorithms above may encounter numerical difficulties or produce solutions of poor quality. One way to reduce the effects of poor scaling is to use an ellipsoidal trust region in place of the spherical trust region defined above. The step is confined to an ellipse in which the lengths of the principal axes are related to the typical values of the corresponding variables. Analytically, the trust-region sub-problem becomes

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|D_k p\| \leqslant \Delta_k\,, \tag{39}$$

where D_k is a diagonal matrix with positive diagonal entries.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The solution of (39) satisfies an equation of the form

$$(J_k^{\mathrm{T}} J_k + \lambda D_k^2) p_k^{\mathrm{LM}} = -J_k^{\mathrm{T}} r_k \,, \tag{40}$$

and, equivalently, solves the linear least-squares problem

$$\min_p \left\| \begin{bmatrix} J_k \\ \sqrt{\lambda} D_k \end{bmatrix} p + \begin{bmatrix} r_k \\ 0 \end{bmatrix} \right\|^2 . \tag{41}$$

The diagonals of the scaling matrix D_k can change from iteration to iteration, as we gather information about the typical range of values for each component of x. If the variation in these elements is kept within certain bounds, then the convergence theory for the spherical case continues to hold, with minor modifications. Moreover, the technique described above for calculating R_λ needs no modification.

§10.3 Algorithms for Nonlinear Least-Squares Problems

For problems in which m and n are large and J is sparse, we may prefer to solve (31) or (39) approximately using the CG-Steihaug algorithm, Algorithm 7.2 from Chapter 7, with $J_k^{\mathrm{T}} J_k$ replacing the exact Hessian $\nabla^2 f_k$. Positive semi-definiteness of the matrix $J_k^{\mathrm{T}} J_k$ makes for some simplification of this algorithm, because negative curvature cannot arise. It is not necessary to calculate $J_k^{\mathrm{T}} J_k$ explicitly to implement Algorithm 7.2; the matrix-vector products required by the algorithm can be found by forming matrix-vector products with J_k and J_k^{T} separately.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Convergence of the Levenberg-Marquardt method**

It is not necessary to solve the trust-region problem

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \Delta_k \tag{31}$$

exactly in order for the Levenberg-Marquardt method to enjoy global convergence properties. The following convergence result can be obtained as a direct consequence of a theorem concerning the global convergence of trust-region method with trust-region radius modification given by Algorithm 4.1 in Chapter 4.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem

Consider solving the minimization problem

$$\min_{p\in\mathbb{R}^n} m_k(p) = f_k + g_k^{\mathrm{T}} p + \frac{1}{2} p^{\mathrm{T}} B_k p \quad \text{s.t.} \quad \|p\| \leqslant \gamma\Delta_k\,, \qquad (3')_4$$

using Algorithm 4.1, where $\gamma \geqslant 1$ is a fixed constant in $(3')_4$. Suppose that $\|B_k\| \leqslant \beta$ for some constant β, that f is bounded from below on the level set $S = \{x \,|\, f(x) \leqslant f(x_0)\}$ and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate solutions of $(3')_4$ satisfy the inequalities

$$m_k(0) - m_k(p_k) \geqslant c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right)$$

for some constant $c_1 \in (0, 1]$. We then have $\liminf_{k\to\infty} \|g_k\| = 0$. Moreover, if $\eta > 0$, then $\lim_{k\to\infty} \|g_k\| = 0$.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Theorem

Let $\eta \in (0, 1/4)$ in Algorithm 4.1 of Chapter 4, and suppose that the level set S defined by $S = \{x \,|\, f(x) \leqslant f(x_0)\}$ is bounded and that the residual functions r_j, $j = 1, 2, \cdots, m$ are Lipschitz continuously differentiable in a neighborhood $\mathcal{N}$ of S. Assume that for each k, the approximate solution p_k of

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \gamma \Delta_k \,,$$

where $\gamma \geqslant 1$, satisfies the inequality

$$m_k(0) - m_k(p_k) \geqslant c_1 \|J_k^{\mathrm{T}} r_k\| \min\left(\Delta_k, \frac{\|J_k^{\mathrm{T}} r_k\|}{\|J_k^{\mathrm{T}} J_k\|}\right) \tag{42}$$

for some constant $c_1 \in (0, 1]$. We then have that

$$\lim_{k\to\infty} \nabla f_k = \lim_{k\to\infty} J_k^{\mathrm{T}} r_k = 0 \,.$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

Proof.

