LEAST SQUARES METHODS IN PHYSICS AND ENGINEERING

Åke Björck
Department of Mathematics
Linköping University, Sweden

Lectures given in the
Academic Training Programme of CERN
1980-1981

GENEVA
1981
ABSTRACT

These lectures deal with numerical methods representing the state of the art, including the available computer software. First a brief background of basic matrix factorizations is given. Dense linear problems are then treated, including methods for updating the solution when rows and variables are added or deleted. Special consideration is given to weighted least squares problems and constrained problems. Regularization of ill-posed problems are briefly surveyed. Sparse linear least squares problems are treated in some detail. Different ordering schemes, including ordering to block-angular form and nested dissection, and iterative methods are discussed. Finally a survey of methods for nonlinear least squares problems is given. The Gauss-Newton method is analyzed and its local convergence derived. Levenberg-Marquardt methods are discussed and different methods using second derivative information surveyed. Special methods are given for separable linear-nonlinear problems.
# CONTENTS

## CHAPTER I: DENSE LINEAR LEAST SQUARES PROBLEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.2 THE METHOD OF NORMAL EQUATIONS</td>
<td>2</td>
</tr>
<tr>
<td>1.3 THE QR DECOMPOSITION</td>
<td>6</td>
</tr>
<tr>
<td>1.4 RANK DETERMINATION AND THE SINGULAR VALUE DECOMPOSITION</td>
<td>9</td>
</tr>
<tr>
<td>1.5 SEQUENTIAL ORTHOGONAL REDUCTION AND UPDATING METHODS</td>
<td>11</td>
</tr>
<tr>
<td>1.6 WEIGHTED LEAST SQUARES PROBLEMS</td>
<td>13</td>
</tr>
<tr>
<td>1.7 CONSTRAINED LINEAR LEAST SQUARES PROBLEMS</td>
<td>15</td>
</tr>
<tr>
<td>1.8 REGULARIZATION OF ILL-POSED LEAST SQUARES PROBLEMS</td>
<td>17</td>
</tr>
<tr>
<td>1.9 ITERATIVE REFINEMENT OF LEAST SQUARES SOLUTIONS</td>
<td>19</td>
</tr>
<tr>
<td>1.10 SOFTWARE FOR DENSE LEAST SQUARES PROBLEMS</td>
<td>21</td>
</tr>
<tr>
<td>1.11 REFERENCES</td>
<td>23</td>
</tr>
</tbody>
</table>

## CHAPTER II: SPARSE LINEAR LEAST SQUARES PROBLEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 INTRODUCTION</td>
<td>25</td>
</tr>
<tr>
<td>2.2 THE METHOD OF NORMAL EQUATIONS</td>
<td>25</td>
</tr>
<tr>
<td>2.3 COLUMN ORDERINGS FOR SPARSE PROBLEMS</td>
<td>28</td>
</tr>
<tr>
<td>2.4 SEQUENTIAL ORTHOGONALIZATION METHOD</td>
<td>29</td>
</tr>
<tr>
<td>2.5 BLOCK ANGULAR FORM AND NESTED DISSECTION</td>
<td>32</td>
</tr>
<tr>
<td>2.6 ITERATIVE METHODS</td>
<td>35</td>
</tr>
<tr>
<td>2.7 SOFTWARE</td>
<td>38</td>
</tr>
<tr>
<td>2.8 REFERENCES</td>
<td>39</td>
</tr>
</tbody>
</table>
### CHAPTER III: NONLINEAR LEAST SQUARES PROBLEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>INTRODUCTION</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>THE GAUSS-NEWTON METHOD</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>LOCAL CONVERGENCE OF THE GAUSS-NEWTON METHOD</td>
<td>46</td>
</tr>
<tr>
<td>3.4</td>
<td>THE LEVENBERG-MARQUARDT METHOD</td>
<td>49</td>
</tr>
<tr>
<td>3.5</td>
<td>METHODS USING SECOND DERIVATIVE INFORMATION</td>
<td>51</td>
</tr>
<tr>
<td>3.6</td>
<td>CONSTRAINED PROBLEMS</td>
<td>53</td>
</tr>
<tr>
<td>3.7</td>
<td>SEPARABLE NONLINEAR PROBLEMS</td>
<td>57</td>
</tr>
<tr>
<td>3.8</td>
<td>SOFTWARE</td>
<td>60</td>
</tr>
<tr>
<td>3.9</td>
<td>REFERENCES</td>
<td></td>
</tr>
</tbody>
</table>
1. DENSE LINEAR LEAST SQUARES PROBLEMS

1.1 INTRODUCTION

The least squares problem is a computational problem of primary importance in physics and engineering. Applications occur in numerous fields, e.g. in parameter estimation and statistical data analysis. In this chapter we will study variations of the following basic problem.

Let $A$ be a given $m \times n$ matrix with $m \geq n$ and $b$ a given $m \times 1$ vector. Find a solution $x$ to the problem

$$\min_x \|b - Ax\|_2,$$

(1.1)

where $\| \cdot \|_2$ denotes the Euclidian vector norm. Thus we minimize

$$\sum_{i=1}^{m} r_i^2, \quad r = b - Ax,$$

where $r$ is the residual vector. We will call $x$ a least squares solution to the overdetermined system $Ax = b$.

We assume in this chapter that $A$ is a dense matrix, i.e. that most of its elements are nonzero. A matrix $A$ with relatively few nonzero elements is called sparse. We discuss methods for sparse linear least squares problems in chapter 2.

The linear least squares problem arises from the statistical model

$$Ax = b + \epsilon,$$

where $\epsilon$ is a stochastic variable with zero mean and covariance matrix $\mathbf{V}(\epsilon) = \sigma^2 \mathbf{I}$. If more generally $\mathbf{V}(\epsilon) = \mathbf{W}^2$, for some positive definite matrix $\mathbf{W}$, then the appropriate minimization problem is

$$\min_x (b - Ax)^T \mathbf{W}^2 (b - Ax).$$

Methods for such weighted least squares problems are treated in section 1.7.
In the least squares problem there is thus an underlying assumption that all errors are confined to the observation vector \( b \). This assumption is frequently unrealistic; e.g., modelling errors and instrument errors may introduce errors in the matrix \( A \). The method of total least squares of Golub and Van Loan [1980] is a technique which has been devised to compensate for such data errors. In total least squares one computes \( x \) as the solution to the problem

\[
\min_x ||(E|x)||_F, \quad \text{subject to} \quad (A + E)x = b + r,
\]

where \( ||\cdot||_F \) is the Frobenius matrix norm, viz.

\[
||B||_F^2 = \sum_{i,j} |b_{ij}|^2.
\]

We note that in the (classical) least squares problem (1.1) the solution \( x \) is unique if and only if the matrix \( A \) has linearly independent columns, or equivalently if \( \text{rank}(A) = n \). If \( \text{rank}(A) = r \leq n \), then to get a problem with a unique solution one determines \( x \) from

\[
\min_{x \in S} ||x||_2, \quad S = \{ x \mid ||b - Ax||_2 = \min \}.
\]

(1.2)

We will write the solution to (1.2) as

\[
x = A^+b,
\]

where the \( n \times n \) matrix \( A^+ \) is called the pseudoinverse of \( A \).

1.2 THE METHOD OF NORMAL EQUATIONS

We first give a convenient characterization of any vector \( x \), which minimizes \( ||b - Ax||_2 \). The range of the matrix \( A \) is defined by

\[
R(A) = \{ y \mid y = Ax \text{ for some } x \in \mathbb{R}^n \}.
\]

Any solution \( x \) to (1.1) then satisfies the orthogonality relation

\[
(b - Ax) \perp R(A).
\]

This can also be written

\[
A^T(b - Ax) = 0,
\]

yielding the well-known normal equations

\[
A^TAx = A^Tb.
\]

(1.3)
The matrix $A^\top A$ is symmetric and if $\text{rank}(A) = n$ also positive definite. Under appropriate statistical hypotheses the matrix

$$C = \sigma^2 (A^\top A)^{-1}, \quad \sigma^2 = \frac{\|b - Ax\|_2^2}{(m-n)}$$

(1.4)

can be shown to be an estimate of the covariance of the solution $x$.