The smoothness assumption on r_j and the compactness of S imply that we can choose a constant $\beta > 0$ such that $\|J_k^{\mathrm{T}} J_k\| \leqslant \beta$ for all iterates k. Also note that the objective f is bounded below (by zero). Hence, the assumptions of the theorem concerning the global convergence of trust-region method with trust-region radius modification given by Algorithm 4.1 in Chapter 4 are satisfied, and the result follows immediately. □

§10.3 Algorithms for Nonlinear Least-Squares Problems

As in Chapter 4, there is no need to calculate the right-hand side in the inequality

$$m_k(0) - m_k(p_k) \geqslant c_1 \|J_k^{\mathrm{T}} r_k\| \min\left(\Delta_k, \frac{\|J_k^{\mathrm{T}} r_k\|}{\|J_k^{\mathrm{T}} J_k\|}\right) \tag{42}$$

or to check it explicitly. Instead, we can simply require the decrease given by our approximate solution p_k of

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \Delta_k \tag{31}$$

to at least match the decrease given by the Cauchy point, which can be calculated inexpensively in the same way as in Chapter 4. If we use the iterative CG-Steihaug approach, Algorithm 7.2, the condition (42) is satisfied automatically for $c_1 = 1/2$, since the Cauchy point is the first estimate of p_k computed by this approach, while subsequent estimates give smaller values for the model function.

§10.3 Algorithms for Nonlinear Least-Squares Problems

As in Chapter 4, there is no need to calculate the right-hand side in the inequality

$$m_k(0) - m_k(p_k) \geqslant c_1 \|J_k^{\mathrm{T}} r_k\| \min\left(\Delta_k, \frac{\|J_k^{\mathrm{T}} r_k\|}{\|J_k^{\mathrm{T}} J_k\|}\right) \tag{42}$$

or to check it explicitly. Instead, we can simply require the decrease given by our approximate solution p_k of

$$\min_p \frac{1}{2}\|J_k p + r_k\|^2 \quad \text{subject to} \quad \|p\| \leqslant \Delta_k \tag{31}$$

to at least match the decrease given by the Cauchy point, which can be calculated inexpensively in the same way as in Chapter 4. If we use the iterative CG-Steihaug approach, Algorithm 7.2, the condition (42) is satisfied automatically for $c_1 = 1/2$, since the Cauchy point is the first estimate of p_k computed by this approach, while subsequent estimates give smaller values for the model function.

§10.3 Algorithms for Nonlinear Least-Squares Problems

The local convergence behavior of Levenberg-Marquardt is similar to the Gauss-Newton method. Near a solution x_* at which the first term of the Hessian $(\nabla^2 f)(x_*)$ in

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{m} r_j(x)(\nabla^2 r_j)(x) \tag{5}$$

dominates the second term, the model function in (31), the trust region becomes inactive and the algorithm takes Gauss-Newton steps, giving the rapid local convergence expression

$$\begin{aligned} &\|x_k + p_k^{\mathrm{GN}} - x_*\| \\ &\quad \leqslant \int_0^1 \left\| \left(J_k^{\mathrm{T}} J_k\right)^{-1} H\left(x_* + t(x_k - x_*)\right) \right\| \|x_k - x_*\| dt + \mathcal{O}(\|x_k - x_*\|^2) \\ &\quad \approx \left\| \left[J^{\mathrm{T}} J(x_*)\right]^{-1} H(x_*) \right\| \|x_k - x_*\| + \mathcal{O}(\|x_k - x_*\|^2)\,. \end{aligned} \tag{30}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Method for large-residual problems**

In large-residual problems, the quadratic model in (31) is an inadequate representation of the function f because the second-order part of the Hessian $\nabla^2 f(x)$ is too significant to be ignored. In data-fitting problems, the presence of large residuals may indicate that the model is inadequate or that errors have been made in monitoring the observations. Still, the practitioner may need to solve the least-squares problem with the current model and data, to indicate where improvements are needed in the weighting of observations, modeling, or data collection.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Method for large-residual problems**

In large-residual problems, the quadratic model in (31) is an inadequate representation of the function f because the second-order part of the Hessian $\nabla^2 f(x)$ is too significant to be ignored. In data-fitting problems, the presence of large residuals may indicate that the model is inadequate or that errors have been made in monitoring the observations. Still, the practitioner may need to solve the least-squares problem with the current model and data, to indicate where improvements are needed in the weighting of observations, modeling, or data collection.

§10.3 Algorithms for Nonlinear Least-Squares Problems

- **Method for large-residual problems**

In large-residual problems, the quadratic model in (31) is an inadequate representation of the function f because the second-order part of the Hessian $\nabla^2 f(x)$ is too significant to be ignored. In data-fitting problems, the presence of large residuals may indicate that the model is inadequate or that errors have been made in monitoring the observations. Still, the practitioner may need to solve the least-squares problem with the current model and data, to indicate where improvements are needed in the weighting of observations, modeling, or data collection.

§10.3 Algorithms for Nonlinear Least-Squares Problems

On large-residual problems, the asymptotic convergence rate of Gauss-Newton and Levenberg-Marquardt algorithms is only linear – slower than the superlinear convergence rate attained by algorithms for general unconstrained problems, such as Newton or quasi-Newton.

If the individual Hessians $\nabla^2 r_j$ are easy to calculate, it may be better to ignore the structure of the least-squares objective and apply Newton's method with trust region or line search to the problem of minimizing f. Quasi-Newton methods, which attain a superlinear convergence rate without requiring calculation of $\nabla^2 r_j$, are another option. However, the behavior of both Newton and quasi-Newton on early iterations (before reaching a neighborhood of the solution) may be inferior to Gauss-Newton and Levenberg-Marquardt.

§10.3 Algorithms for Nonlinear Least-Squares Problems

On large-residual problems, the asymptotic convergence rate of Gauss-Newton and Levenberg-Marquardt algorithms is only linear – slower than the superlinear convergence rate attained by algorithms for general unconstrained problems, such as Newton or quasi-Newton. If the individual Hessians $\nabla^2 r_j$ are easy to calculate, it may be better to ignore the structure of the least-squares objective and apply Newton's method with trust region or line search to the problem of minimizing f. Quasi-Newton methods, which attain a superlinear convergence rate without requiring calculation of $\nabla^2 r_j$, are another option. However, the behavior of both Newton and quasi-Newton on early iterations (before reaching a neighborhood of the solution) may be inferior to Gauss-Newton and Levenberg-Marquardt.

§10.3 Algorithms for Nonlinear Least-Squares Problems

On large-residual problems, the asymptotic convergence rate of Gauss-Newton and Levenberg-Marquardt algorithms is only linear – slower than the superlinear convergence rate attained by algorithms for general unconstrained problems, such as Newton or quasi-Newton. If the individual Hessians $\nabla^2 r_j$ are easy to calculate, it may be better to ignore the structure of the least-squares objective and apply Newton's method with trust region or line search to the problem of minimizing f. Quasi-Newton methods, which attain a superlinear convergence rate without requiring calculation of $\nabla^2 r_j$, are another option. However, the behavior of both Newton and quasi-Newton on early iterations (before reaching a neighborhood of the solution) may be inferior to Gauss-Newton and Levenberg-Marquardt.

§10.3 Algorithms for Nonlinear Least-Squares Problems

Of course, we often do not know beforehand whether a problem will turn out to have small or large residuals at the solution. It seems reasonable, therefore, to consider hybrid algorithms, which would behave like Gauss-Newton or Levenberg-Marquardt if the residuals turn out to be small (and hence take advantage of the cost savings associated with these methods) but switch to Newton or quasi-Newton steps if the residuals at the solution appear to be large.

§10.3 Algorithms for Nonlinear Least-Squares Problems

There are a couple of ways to construct hybrid algorithms. One approach, due to Fletcher and Xu (see Fletcher [101]), maintains a sequence of positive definite Hessian approximations B_k. If the Gauss-Newton step from x_k reduces the function f by a certain fixed amount, then this step is taken and B_k is overwritten by $J_k^{\mathrm{T}} J_k$. Otherwise, a direction is computed using B_k, and the new point x_{k+1} is obtained by performing a line search. In either case, a BFGS-like update is applied to B_k to obtain a new approximation B_{k+1}. In the zero-residual case, the method eventually always takes Gauss-Newton steps (giving quadratic convergence), while it eventually reduces to BFGS in the nonzero-residual case (giving superlinear convergence).

§10.3 Algorithms for Nonlinear Least-Squares Problems

A second way to combine Gauss-Newton and quasi-Newton ideas is to maintain approximations to just the second-order part of the Hessian. In other words, we design a sequence of matrices S_k that approximate only the summation term $\sum_{j=1}^{m} r_j(x_k)\nabla^2 r_j(x_k)$ in (5), and then use the overall Hessian approximation

$$B_k = J_k^{\mathrm{T}} J_k + S_k$$

in a trust-region or line search model for calculating the step p_k. Updates to S_k are devised so that the approximate Hessian B_k mimics the behavior of the corresponding exact quantities over the step just taken. The update formula is based on a secant equation, which arises also in the context of unconstrained minimization in Chapter 6.