We assume in the rest of this section that $\text{rank}(A) = n$. Then we can use the Cholesky algorithm, see Dahlquist and Björck [1974, p. 158], to compute the factorization

$$A^\top A = R^\top R,$$

(1.5)

where $R$ is an upper triangular matrix. The matrix $R$ in (1.5) is essentially uniquely determined.

**Lemma 1.1** To a given positive definite matrix $A^\top A$ there exists a unique upper triangular matrix $R$ with positive diagonal such that $A^\top A = R^\top R$.

**Proof:** The existence follows from the Cholesky algorithm. To show uniqueness assume that

$$A^\top A = R_1^\top R_1 = R_2^\top R_2.$$

Then it follows that

$$(R_2^{-1})^\top R_1 = R_2 R_1^{-1} = D \quad (D \text{ a diagonal matrix}),$$

since a matrix which is both upper and lower triangular is a diagonal.

But then $R_2 = DR_1$ and thus

$$R_1^\top R_1 = R_1^\top D^2 R_1,$$

and by equating the diagonal elements on both sides we see that $D^2 = I$.

But since both $R_1$ and $R_2$ have a positive diagonal, $D = I$ and $R_1 = R_2$.

The important conclusion from this lemma is that the $R$ computed by the Cholesky algorithm from $A^\top A$ must mathematically be the same upper triangle as computed by the orthogonalization methods described in section 1.3. We will call this matrix $R$ the Cholesky factor of $A^\top A$. 
In the method of normal equations the solution \( x \) is computed as follows. From (1.3) and (1.5) we have

\[
R^T(Rx) = A^Tb \iff R^Ty = A^Tb, \quad Rx = y,
\]

so \( x \) is obtained by solving two triangular systems of equations. This classical method of solution has the drawback that big errors may be introduced by rounding already when \( A^T A \) is computed. This is well illustrated by the following example.

**Example:** Assume that \( \varepsilon \) is a small number, which satisfies \( \varepsilon < \text{macheps}^{1/2} \).

(Here and in the following macheps denotes the relative precision of the computer arithmetic.) Let

\[
A = \begin{pmatrix}
1 & 1 & 1 \\
\varepsilon & 1 & 1 \\
\varepsilon & \varepsilon & 2
\end{pmatrix}.
\]

Then, since the computed value of \( (1+\varepsilon^2) \) equals 1, we will get

\[
A^T A = \begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{pmatrix}.
\]

This matrix is of rank one and we have lost all information from the last three rows in \( A \).

We now ask the question how well the solution \( x \) to (1.1) is determined from the data \( A \) and \( b \), i.e., if \( A \) and \( b \) are perturbed by the amounts \( \delta A \) and \( \delta b \), how much changes \( x \)? Such a perturbation analysis is more difficult to carry out for the least squares problem than for the case of a square system of linear equations \( Ax = b \). We will only state the result here and for a derivation we refer to Lawson and Hanson [1974, chap. 9].

The analysis given there was originally developed by Wedin [1973]. We first define the condition number \( \kappa(A) \) of a matrix by

\[
\kappa(A) = \max_{\|x\|_2 = 1} \frac{\|Ax\|_2}{\|x\|_2} \cdot \min_{\|x\|_2 = 1} \frac{\|Ax\|_2}{\|x\|_2}. \tag{1.6}
\]
Lemma 1.2. Let the matrix $A$ be perturbed to $A + \delta A$ and the vector $b$ to $b + \delta b$, where
\[ ||\delta A||_2 \leq \varepsilon_1 ||A||_2, \quad ||\delta b||_2 \leq \varepsilon_2 ||b||_2. \]

Denote the solution to the corresponding perturbed problem by $x + \delta x$.

Then if $\text{rank}(A) = n$ and $\kappa \varepsilon < 1$
\[ ||\delta x||_2 \leq \frac{\kappa}{1 - \kappa \varepsilon_1} \left\{ \varepsilon_1 ||x||_2 + \kappa \varepsilon_1 ||A||_2 + \varepsilon_2 ||A||_2 \right\}. \quad (1.7) \]

From (1.7) we first note that for a consistent problem, i.e. when $r = b - Ax = 0$, the condition number for computing $x$ is $\kappa(A)$. However, for large residual problems the second term in (1.7) will dominate and this is proportional to $\kappa^2(A)$. It is seen that for this second term to be negligible is that
\[ ||r||_2 \ll \kappa^{-1} ||A||_2 ||x||_2, \quad (1.8) \]
and then $\kappa(A)$ will be the relevant condition number for computing $x$.

However, if we explicitly form the normal equations (1.3) we will introduce perturbations in $x$ proportional to $\kappa^2(A)$, since $\kappa(A^TA) = \kappa^2(A)$. Thus the method of normal equations is not in general a stable method for solving the least squares problem. Therefore we consider in the next section methods based on orthogonal transformations.

It should be emphasized that the estimate (1.7) is relevant only if the matrix $A$ is properly scaled. Note that any scaling $D_1AD_2$ of $A$, where $D_1$ and $D_2$ are diagonal matrices, induces the same scaling $D_1\delta A D_2$ of the error matrix. Consequently one often recommends (LINPACK Users' Guide - Dongarra et al. [1979]) the following scaling strategy:

"Estimate the absolute size of the errors in the matrix $A$ and then scale so that as nearly as possible all estimates are equal."

However, for the least squares problem row scalings are in general not allowed, since they represent an impermissible change in the problem. The implication of this will be further discussed in connection with weighted least squares problems in section 1.7.
1.3 THE QR DECOMPOSITION

Let $Q$ be an orthogonal $m \times m$ matrix, i.e. $Q^T Q = QQ^T = I$. Then, since orthogonal transformations preserve the Euclidian length, we have the basic relation

$$\|b - Ax\|_2 = \|Q^T b - Q^T Ax\|_2.$$  

We can thus minimize the right-hand side instead. The idea is now to choose $Q$ so that $Q^T A$ gets a simple form.

**Lemma 1.3** Let $A$ be an $m \times n$ matrix of rank $r$. Then there exists an orthogonal matrix $Q$ and a permutation matrix $E$ so that

$$Q^T A E = \begin{pmatrix} R & 0 \\ 0 & S \end{pmatrix},$$  

where $R$ is an $r \times r$ upper triangular and nonsingular matrix.

We note that if $\text{rank}(A) = n$, then we can take $E = I$ and by uniqueness (lemma 1.1) $R$ must be the Cholesky factor of $A^T A$. To prove lemma 1.3, we will construct the orthogonal matrix $Q^T$ as a product of elementary orthogonal matrices of the following type.

**Definition:** We call $R = R(\phi)$ a Givens' rotation (or plane rotation), where

$$R(\phi) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \cos \phi, \ s = \sin \phi.$$  

Givens' rotations are extremely useful and play an important part in many matrix algorithms. The basic use of these rotations is illustrated in the following lemma.

**Lemma 1.4** Given a vector $v = (a, \beta)^T$, there is a plane rotation such that

$$R(\phi) \begin{pmatrix} a \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \end{pmatrix}.$$  

**Proof:** Take

$$c = a/\mu, \ s = \beta/\mu, \ \mu = (a^2 + \beta^2)^{1/2}.$$  

First define the condition number $\kappa(A)$ of a matrix by

$$\kappa(A) = \max_{\|x\|_2 = 1} \frac{\|Ax\|_2}{\min_{\|x\|_2 = 1} \|Ax\|_2}.$$  

(1.6)
We can imbed the Givens' rotation $R(\phi)$ in a unit matrix $I_m$ of order $m$ in the following way, to get a rotation $R_{ij}(\phi)$ in the plane $(i,j), j > i$,

\[
R_{ij}(\phi) = \begin{pmatrix} i & j & \cdots & \cdots & \cdots \\
\vdots & \vdots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \ddots & \ddots & \ddots \\
i & j & \cdots & \cdots & \cdots
\end{pmatrix}.
\]

(1.12)

A sequence of such transformations can be used to perform numerically the reduction of $A$ to the upper trapezoidal form of (1.9). The zeros are introduced columnwise from left to right, using lemma 1.4 to determine the appropriate rotation angle. A typical step is illustrated below.