§10.3 Algorithms for Nonlinear Least-Squares Problems

A second way to combine Gauss-Newton and quasi-Newton ideas is to maintain approximations to just the second-order part of the Hessian. In other words, we design a sequence of matrices S_k that approximate only the summation term $\sum_{j=1}^{m} r_j(x_k)\nabla^2 r_j(x_k)$ in (5), and then use the overall Hessian approximation

$$B_k = J_k^{\mathrm{T}} J_k + S_k$$

in a trust-region or line search model for calculating the step p_k. Updates to S_k are devised so that the approximate Hessian B_k mimics the behavior of the corresponding exact quantities over the step just taken. The update formula is based on a secant equation, which arises also in the context of unconstrained minimization in Chapter 6.

§10.3 Algorithms for Nonlinear Least-Squares Problems

In the present instance, there are a number of different ways to define the secant equation and to specify the other conditions needed for a complete update formula for S_k. In the probably best-known algorithm due to Dennis, Gay and Welsch, the secant equation

$$S_{k+1}(x_{k+1} - x_k) = \text{the right-hand side}$$

is motivated in the following way. Ideally, S_{k+1} should be a close approximation to the exact second-order term at $x = x_{k+1}$; that is,

$$S_{k+1} \approx \sum_{j=1}^{m} r_j(x_{k+1}) \nabla^2 r_j(x_{k+1}) \,.$$

so the right-hand side of the secant equation should approximate

$$\sum_{j=1}^{m} r_j(x_{k+1}) \nabla^2 r_j(x_{k+1}) (x_{k+1} - x_k) \,.$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

To avoid the computation of $\nabla^2 r_j$, we replace each of them with an approximation $(B_j)_{k+1}$ and impose the condition that $(B_j)_{k+1}$ should mimic the behavior of its exact counterpart $\nabla^2 r_j$ over the step just taken:

$$\begin{aligned}(B_j)_{k+1}(x_{k+1} - x_k) &= \nabla r_j(x_{k+1}) - \nabla r_j(x_k) \\ &= (\text{row } j \text{ of } (J(x_{k+1}))^{\mathrm{T}} - (\text{row } j \text{ of } (J(x_k))^{\mathrm{T}}.\end{aligned}$$

This condition leads to a secant equation on S_{k+1}, namely,

$$\begin{aligned}S_{k+1}(x_{k+1} - x_k) &= \sum_{j=1}^{m} r_j(x_{k+1})(B_j)_{k+1}(x_{k+1} - x_k) \\ &= \sum_{j=1}^{m} r_j(x_{k+1})\left[(\text{row } j \text{ of } J(x_{k+1}))^{\mathrm{T}} - (\text{row } j \text{ of } (J(x_k))^{\mathrm{T}}\right] \\ &= J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_{k+1}\,.\end{aligned}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

To avoid the computation of $\nabla^2 r_j$, we replace each of them with an approximation $(B_j)_{k+1}$ and impose the condition that $(B_j)_{k+1}$ should mimic the behavior of its exact counterpart $\nabla^2 r_j$ over the step just taken:

$$\begin{aligned}(B_j)_{k+1}(x_{k+1} - x_k) &= \nabla r_j(x_{k+1}) - \nabla r_j(x_k) \\ &= (\text{row } j \text{ of } (J(x_{k+1}))^{\mathrm{T}} - (\text{row } j \text{ of } (J(x_k))^{\mathrm{T}}.\end{aligned}$$

This condition leads to a secant equation on S_{k+1}, namely,

$$\begin{aligned}S_{k+1}(x_{k+1} - x_k) &= \sum_{j=1}^{m} r_j(x_{k+1})(B_j)_{k+1}(x_{k+1} - x_k) \\ &= \sum_{j=1}^{m} r_j(x_{k+1})\left[(\text{row } j \text{ of } J(x_{k+1}))^{\mathrm{T}} - (\text{row } j \text{ of } (J(x_k))^{\mathrm{T}}\right] \\ &= J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_{k+1}\,.\end{aligned}$$