**Example:** To zero elements in the $k$:th column below the main diagonal rotations are performed in the sequence of planes $(k,k+1), (k,k+2), \ldots, (k,m)$, where $A$ is an $m \times n$ matrix. (Below $k = 3$, $m = 6$, $n = 5$.)

\[
\begin{array}{ccc}
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\end{array}
\]

$x = \text{nonzero element} \\
m = \text{element being zeroed}$

Note that to end up with a matrix of the form (1.9) we have to permute a column with at least one nonzero element in rows $k$ to $m$ before the $k$:th step in the reduction. If no such column exists we have already achieved upper trapezoidal form.

From the factorization (1.9) we find $r = \text{rank}(A)$ and get a permutation of the columns in $A$

\[
AE = \{A_1, A_2\}
\]

such that

$A_1$ has linearly independent columns

\[
A_2 = A_1 R^{-1} E.
\]
If the rotations used to reduce A are also applied to the right-hand side b, i.e. we also compute

\[ Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad c_1 = \text{r} \times 1 \text{ matrix}, \]

then the least squares problem (1.1) is reduced to the form

\[ \min_x \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix} \right\|_2, \quad x = \tilde{x}, \quad \text{r} \times \text{r} \text{ matrix}. \tag{1.13} \]

Since x does not influence the residual in the last (m-r) components in (1.13) it immediately follows that a particular solution is

\[ \tilde{x}_1 = R^{-1} c_1, \quad \tilde{x}_2 = 0. \tag{1.14} \]

The residual norm of a solution is \( \| b - Ax \|_2 = \| c_2 \|_2 \). In general, the solution (1.13) is not the minimum norm solution (1.2).

By making further orthogonal transformations from the right we can obtain the factorization

\[ Q_1 A R_2 = \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix}, \quad R = \text{r} \times \text{r} \text{ matrix}. \]

From this factorization the minimum norm solution can be obtained as

\[ x = A^+ b = Q_2 y, \quad y = \begin{pmatrix} \tilde{R}^{-1} c_1 \\ 0 \end{pmatrix}. \]

We remark that in performing the reduction of A to the form (1.9) we could also have introduced the zeros in a row of A at a time. This alternative has advantages, especially for large sparse problems, and we return to this in chapter 2.

Golub [1965] was the first to describe in detail how to use orthogonal transformations to solve least squares problems. Instead of plane rotations (1.1) Golub used so-called Householder transformations, which are more efficient for small dense problems. For a description of Golub’s algorithm see also Businger and Golub [1965], Lawson and Hanson [1974, pp. 53-66] and Eldersveld [1976, pp. 22-27]. We have preferred here to present an algorithm using plane rotations, since this is more easily adapted to large problems and updating problems. Finally we remark that there exists a fast, square root free version of the plane rotations, which at the cost
of added complication makes them almost as fast as Householder transformations; see Lawson and Hanson [1974, pp. 60-62].

1.4 RANK DETERMINATION AND THE SINGULAR VALUE DECOMPOSITION

In the factorization (1.9) the columns of $A$ are often permuted to give a matrix $R$, which diagonal elements satisfy

$$r_{11} \geq r_{22} \geq \ldots \geq r_{nn}, \quad r_{kk}^2 \geq \sum_{i=1}^{k} r_{ij}^2, \quad j = k, \ldots, n.$$  

Note that these relations imply that if $r_{kk} = 0$, then rows $k, \ldots, n$ of $R$ are zero. If the factorization is normalized in this way, then we may estimate the condition number of $A$ by

$$\kappa(A) \geq \frac{r_{11}}{r_{nn}},$$

provided $\text{rank}(A) = n$. Empirically it has been found that the right-hand side above underestimates $\kappa(A)$ usually only by a factor of 2-10; see Dongarra et al. [1979, p. 9.25].

The estimate (1.15) says that if a small element appears in the diagonal of $R$, then $A$ must be ill-conditioned. However, there exist triangular matrices, which are highly ill-conditioned, although no diagonal element is small as the following example shows.

Example: Consider the matrices $R_n$ of dimension $n \times n$, where

$$R_5 = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}.$$  

If we take $x = (2^{-1}, 2^{-2}, \ldots, 2^{-n})^T$, then $R_n x = 2^{-n}(1,1,\ldots,1)^T$, and it follows that

$$\min_{||x||_2=1} ||R_n x||_2 < 2\sqrt{n} \cdot 2^{-\frac{n}{2}}.$$  

Therefore the matrix $R_n$ has a condition number of order $2^n$, and there exists a perturbation $E_n$, such that $R_n + E_n$ is singular, with $||E_n||_2 < 2\sqrt{n} \cdot 2^{-\frac{n}{2}}$.

The matrices $R_n$ in the example above do not satisfy the normalization condition. However, the example can be modified to show that there
exist triangular matrices of order \( n \), normalized as above, for which 
(1.15) underestimates \( \kappa(A) \) by a factor of order \( 2^n \). Even though such 
matrices do not seem to occur frequently in practice, this shows that 
the determination of the rank of \( A \) cannot always reliably be performed 
directly from the factorization (1.9).

To get a theoretically more satisfactory way of computing the rank 
of \( A \), we continue the reduction of \( A \) as follows. By applying further 
rotations from right and left to (1.15), the matrix \( R \) can be transformed 
into upper bidiagonal form. A typical stage in this reduction is shown 
below (see Eldén [1977]).

\[
\begin{array}{cccc}
5 & 3 & 1 \\
6 & 4 & 2 \\
7 & 2 & \end{array}
\]

\( \otimes \) = zeroed element

\( + \) = new nonzero introduced

The numbers indicate in which order the rotations are applied. In a column 
transformation an element in the first row is annihilated and a new non-
zero element created below the diagonal. The latter is zeroed by a row 
transformation. Finally, by a variant of the QR algorithm (see Golub and 
Reinsch [1970]) the bidiagonal matrix obtained can be further transformed 
into a diagonal matrix, which contains the singular values \( \sigma_i \) of \( A \).

The algorithm briefly outlined above computes the singular value 
decomposition of \( A \), which can be written

\[
A = U \Sigma V^T = (U_1, U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix}, \quad U_1 = m \times r \text{ matrix,} \\
V_1 = n \times r \text{ matrix,}
\]

where \( U \) and \( V \) are orthogonal matrices of dimensions \( m \times m \) and \( n \times n \) and

\[
\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r), \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.
\]

It is now easily shown that the condition number of \( A \) is

\[
\kappa(A) = \sigma_1 / \sigma_r,
\]

a definition which applies also when \( \text{rank}(A) = r < n \). Also \( (A + \delta) \) can be
singular only if $||E||_2 \geq \sigma_n$. Therefore the numerical rank of a matrix can be taken to be the number of singular values larger than a suitable tolerance. However, the singular values of a matrix $A$ change under scalings of the rows and columns of $A$. It is very important to note that the singular value decomposition is useful only if $A$ is properly scaled.

In terms of the singular value decomposition the pseudoinverse solution to the least squares problem can be written

$$x = V_{\Sigma^{-1}} U_{\Sigma^{-1}}^T b = \sum_{i=1}^{r} \sigma_i^{-1} (u_i^T b) v_i,$$

(1.18)

where $u_i$ and $v_i$ denote the $i$:th column in $U$ and $V$, respectively.