§10.3 Algorithms for Nonlinear Least-Squares Problems

As usual, this condition does not completely specify the new approximation S_{k+1}. Dennis, Gay, and Welsch add requirements that S_{k+1} be symmetric and that the difference $S_{k+1} - S_k$ from the previous estimate S_k be minimized in a certain sense, and derive the following update formula:

$$S_{k+1} = S_k + \frac{(y^\sharp - S_k s)y^{\mathrm{T}} + y(y^\sharp - S_k s)^{\mathrm{T}}}{y^{\mathrm{T}} s} - \frac{(y^\sharp - S_k s)^{\mathrm{T}} s}{(y^{\mathrm{T}} s)^2} yy^{\mathrm{T}}, \quad (43)$$

where

$$s = x_{k+1} - x_k, \quad y = J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_k, \quad y^\sharp = J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_{k+1}.$$

Note that (43) would be identical to the DFP update for unconstrained minimization if $y = y^\sharp$. Dennis, Gay, and Welsch use their approximate Hessian $J_k^{\mathrm{T}} J_k + S_k$ in conjunction with a trust-region strategy, but a few more features are needed to enhance its performance.

§10.3 Algorithms for Nonlinear Least-Squares Problems

As usual, this condition does not completely specify the new approximation S_{k+1}. Dennis, Gay, and Welsch add requirements that S_{k+1} be symmetric and that the difference $S_{k+1} - S_k$ from the previous estimate S_k be minimized in a certain sense, and derive the following update formula:

$$S_{k+1} = S_k + \frac{(y^\sharp - S_k s)y^{\mathrm{T}} + y(y^\sharp - S_k s)^{\mathrm{T}}}{y^{\mathrm{T}} s} - \frac{(y^\sharp - S_k s)^{\mathrm{T}} s}{(y^{\mathrm{T}} s)^2} yy^{\mathrm{T}}, \quad (43)$$

where

$$s = x_{k+1} - x_k, \quad y = J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_k, \quad y^\sharp = J_{k+1}^{\mathrm{T}} r_{k+1} - J_k^{\mathrm{T}} r_{k+1}.$$

Note that (43) would be identical to the DFP update for unconstrained minimization if $y = y^\sharp$. Dennis, Gay, and Welsch use their approximate Hessian $J_k^{\mathrm{T}} J_k + S_k$ in conjunction with a trust-region strategy, but a few more features are needed to enhance its performance.

§10.3 Algorithms for Nonlinear Least-Squares Problems

One deficiency of its basic update strategy for S_k is that this matrix is not guaranteed to vanish as the iterates approach a zero-residual solution, so it can interfere with superlinear convergence. This problem is avoided by scaling S_k prior to its update; we replace S_k by $\tau_k S_k$ on the right-hand side of (43), where

$$\tau_k = \min\left(1, \frac{|s^{\mathrm{T}} y^\sharp|}{|s^{\mathrm{T}} S_k s|}\right).$$

A final modification in the overall algorithm is that the S_k term is omitted from the Hessian approximation when the resulting Gauss-Newton model produces a sufficiently good step.

§10.3 Algorithms for Nonlinear Least-Squares Problems

One deficiency of its basic update strategy for S_k is that this matrix is not guaranteed to vanish as the iterates approach a zero-residual solution, so it can interfere with superlinear convergence. This problem is avoided by scaling S_k prior to its update; we replace S_k by $\tau_k S_k$ on the right-hand side of (43), where

$$\tau_k = \min\left(1, \frac{|s^{\mathrm{T}} y^{\sharp}|}{|s^{\mathrm{T}} S_k s|}\right).$$

A final modification in the overall algorithm is that the S_k term is omitted from the Hessian approximation when the resulting Gauss-Newton model produces a sufficiently good step.

§10.3 Algorithms for Nonlinear Least-Squares Problems

One deficiency of its basic update strategy for S_k is that this matrix is not guaranteed to vanish as the iterates approach a zero-residual solution, so it can interfere with superlinear convergence. This problem is avoided by scaling S_k prior to its update; we replace S_k by $\tau_k S_k$ on the right-hand side of (43), where

$$\tau_k = \min\left(1, \frac{|s^{\mathrm{T}} y^{\sharp}|}{|s^{\mathrm{T}} S_k s|}\right).$$

A final modification in the overall algorithm is that the S_k term is omitted from the Hessian approximation when the resulting Gauss-Newton model produces a sufficiently good step.

§10.4 Orthogonal Distance Regression

In the example in Section 10.1 we assumed that no errors were made in noting the time at which the blood samples were drawn, so that the differences between the model $\varphi(x; t_j)$ and the observation y_j were due to inadequacy in the model or measurement errors in y_j. We assumed that any errors in the ordinates – the times t_j – are tiny by comparison with the errors in the observations. This assumption often is reasonable, but there are cases where the answer can be seriously distorted if we fail to take possible errors in the ordinates into account. Models that take these errors into account are known in the statistics literature as errors-in-variables models, and the resulting optimization problems are referred to as total least squares in the case of a linear model or as orthogonal distance regression in the nonlinear case.

§10.4 Orthogonal Distance Regression

In the example in Section 10.1 we assumed that no errors were made in noting the time at which the blood samples were drawn, so that the differences between the model $\varphi(x; t_j)$ and the observation y_j were due to inadequacy in the model or measurement errors in y_j. We assumed that any errors in the ordinates – the times t_j – are tiny by comparison with the errors in the observations. This assumption often is reasonable, but there are cases where the answer can be seriously distorted if we fail to take possible errors in the ordinates into account. Models that take these errors into account are known in the statistics literature as errors-in-variables models, and the resulting optimization problems are referred to as total least squares in the case of a linear model or as orthogonal distance regression in the nonlinear case.

§10.4 Orthogonal Distance Regression

We formulate this problem mathematically by introducing perturbations δ_j for the ordinates t_j, as well as perturbations ε_j for y_j, and seeking the values of these $2m$ perturbations that minimize the discrepancy between the model and the observations, as measured by a weighted least-squares objective function. To be precise, we relate the quantities t_j, y_j, δ_j, and ε_j by

$$y_j = \varphi(x; t_j + \delta_j) + \varepsilon_j, \quad j = 1, 2, \cdots, m, \tag{44}$$

and define the minimization problem as

$$\min_{x,\delta_j,\varepsilon_j} \frac{1}{2} \sum_{j=1}^{m} w_j^2 \varepsilon_j^2 + d_j^2 \delta_j^2 \quad \text{subject to (44).} \tag{45}$$

The quantities w_j and d_j are weights, selected either by the modeler or by some automatic estimate of the relative significance of the error terms.

§10.4 Orthogonal Distance Regression

It is easy to see how the term "orthogonal distance regression" originates when we graph this problem; see Figure 2 (in the next slide). If all the weights w_j and d_j are equal, then each term in the summation (45) is simply the shortest distance between the point (t_j, y_j) and the curve $\varphi(x; t)$ (plotted as a function of t). The shortest path between each point and the curve is orthogonal to the curve at the point of intersection.