For a good example of how to use the singular value decomposition to analyze a least squares problem see Lawson and Hanson [1974, chapter 26].

1.5 SEQUENTIAL ORTHOGONAL REDUCTION AND UPDATING METHODS

In computing the QR decomposition in section 1.3 we introduced the zeros columnwise. This is not the only possible order, and we now describe an algorithm where the zeros are introduced instead in a rowwise fashion.

We assume that the matrix $A$ and the right-hand side $b$ are stored by rows and denote the $i$:th equation by

$$a_i^T x = b_i, \quad i = 1,2,\ldots,m.$$

(1.19)

The reduction proceeds a row at a time, and before the $i$:th step only the first $(i-1)$ equations have been reduced to trapezoidal form (or triangular form if $i < n$) as illustrated below.

Example: Take $n = 7$, $i = 4$.

\[
\begin{align*}
\begin{array}{cccccccc}
\text{already processed equations} & & & & & \text{new equation} \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times & \times & \times \\
\& \& \& \& \& \& \& \& \& \end{array}
\end{align*}
\]
Thus, the $i$:th equation is reduced in $j = \min(i-1,n)$ minor steps, $k = 1,2,\ldots,j$ as follows:

If $a_{ik} \neq 0$ then rotate in plane $(k,i)$ to zero $a_{ik}$.

These rotations are also applied to the elements in $b$, if these are available.

The rowwise reduction described has several advantages:

(i) We need storage only for the resulting upper triangle $R$ and two extra $n$-vectors.

(ii) For very large problems this method is easily modified to use auxiliary storage-like disk or tape, see Dahlquist and Björck [1974, p.169].

(iii) The method can be developed into an efficient method for sparse least squares problems (see chapter 2).

It is also immediately clear that this method solves the problem of updating a least squares solution when new equations are added. Removing a row (equation), sometimes called downdating, can also be done by this sequential algorithm. It can easily be shown that deleting the equation $a^T_kx = b_k$ is equivalent to adding the equation

\[ i_{k}^T x = i b_k, \quad i^2 = -1. \]

This can be performed using only real arithmetic as described in Lawson and Hanson [1974, pp.229--231]. However, downdating is not always a stable process!

We now discuss adding and deleting columns (variables), without recomputing the QR decomposition. Assume that the orthogonal matrix $Q$ has been saved in the decomposition

\[ Q^T(A \; b) = \begin{pmatrix} R_n & c_n \\ 0 & d_n \end{pmatrix}, \quad R_n = n \times n \text{ matrix}. \]

Adding a column, $A := (A \; a_{n+1})$ we first compute
\[ Q^T a_{n+1} = (r_{n+1}) \]

and then reduce, for \( c_n = n \times 1 \) matrix and \( c_{n+1} = (n+1) \times 1 \) matrix,

\[
\begin{pmatrix} R_n & r_{n+1} \\ a_{n+1} & d_n \end{pmatrix} \rightarrow \begin{pmatrix} R_{n+1} & c_{n+1} \\ d_{n+1} \end{pmatrix}
\]

as in the last step of the columnwise QR-decomposition method.

To delete a column in \( A \),

\[ A := (a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_n), \]

we can delete the corresponding column in \( R_n \) and then try to transform back to triangular form. The way this is done is illustrated in the following example:

Example: Take \( n = 6 \) and \( k = 2 \).

\[
\begin{array}{cccccc}
  x & x & x & x & x & x \\
  x & x & x & x & x & x \\
  0 & x & x & x & x & x \\
  0 & x & x & x & x & x \\
  0 & 0 & x & x & x & x \\
  0 & 0 & 0 & 0 & 0 & x
\end{array}
\]

We zero the elements appearing below the diagonal in columns \( k \) to \( n-1 \), using a sequence of rotations in the planes \((k,k+1), (k+1,k+2), \ldots, (n-1,n)\).

The updating algorithms described here are very convenient in practice and avoid recomputing the QR decomposition, which would cost about \( mn^2 \) operations.

1.6 WEIGHTED LEAST SQUARES PROBLEMS

As remarked in section 1.1, the equations in the system \( Ax = b \) should be weighted by the positive definite matrix \( W^{1/2} \), where \( W^{-1} \) is the covariance matrix of the errors in the right-hand side \( b \). We
then have to solve
\[
\min_{x} \| W(b - Ax) \|_2. \tag{1.20}
\]

We first consider the case when the errors are independent. Then
\( W \) becomes a diagonal matrix, and we assume that the equations have
been ordered so that
\[
W = \text{diag}(w_1, w_2, \ldots, w_m), \quad w_1 \geq w_2 \geq \cdots \geq w_m.
\]
In some applications the weights \( w_i \) may vary widely in size, e.g.
when certain linear combinations of the unknowns are known to high
accuracy. We call such problems, where \( w_1/w_m >> 1 \), stiff least squares
problems.

We first remark that the method of normal equations is in general
not well suited for solving stiff problems. Consider as an example the
weighted problem, where
\[
A = \begin{bmatrix} wA_1 \\ A_2 \end{bmatrix}, \quad w >> 1. \tag{1.21}
\]
In the method of normal equations we first compute
\[
A^T A = w^2 A_1^T A_1 + A_2^T A_2.
\]
If \( w \) is so large that to working precision \( w^2 + 1 = w^2 \), then we might
lose all information contained in \( A_2 \). [Note, however, that in the special
case when \( A_1 \) involves only rank\((A_1)\) variables we are still in good shape!]

The matrix (1.21) in this example has a large condition number. It

can be shown that
\[
\kappa(A) \geq w \frac{\sigma_{\max}(A_1)}{\sigma_{\min}(A_2)}.
\]
However, this is not a relevant condition number for the weighted least
squares problems when \( w >> 1 \), since then the rows in (1.21) are not well
scaled. Scaling the rows of \( A \) would, on the other hand, represent an im-
permissible change of the problem. This is the reason why stiff problems
must be solved with special care.
Stiff problems can be stably solved by the QR decomposition method described above, provided that the equations are sorted roughly in order of decreasing weight. This is an essential condition; a common misconception is that orthogonal transformations are always stable. (Note that we have assumed here that the matrix $A$ in (1.20) is well row scaled.) Another stable method for stiff problems is the Peters-Wilkinson method; see Peters and Wilkinson [1970] and Björck and Duff [1980].

When the covariance matrix is not a diagonal matrix, then one possible method is to compute the Cholesky factorization $W^T = L^T L$, and solve
\[
\min_{x} ||L^T b - (L^T A)x||_2.
\]
If $W$ is ill-conditioned, this is not a stable method, and we could instead use the eigendecomposition
\[
W^2 = F^T D^2 F, \quad D = \text{diag}(d_1, d_2, \ldots, d_m),
\]
and $F$ orthogonal. We apply the transformation $F$ also to $(A \vert b)$ and solve
\[
\min_{x} ||D(\tilde{b} - \tilde{A}x)||_2, \quad \tilde{b} = Fb, \quad \tilde{A} = FA.
\]
We recommend however, an alternative more efficient algorithm by Paige [1979].

1.7 CONSTRAINED LINEAR LEAST SQUARES PROBLEMS

Often we want some equations to be satisfied exactly, i.e. we want to solve a problem of the form
\[
\min_{x} ||b - Ax||_2, \quad \text{subject to} \quad Cx = d,
\]
where $C$ is a $p \times n$ matrix. An algorithm for solving this problem must check the constraints for consistency, and decide what to do if they are inconsistent. For this reason it is preferable to use a formulation which does not assume consistency. Therefore we reformulate the problem as
\[
\min_{x \in S} ||b - Ax||_2, \quad S = \{x | ||d - Cx||_2 = \text{min}\}. \quad (1.22)
\]
(This idea is due to Wedin, private communication.)

The constrained problem can be interpreted as the limiting case of a weighted problem, where the weights associated with the equations $Cx = d$ tend to infinity. Since we know that the method based on QR decomposition is stable for stiff problems, we now consider how this method behaves when some weights tends to infinity.

Example: We consider applying a plane rotation to two rows, where the first row has large elements:

$$
\begin{pmatrix}
1 & w^{-1} \\
-w^{-1} & 1
\end{pmatrix}
\begin{pmatrix}
w \\
w
\end{pmatrix}
= 
\begin{pmatrix}
w \\
0
\end{pmatrix}.
$$

We have assumed that $w$ is so large that to working accuracy $w^2 + 1 = w^2$. Then, the first row will not be changed, and the $(1,2)$-element in the (approximate) plane rotation can be put equal to zero. But this means that in the limit we are just doing an elimination, using the row with large norm as pivot row.

This means that the sequential orthogonalization method of section 1.5 can easily be modified to solve the problem (1.22). We apply this algorithm to the augmented system

$$
\begin{pmatrix}
C \\
A
\end{pmatrix}x = 
\begin{pmatrix}
d \\
b
\end{pmatrix},
$$

and formally assign the equations $Cx = d$ infinite weights. We proceed exactly as in section 1.5, except that when an element $a_{ik}$ in $A$ is to be zpered, then we do that by an elimination if $i$ corresponds to an equation from the constraints (i.e. an equation having infinite weight.) The difference from the unconstrained algorithm is thus very small! In a certain sense constraints simplify the problem, since rotations can be changed to eliminations.

Problems with inequality constraints

$$
\min \| b - Ax \|_2, \quad \text{subject to } \quad Cx \geq d,
$$

are in general much harder to solve. Indeed, just the problem of deciding if the inequalities $Cx \geq d$ have a feasible solution may not be easy.
Algorithms for problem (1.24) are rather complicated, and we will not describe any details here. Note that (1.24) can be reformulated as a quadratic programming problem but, for the same reason as we often wish to avoid using the normal equations, a special algorithm would be preferable. Such an algorithm, using stable orthogonal transformations, was given by Stoer [1971]. More recently an alternative algorithm has been developed by Haskell and Hanson [1978].

Frequently the inequalities in (1.24) have the simple form of box constraints, i.e. we have upper and lower bounds on \( x \), \( a \leq x \leq b \). Such special constraints are much easier to handle and special algorithms exist for this case. If only one-sided bounds on \( x \) are specified, then it is no restriction to assume that these are \( x \geq 0 \). We then have a non-negative least squares problem (NNLS). An algorithm for NNLS is described in Lawson and Hanson [1974]. Zimmermann [1977] gives an algorithm for problems with upper and lower bounds.

1.8 REGULARIZATION OF ILL-POSED LEAST SQUARES PROBLEMS

Many scientific reconstruction problems, e.g. in geophysics, atmospheric studies, and image reconstruction, give rise to ill-posed problems for partial differential or integral equations. For such problems the solution does not depend continuously on the data, i.e. they have an unbounded inverse. One example is given by a Fredholm integral equation of the first kind

\[
\int_a^b K(x,y) f(y) dy = g(x), \quad c \leq x \leq d,
\]

which is ill-posed if the kernel \( K(x,y) \) is a smooth function.

When an ill-posed problem is discretized, it will give rise to a system of equations which is highly ill-conditioned, see, for example, Hanson and Phillips [1975]. Moreover, the condition number will increase rapidly with the dimension of the system. Any attempt to solve this system without modification will give useless results since data and round-off error will cause the solution to be completely inaccurate.
Assume now that an ill-posed problem has been discretized, giving a linear system of equations
\[ K \Phi = \phi, \]
where \( K \) is an \( m \times n \) matrix. Then \( K \) will have a set of singular values \( \sigma_i(K) \), which are very close to zero. These small singular values usually correspond to the irregular part of the solution, which one wants to suppress. A convenient way to do this is to compute the singular value decomposition \( K = U \Sigma V^T \) and take a truncated expansion (1.18)
\[ f(\delta) = \sum_{\sigma_i > \delta} \sigma_i^{-1} (u_i^T \phi) v_i, \]
as a regularized solution. Here \( \delta \) is a cutoff, which has to be determined.

Another way to get a properly formulated problem is to take \( f = f(\mu) \), where \( f(\mu) \) is the solution to
\[ \min_{\mu} \left( \| \phi - K \mu \|_2^2 + \mu^T | \mu \|_2^2 \right), \]
\( \mu \) being a regularization parameter.

In both the methods above, a key problem is to determine the parameters \( \delta \) and \( \mu \), respectively. A very satisfactory way to do this is to use the generalized cross validation by Golub, Heath and Wahba [1979]. This requires (1.26) or (1.27) to be solved for a sequence of numerical values of \( \delta \) and \( \mu \), respectively, which obviously is simple to do for (1.26). For the regularization method (1.27) we proceed in two steps:

(i) First transform \( K \) to bidiagonal form,
\[ Q^T K P = \begin{pmatrix} \beta_1 & \beta_2 & \cdots & \beta_n \\ 0 & \alpha_1 & \cdots & \beta_{n-1} \\ 0 & 0 & \cdots & \alpha_n \end{pmatrix}, \]
and
\[ Q^T \phi = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \]

(ii) Put \( f = P \mu \) and rewrite (1.27) as
\[
\min \left\{ \| c - By \|_2^2 + \nu^2 \| y \|_2^2 \right\} = \min \left\{ \| t_0 \|_2 - \left( \begin{array}{c} \bar{a} \\ \bar{b} \end{array} \right) \right\}. 
\]

A solution \( y = y(u) \) to this problem can be computed in only \( O(n) \) operations and then \( f = By \) is obtained in \( O(n^2) \) operations (Eldén [1977]).

The two methods (1.26) and (1.27) often give very similar results. An advantage with the latter is that (1.27) can be generalized to

\[
\min_x \| g - Kf \|_2^2 + \nu^2 \| Lx \|_2^2, 
\]

where \( L \) is a general \( p \times n \) matrix. For many problems taking \( L \) to approximate some differential operator will give a more appropriate regularization than (1.27). The method outlined above can be modified to handle also this more general case; see Eldén [1977]. We also refer to the survey of methods for ill-posed problems given by Björck and Eldén [1979].

In many applications the matrix \( K \) will be large and sparse. Then the reduction of \( K \) to bidiagonal form by using plane rotations is not feasible, since these transformations will destroy sparsity. Then an iterative reduction to bidiagonal form can be achieved by a Lanczos type method described by Paige and Saunders [1978]. This will give after \( k \) iterations a bidiagonal \( k \times k \) matrix \( B_k \) and a vector \( c_k \) such that the problem of minimizing \( \| g - Kf \|_2 \) can be approximated by

\[
\min_{y_k} \| c_k - B_k y_k \|_2. 
\]

One can now apply regularization methods to this reduced system. Resulting algorithms are discussed in more detail in Björck [1980].

1.9 ITERATIVE REFINEMENT OF LEAST SQUARES SOLUTIONS

Often a cheap way to improve the accuracy of a computed solution - or to estimate its accuracy - is to use a technique called iterative refinement, described below. The solution \( x \) and corresponding residual
vector $r$ to the least squares problem (1.1) satisfies the symmetric (indefinite) system of equations

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

(1.28)

Now, let $x_0$ and $r_0$ be a computed solution and residual and introduce the residual vectors to the system (1.28)

$$
\begin{align*}
f_0 &= b - r_0 - Ax_0, \\
g_0 &= -A^Tr_0.
\end{align*}
$$

Assume now that we have computed the factorization

$$
Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}.
$$

Then there are two algorithms, depending on whether $Q$ has been saved or not, for computing corrections $\delta x_0$ and $\delta r_0$.