§10.4 Orthogonal Distance Regression

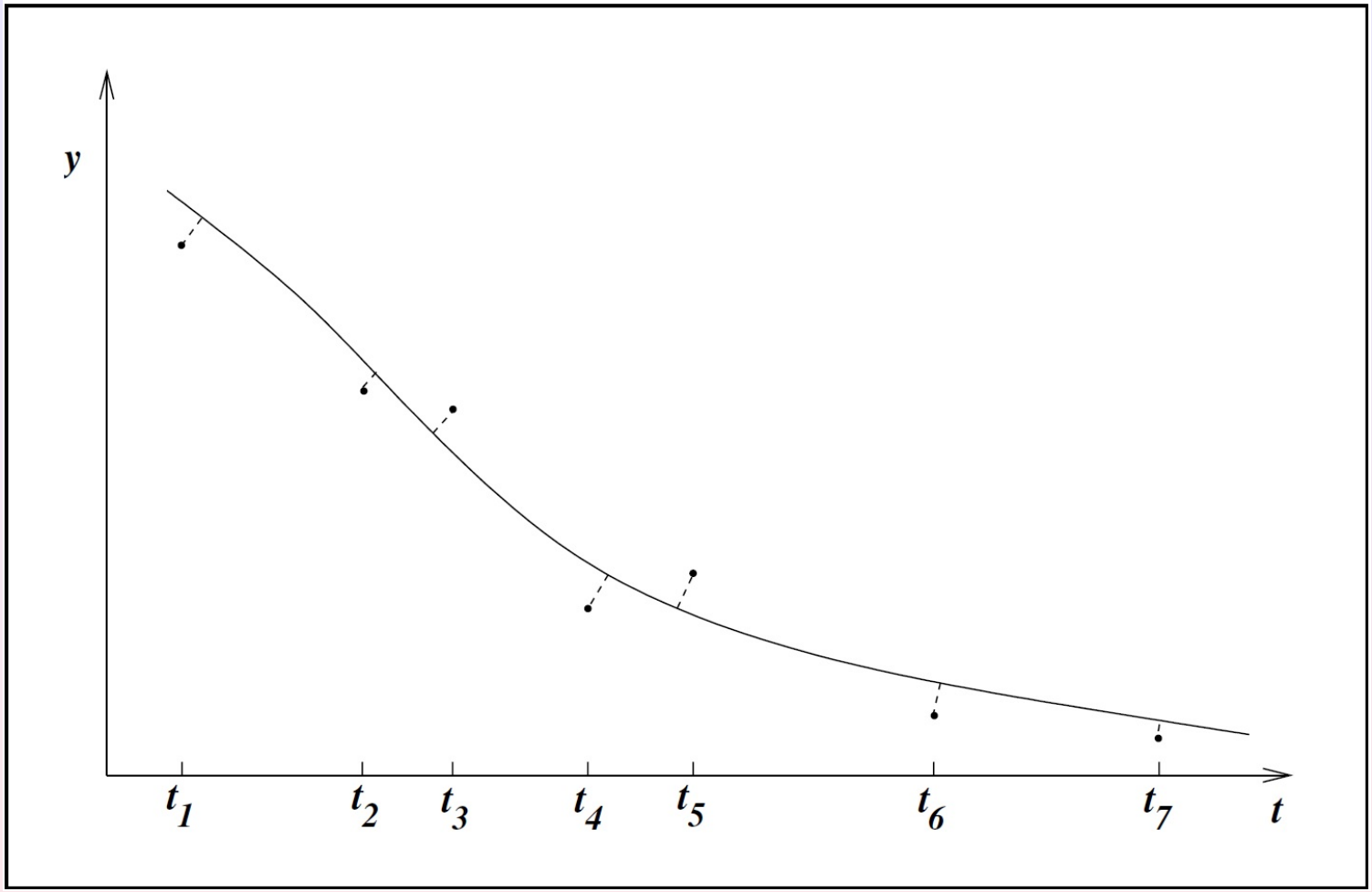

Figure 2: Orthogonal distance regression minimizes the sum of squares of the distance from each point to the curve.

§10.4 Orthogonal Distance Regression

Using the constraints (44) to eliminate the variables ε_j from (45), we obtain the unconstrained least-squares problem

$$\begin{aligned} \min_{x,\delta} F(x,\delta) &= \frac{1}{2}\sum_{j=1}^{m} w_j^2 \left[y_j - \varphi(x; t_j + \delta_j)\right]^2 + d_j^2 \delta_j^2 \\ &= \frac{1}{2}\sum_{j=1}^{2m} r_j^2(x,\delta) \end{aligned} \tag{46}$$

where $\delta = (\delta_1, \delta_2, \cdots, \delta_m)^{\mathrm{T}}$ and we have defined

$$r_j(x,\delta) = \begin{cases} w_j\left[\varphi(x; t_j + \delta_j) - y_j\right] & \text{if } j = 1, 2, \cdots, m, \\ d_{j-m}\delta_{j-m} & \text{if } j = m+1, \cdots, 2m. \end{cases} \tag{47}$$

Note that (46) is now a standard least-squares problem with $2m$ residuals and $m + n$ unknowns, which we can solve by using the techniques in this chapter.

§10.4 Orthogonal Distance Regression

A naive implementation of this strategy may, however, be quite expensive, since the number of parameters $(m + n)$ and the number of observations $(2m)$ may both be much larger than for the original problem. Fortunately, the Jacobian matrix for (46) has a special structure that can be exploited in implementing the Gauss-Newton or Levenberg-Marquardt methods. For instance, we have

$$\frac{\partial r_j}{\partial \delta_i} = \frac{\partial [\varphi(t_j + \delta_j; x) - y_j]}{\partial \delta_i} = 0 \quad \text{for } i, j = 1, 2, \cdots, m, i \neq j,$$

and

$$\frac{\partial r_j}{\partial x_i} = 0 \quad \text{for } j = m + 1, \cdots, 2m, i = 1, 2, \cdots, n.$$

Additionally, we have for $j = 1, 2, \cdots, m$ and $i = 1, 2, \cdots, m$ that

$$\frac{\partial r_{m+j}}{\partial \delta_i} = \begin{cases} d_j & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

§10.4 Orthogonal Distance Regression

A naive implementation of this strategy may, however, be quite expensive, since the number of parameters $(m + n)$ and the number of observations $(2m)$ may both be much larger than for the original problem. Fortunately, the Jacobian matrix for (46) has a special structure that can be exploited in implementing the Gauss-Newton or Levenberg-Marquardt methods. For instance, we have

$$\frac{\partial r_j}{\partial \delta_i} = \frac{\partial[\varphi(t_j + \delta_j; x) - y_j]}{\partial \delta_i} = 0 \quad \text{for } i, j = 1, 2, \cdots, m, i \neq j,$$

and

$$\frac{\partial r_j}{\partial x_i} = 0 \quad \text{for } j = m + 1, \cdots, 2m, i = 1, 2, \cdots, n.$$