**I. $Q$ is available:**

$$
\begin{align*}
h_0 &= R^{-T}g_0, \\
Q^T r_0 &= \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}, \\
\delta r_0 &= Q \begin{pmatrix} \eta_0 \\ \nu_0 \end{pmatrix}, \\
\delta x_0 &= R^{-1}(u_0 - h_0).
\end{align*}
$$

**II. $Q$ is not available:**

$$
\begin{align*}
z_0 &= A^T r_0 - g_0, \\
\delta x_0 &= R^{-1}(R^{-T}z_0), \\
\delta r_0 &= f_0 - A x_0.
\end{align*}
$$

The refined solution is then taken to be

$$
x_1 = x_0 + \delta x_0, \\
r_1 = r_0 + \delta r_0.
$$

This improvement can obviously be iterated by computing residuals to (1.28) corresponding to $x_1$ and $r_1$, etc.

If the residuals $f_i$ and $g_i$, $i = 0, 1, \ldots$ are sufficiently accurately computed (e.g. using double precision accumulation of the inner products) then as shown by Björck [1978] the improvement in accuracy in the refined solution will be substantial. Especially the algorithm
using $Q$ is very satisfactory.

Sometimes it is argued that it is not worthwhile to improve a computed solution when $A$ and $b$ are not exactly known, since $x_0$ then will be as good a solution as the data warrant. However, for problems which are for some reason badly scaled, iterative refinement will correct a bad initial solution, even if the residuals $f_i$ and $g_i$ are computed in only single precision. The following example is taken from Björck [1978].

**Example:** Let $A$ be a $21 \times 6$ matrix with elements given by

$$A = (a_{ij}), \quad a_{ij} = (i-1)^{j-1}.$$ 

This corresponds to fitting a fifth-degree polynomial to observations at the points $0, 1, \ldots, 20$. The right-hand side was chosen so that the exact solution was $x = (1, 1, \ldots, 1)^T$ and the residual $r$ of moderate size. This problem was run, using double precision accumulation for the residuals, on a UNIVAC 1108, with 7.8 and 18.7 decimal digits accuracy in single and double precision, respectively.

The initial accuracy of the solution, using the QR method, was 3.4 digits. The refinement gave an improvement of about 6 digits per step and the final accuracy was 12.4 digits. With a right-hand side corresponding to the same solution $x$, but a zero residual, the initial and final accuracy were 2.9 and 18.7 decimals, respectively.

1.11 SOFTWARE FOR DENSE LEAST SQUARES PROBLEMS

Almost all numerical libraries like NAG, IMSL, etc., include subroutines for computing the Cholesky factorization (1.5) of a symmetric positive definite matrix $A^T A$, for QR factorization of a rectangular matrix $A$ (1.9) and for the singular value decomposition (1.16). Thus, all basic methods for solving dense linear least squares problems can easily be implemented. Therefore we here only list some of the more specialized packages and subroutines for linear least squares.
The monograph by Lawson and Hanson [1974] describes a package of subroutines for linear least squares problems. This package, called LSQ, is distributed by IMSL* and can be obtained for a nominal charge. The package includes some more special subroutines, e.g. for sequential solution of banded systems, for least squares with non-negativity constraints, and for least distance programming

$$\min ||x||_2, \text{ subject to } CX = d.$$ 

LINPACK (see Dongarra et al. [1979]) contains up-to-date subroutines for QR factorization and singular value decomposition, and the users' guide gives an excellent discussion of the use of these for solving least squares problems. LINPACK also includes subroutines for efficiently recomputing the Cholesky factor for A, when A is modified by addition of a row, deletion of a row, or permutation of its columns.

For inequality constrained problems we also mention the more recent subroutines by Haskell and Hanson [1978], Schittkowski and Stoer [1979], and Zimmermann [1977].

For computing accurate solutions of weighted linear least squares problems with equality constraints there is a subroutine by Wampler [1979]. This subroutine uses a modified Gram-Schmidt approach developed by Björck [1967] and includes the option of iterative refinement. This subroutine is now also included in the IMSL Library.

---

* IMSL
Sixth Floor-NBC Building
7500 Bellaire Blvd.
Houston, TX 77036 Phone: (713) 772-1927
1.11 REFERENCES


II. SPARSE LINEAR LEAST SQUARES PROBLEMS

2.1 INTRODUCTION

In this section we consider methods for solving the linear least squares problem

$$\min_x \| b - Ax \|^2_2 \quad (2.1)$$

when the matrix A is large and sparse, i.e. A only has a few non-zeros in each row. Such problems arise in a variety of applications, e.g. in photogrammetry, geodesy, surface fitting, etc.

A sparse least squares problem of spectacular size is described in Kolata [1978]. This is the problem of least squares adjustment of coordinates (latitudes and longitudes) of the geodetic stations comprising the North American Datum. This task, to be completed in 1983, consists of about six million equations and 400,000 unknowns (twice the number of stations). Since the choice of method and computer is still open it has spurred quite a lot of research into methods for sparse least squares problems. A detailed discussion of round-off error accumulation for this problem is given by Meisel [1980].

A survey of direct and iterative methods for sparse least squares problems is given in Björck [1976]. For direct methods see also Duff and Reid [1976]. In this chapter will be described some more recent developments.

2.2 THE METHOD OF NORMAL EQUATIONS

The main challenge in devising direct methods for sparse problems is to avoid fill-in, which is the term used to denote when a zero matrix element becomes nonzero. Since it is difficult to avoid fill-in when using orthogonalization methods, the method of normal equations is frequently used for sparse problems.

We discuss first the fill-in when the matrix $A^T A$ is formed. We have
\[
(A^T A)_{ij} = a_i^T a_j, \quad A = (a_1, a_2, \ldots, a_n),
\]

where we have denoted the \(i\)th column of \(A\) by \(a_i\), \(j = 1, 2, \ldots, n\).

If we ignore the possibility of cancellation in the inner products \(a_i^T a_j\), it follows that

\[
(A^T A)_{ij} \neq 0 \iff k \text{ such that } a_k \neq 0 \text{ and } a_k \neq 0.
\]

(See also Figure 2.1.)

\[
\begin{pmatrix}
  a_i \\
  a_j
\end{pmatrix}
\]

\[
\text{k:th}
\]

\[
\text{row}
\]

\[
\begin{pmatrix}
  x & x \\
  x & x
\end{pmatrix}
\]

Figure 2.1

From (2.1) it is immediately clear that the ordering of the rows of \(A\) will not affect either the number or location of nonzero elements in \(A\). The number of nonzeros will also not be affected by the ordering of the columns of \(A\). It also follows that even one dense row in \(A\) will cause \(A^T A\) to be a dense matrix.

Example:

\[
A = \begin{pmatrix}
  x & x & x & x & x \\
  x & x & x & x & x \\
  x & x & x & x & x
\end{pmatrix} \Rightarrow A^T A \text{ full!}
\]

Note that in this example the Cholesky factor \(R\) is again sparse. This follows from the uniqueness of \(R\) (cf. Lemma 1.1, page 3), since \(A\) is already an upper triangle.

Sparse problems, with only a few dense rows, can be treated by an updating technique. First the solution to the sparse problem (with the dense rows deleted) is computed, and then this solution is updated adding the dense rows. Note that the factorization of \(A\) is not updated - only the solution. Such methods are described in Björck and Duff [1980] and George and Heath [1980]. We assume here in the following that all rows of \(A\) are sparse.
For the following important class of matrices the fill-in when forming $A^T A$ will always be small.

Definition: $A$ is said to be a bandmatrix with bandwidth $w$ if

$$a_{ij} 
eq 0 \text{ and } a_{ik} 
eq 0 \implies |j - k| \leq w.$$

(See figure 2.2.)

$i$:th row: $x \ x \ \ldots \ x$

$\leftarrow w \rightarrow$

Figure 2.2

For this structure to have a practical significance we need to have $w \ll n$. It follows immediately that if $A$ has bandwidth $w$ then

$$(A^T A)_{ij} = 0 \quad \text{if} \quad |i - j| > w,$$

i.e. $A^T A$ has bandwidth $w$. But then also the Cholesky factor $R$ of $A^T A$ will have bandwidth $w$, and thus is sparse if $w \ll n$. Matrices $A$ of small bandwidth occur often in practice - the physical explanation may be that only variables close to each other are coupled by observations, as in some geodetic networks.

The bandwidth of a matrix $A$ is affected by the column ordering - row ordering will have no effect at all. Let $P$ be a permutation matrix and put $A = AP$. Then

$$A^T A = P^T (A^T A) P,$$

i.e. a reordering of columns in $A$ corresponds to a symmetric reordering of rows and columns of $A^T A$. There are well-known algorithms for finding a permutation $P$, which gives a small bandwidth, e.g. the reverse Cuthill-McKee algorithm; see George and Liu [1981]. Such orderings may be used when solving (2.1) with the method of normal equations. In the next section we discuss more generally column orderings which give rise to sparse Cholesky factors.
2.3 COLUMN ORDERINGS FOR SPARSE PROBLEMS

Since $A^T A$ is a symmetric, positive definite matrix, standard sparse matrix techniques (see George and Liu [1981]) can be used to obtain an ordering which will give minimal fill-in when computing the Cholesky factor. Note that since $A^T A$ is positive definite (we assume the columns of $A$ to be linearly independent) factorization based on any ordering will be numerically stable.

If we assume that no cancellation creating "accidental" zeros occurs when forming $A^T A$, then we can proceed as suggested by George and Heath [1980]. The basic steps are:

(i) Determine the structure (not the numerical values) of $B = A^T A$.

(ii) Apply an ordering algorithm to $B$, yielding a permutation matrix $P$ such that $B = P^T BF$ has a sparse Cholesky factor $R$.

(iii) Apply a "symbolic factorization" algorithm to find the structure of $R$.

For these steps there is high quality software available. We emphasize that in these steps we operate only on the (zero - nonzero) structure of $A$ so that no numerical computation is done. Note that this means we cannot detect cancellations when computing $R$ - we assume that $R + R^T$ is as dense as $A^T A$. The advantage of doing these symbolic computations is that the numerical computations can then be performed in a fixed data structure, where only nonzero elements are stored. We illustrate a possible data structure for storing a sparse matrix $R$ below in Figure 2.3.

$$
R = \begin{pmatrix}
1 & r_{12} & r_{14} \\
 r_{21} & 2 & r_{24} \\
 r_{31} & r_{32} & 3 \\
 r_{41} & r_{42} & r_{43} & 4 \\
 r_{51} & r_{52} & r_{53} & r_{54} & 5 \\
 r_{61} & r_{62} & r_{63} & r_{64} & r_{65} & 6 \\
\end{pmatrix}
$$
The elements in the integer vector XNZ point to the position of the first off-diagonal element in the rows.

An ordering algorithm for step (ii) above, which is effective and fast, is based on the minimum degree criterion. If we let \( v_i \) equal the number of nonzeros in the unreduced part of the \( i \)-th row, then if
\[
\min_j v_i, \quad k \leq i \leq n,
\]
the \( j \)-th row is chosen as the pivot row in the \( k \)-th step. (If there are ties we choose the smallest \( j \).) See also figure 2.4.

![Figure 2.4](image)

2.4 SEQUENTIAL ORTHOGONALIZATION METHOD

The sequential orthogonalization method described in section 1.5 has been implemented for sparse problems by George and Heath [1980]. The key observation is that there will always be space in the data structure for \( R \) (as determined in section 2.3) for rotating any row \( a^T \) of \( A = AF \) (see figure 2.5). When a row is processed, intermediate
nonzeros will, however, be created in this row. Therefore, it is convenient to use a full working vector of length \( n \) for the row being processed.

Although the ultimate sparsity of \( R \) will not depend on the order in which the rows are processed, the intermediate fill-in in the working row and therefore the overall number of operations for the reduction can vary considerably with different orderings. The following example illustrates that the fill-in can be drastically affected.

**Example:**
\[
A = \begin{pmatrix}
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
\end{pmatrix}
\]

\( PA = \begin{pmatrix}
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
x & x & x & x \\
\end{pmatrix}
\]

Cost for reducing \( A \): \( O(n^2) \)  \( PA \): \( O(mn^2) \)

George and Heath [1980] use the simple heuristic strategy of sorting the rows into increasing order with respect to the maximum column subscript (in the permuted column order of \( A \)), which worked well on the problems considered by them. Note that this is a good ordering for matrices of banded form. More research is needed on optimal row-orderings, however. We also stress, that for stability reasons, rows with large norms must be processed first.

George, Heath and Plemmons [1980] have given an implementation of the method described above using auxiliary storage. Their algorithm uses a method of partitioning and an algorithm for handling external files containing intermediate results. This is important since the storage
needed for large problems often exceeds even the size of virtual storage and so cannot be handled by a paging algorithm. Some extensions of this algorithm are given by Heath [1981].

For sparse problems the computation of the covariance matrix \((R^T R)^{-1}\) by the standard algorithm can be very time consuming. A more efficient algorithm for computing elements in \((R^T R)^{-1}\) associated with nonzero elements in \(R\) is given by Golub and Flemmonds [1980]. (Note that this includes the diagonal elements, giving the variances of \(x\).)

For dense least squares an advantage with the sequential orthogonalization algorithm is that it immediately allows for the updating of the solution when new equations are added. For the sparse case, new rows can be added only provided they fit into the data structure for \(R\).

To treat dense rows, which have been kept out to avoid fill-in, one can update only the solution - not the factorization. We outline here this procedure, following the description by George and Heath [1980]. We wish to solve

\[
\min_x \|\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} - \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} x \|_2^2, \quad A_1 \in \mathbb{R}^{m_1 \times n}, \quad A_2 \in \mathbb{R}^{m_2 \times n},
\]

where we assume for simplicity that \(A_1\) has full rank equal to \(n\). Denote by \(y\) the solution to the original problem (excluding the equations \(A_2 x = b_2\)), and let

\[
Q^T A_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad R_1 \in \mathbb{R}^{n \times n} \text{ matrix}.
\]

We want to compute the correction \(z = x - y\), and do that in two steps.

(i) Compute \(r_2(y) = b_2 - A_2 y\) and put \(v = r_2(y) - A_2 R_1^{-1} u\), \(u = R_1 z\).

(ii) Let \(C = (A_2 R_1^{-1} I)\) and solve

\[
\min \|\begin{pmatrix} u \\ v \end{pmatrix}\|_2^2, \text{ subject to } C \begin{pmatrix} u \\ v \end{pmatrix} = r_2(y).
\]

This is a minimum length problem, which if \(m_2\) is small is
easily solved as a small dense problem. We finally get
\[ z = R^{-1}Q_x u, \quad x = y + z. \]

2.5 BLOCK ANGULAR FORM AND NESTED DISSECTION

In this section we discuss two special column ordering methods, which are of interest for sparse problems. A special structure, which occurs in some applications is the block angular form

\[
A = \begin{pmatrix}
A_{11} & A_{1,m+1} \\
A_{22} & A_{2,m+1} \\
& \ddots \ & \ddots \\
& & A_{m\times m} & A_{m,m+1}
\end{pmatrix}, \quad X = \begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{m+1}
\end{pmatrix}
\]

where each \( A_{ii} \) is a rectangular matrix. Thus, the variables \( x_i \) corresponding to the block \( A_{ii} \) are only coupled to the variables \( x_{i+1}, i \leq m \).

This form sometimes arises naturally from the problem, see, for example, the photogrammetry problem considered by Golub, Lark and Pageano [1979]. It is however of more general interest because Weil and Kettler [1971] have given a heuristic algorithm for permuting a general sparse matrix into this form.