Additionally, we have for $j = 1, 2, \cdots, m$ and $i = 1, 2, \cdots, m$ that

$$\frac{\partial r_{m+j}}{\partial \delta_i} = \begin{cases} d_j & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

§10.4 Orthogonal Distance Regression

Hence, we can partition the Jacobian of the residual function r defined by (47) into blocks and write

$$J(x,\delta) = \begin{bmatrix} \widehat{J} & V \\ 0 & D \end{bmatrix}, \tag{48}$$

where V and D are $m \times m$ diagonal matrices and $\widehat{J}$ is the $m \times n$ matrix of partial derivatives of the functions $w_j \varphi(t_j + \delta_j; x)$ with respect to x. Boggs, Byrd, and Schnabel [30] apply the Levenberg-Marquardt algorithm to (46) and note that block elimination can be used to solve the sub-problems

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p = -J^{\mathrm{T}}r \tag{33}$$

and

$$\min_p \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix} p + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|^2 \tag{35}$$

efficiently.

§10.4 Orthogonal Distance Regression

Hence, we can partition the Jacobian of the residual function r defined by (47) into blocks and write

$$J(x,\delta) = \begin{bmatrix} \widehat{J} & V \\ 0 & D \end{bmatrix}, \tag{48}$$

where V and D are $m \times m$ diagonal matrices and $\widehat{J}$ is the $m \times n$ matrix of partial derivatives of the functions $w_j \varphi(t_j + \delta_j; x)$ with respect to x. Boggs, Byrd, and Schnabel [30] apply the Levenberg-Marquardt algorithm to (46) and note that block elimination can be used to solve the sub-problems

$$(J^{\mathrm{T}}J + \lambda \mathrm{I})p = -J^{\mathrm{T}}r \tag{33}$$

and

$$\min_p \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda}\mathrm{I} \end{bmatrix} p + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|^2 \tag{35}$$

efficiently.

§10.4 Orthogonal Distance Regression

Given the partitioning (48), we can partition the step vector p and the residual vector r accordingly as

$$p = \begin{bmatrix} p_x \\ p_\delta \end{bmatrix}, \quad r = \begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \end{bmatrix},$$

and write the normal equations (33) in the partitioned form

$$\begin{bmatrix} \hat{J}^{\mathrm{T}}\hat{J} + \lambda \mathrm{I} & \hat{J}^{\mathrm{T}}V \\ V\hat{J} & V^2 + D^2 + \lambda \mathrm{I} \end{bmatrix} \begin{bmatrix} p_x \\ p_\delta \end{bmatrix} = - \begin{bmatrix} \hat{J}^{\mathrm{T}}\hat{r}_1 \\ V\hat{r}_1 + D\hat{r}_2 \end{bmatrix}.$$

Since the lower right sub-matrix $V^2 + D^2 + \lambda \mathrm{I}$ is diagonal, it is easy to eliminate p_δ from this system and obtain a smaller $n \times n$ system to be solved for p_x alone. The total cost of finding a step is only marginally greater than for the $m \times n$ problem arising from the standard least-squares model.

§10.4 Orthogonal Distance Regression

Given the partitioning (48), we can partition the step vector p and the residual vector r accordingly as

$$p = \begin{bmatrix} p_x \\ p_\delta \end{bmatrix}, \quad r = \begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \end{bmatrix},$$

and write the normal equations (33) in the partitioned form

$$\begin{bmatrix} \hat{J}^{\mathrm{T}}\hat{J} + \lambda \mathrm{I} & \hat{J}^{\mathrm{T}}V \\ V\hat{J} & V^2 + D^2 + \lambda \mathrm{I} \end{bmatrix} \begin{bmatrix} p_x \\ p_\delta \end{bmatrix} = - \begin{bmatrix} \hat{J}^{\mathrm{T}}\hat{r}_1 \\ V\hat{r}_1 + D\hat{r}_2 \end{bmatrix}.$$

Since the lower right sub-matrix $V^2 + D^2 + \lambda \mathrm{I}$ is diagonal, it is easy to eliminate p_δ from this system and obtain a smaller $n \times n$ system to be solved for p_x alone. The total cost of finding a step is only marginally greater than for the $m \times n$ problem arising from the standard least-squares model.