Problems of block angular form can efficiently be solved by the following algorithm:

**Solution algorithm for block angular problems**

(i) Reduce each diagonal block to triangular form by orthogonal transformations

\[
Q_i(A_{ii} A_{i,m+1} | b_i) = \begin{pmatrix}
R_i & S_i & c_i \\
0 & T_i & d_i
\end{pmatrix}, \quad i = 1, 2, \ldots, m.
\]

(The corresponding block in the last block column of \( A \) and the righthand side also take part.) Note that these computations can be done in parallel.

rows into increasing order with respect to the maximum column subscript (in the permuted column order of \( A \)), which worked well on the problems considered by them. Note that this is a good ordering for matrices of banded form. More research is needed on optimal row-orderings, however. We also stress, that for stability reasons, rows with large norms must be processed first.

George, Heath and Flemmons [1980] have given an implementation of the method described above using auxiliary storage. Their algorithm uses a method of partitioning and an algorithm for computing a suitable ordering.
(ii) Solve $x_{m+1}$ from the problem

$$
\min_{x_{m+1}} \left| \left| d - T x_{m+1} \right| \right|^2_2, \quad T = \begin{pmatrix}
T_1 \\
\vdots \\
T_m
\end{pmatrix}, \quad d = \begin{pmatrix}
d_1 \\
\vdots \\
d_m
\end{pmatrix}.
$$

(iii) Backsubstitution in the triangular systems

$$
B_i x_i = c - S_i x_{m+1}, \quad i = 1, 2, \ldots, m,
$$

now gives $x_1, \ldots, x_m$.

A more generally useful way to obtain a favourable blocking of variables in sparse problems is by substructuring or dissection. This process has been used by geodesists for almost a decade, and is in that context called Helmert blocking.

A nested dissection ordering of the columns of $A$ can be obtained in the following way; see also George and Heath [1980]. We introduce the graph $G = (X, E)$ associated with $B = A^T A$, where

$X$ : set of nodes $x_i$, $i = 1, 2, \ldots, n$,

$E$ : set of edges $(x_i, x_j) \in E$ implies $B_{ij} = B_{ji} \neq 0$.

Ordering of the columns of $A$ is now equivalent to a labelling of the nodes of $G$. We illustrate the process of nested dissection on a simple problem below, taken from George and Heath [1980].
We make repeated cuts $S_1, S_2, \ldots$ in the graph $G$, each cut separating one group of variables (nodes) from another. When the remaining isolated groups of nodes contain less than $m$ (a preassigned constant) nodes, we stop and label these groups (in the example $S_1, S_2, \ldots, S_k$). The nodes are then ordered in reverse order, first those in $S_k$, then those in $S_{k-1}, S_{k-2}, \ldots, S_1$. Note that the dissection does not provide an ordering for the variables within each group $S_i$. For this we may use, for example, the minimum degree algorithm described above.

Note that with only one cut, we get a block angular form with $m = 2$ blocks. In geodetic problems the cuts - or separators - correspond to geographic boundaries. A similar interpretation can often be made in other network problems, and then the cuts are easily found from the problem formulation. A general algorithm for selecting the cuts can be found in George and Liu [1981].

In the algorithm by George, Heath and Flemmons [1980] a partitioning of the rows corresponding to the column partitioning is made so that the resulting permuted matrix $A$ is upper block triangular. The transformation of $A$ to triangular form then proceeds in $p$ steps ($p = \text{the number of partitions}$) - in the $i$:th step variables $x_i$ corresponding to $S_{p+1-i}$ are eliminated, $i = 1, 2, \ldots, p$. This algorithm has been successfully used to solve the adjustment problem for large geodetic nets with up to 60,000 unknowns.

2.6 ITERATIVE METHODS

For large sparse linear least squares problems iterative methods are sometimes an alternative. We can apply any iterative method for symmetric positive definite (or semidefinite) systems to the normal equations

\[ A^T A \mathbf{x} = A^T \mathbf{b}. \]

It is important to note that these iterative methods may be formulated in such a way that the explicit formation of \( A^T A \) and \( A^T \mathbf{b} \) is avoided.

**Example:** Richardson's first-order stationary method can be written

\[ x_{k+1} = x_k + \alpha (b - A x_k), \]

where \( x_k \) denotes the \( k \):th approximation to the solution vector.

There are two advantages with this formulation.

(i) **Accuracy.** Perturbations in \( A^T A \) may change the solution much more than perturbations of similar size in \( A \) itself.

(ii) **Operation count.** For problems which are only moderately sparse \( A^T A \) may be much less sparse than \( A \). (For example if \( A \) has about \( n^{1/2} \) nonzeros in each row, randomly distributed, then \( A^T A \) will be almost full!)

The basic drawback with iterative methods is that often the rate of convergence will be unacceptably slow. The rate of convergence will depend on \( \kappa^2(A) \), which means that the number of iterations needed is squared compared to a symmetric, positive definite system with the same condition.

Convergence can sometimes be accelerated by a transformation of variables

\[ \bar{\mathbf{x}} = C \mathbf{x}, \quad b - A \mathbf{x} = b - (AC^{-1}) \bar{\mathbf{x}}, \]

where \( C \) is an \( n \times n \) non-singular matrix chosen so that

\[ \kappa(AC^{-1}) \ll \kappa(A). \]

Such a transformation of the problem is also called a preconditioning.

We remark that \( AC^{-1} \) is not computed but treated as the product of two operators.
An optimal precondition matrix would obviously be $C = R$, the Cholesky factor of $A$. Then $\kappa(AC^{-1}) = 1$, since $AR^{-1}$ is orthogonal, and iterative methods converge in one step. Thus, taking $C$ to be an approximation to $R$ is often a good choice.

(i) Partial factorization: Take $C = \tilde{R}$, where
\[ A^T \tilde{A} = \tilde{R}^T \tilde{R} , \] but $\tilde{R}$ sparse.

The simplest choice, which corresponds to a diagonal scaling of the columns of $A$, is:

(ii) Diagonal scaling: Take
\[ C = \text{diag}(||a_1||_2, \ldots, ||a_n||_2) , \]
so that $AC^{-1}$ has columns of unit length. This can be shown to be close to the optimal diagonal scaling, with respect to minimizing $\kappa(AC^{-1})$.

Another preconditioning, which is cheap to use, is the SSOR preconditioning described in Björck and Elfving [1979].

(iii) SSOR preconditioning: $C = C(\omega), \ 0 < \omega < 2$. The choice $\omega = 0$, gives diagonal preconditioning. Often $\omega = 1$ is almost optimal.

We have seen above that in many problems there is a natural partitioning of $A$ into blocks of columns
\[ A = \begin{pmatrix} A_1 & A_2 & \cdots & A_s \end{pmatrix}, \quad A_i^T A_i = R_i^T R_i . \]
If the Cholesky factors $R_i$ can be computed, then there are generalizations of (ii) and (iii), e.g. block-diagonal scaling
\[ C = \text{diag}(R_1, R_2, \ldots, R_s) . \]

Among iterative methods the conjugate gradient method offers many advantages. The following special version due to Hestenes and Stiefel [1952] should be used:

Take $x_0 = 0$, $r_0 = b$, $p_0 = s_0 = A^T r_0$,

and for $k = 0, 1, 2, \ldots$ compute
\[ q_k = A p_k, \quad a_k = \frac{\| s_k \|_2}{\| q_k \|_2}, \]
\[ x_{k+1} = x_k + a_k p_k, \quad r_{k+1} = r_k - a_k q_k, \]
\[ s_{k+1} = A^T r_{k+1}, \quad b_k = \frac{\| s_k + 1 \|_2}{\| s_k \|_2}, \]
\[ p_{k+1} = s_{k+1} + b_k r_k. \]

The main work in each iteration is generally in the matrix-vector products \( A p_k \) and \( A^T r_{k+1} \) [or with preconditioning \( A(C^{-1} p_k), C^{-1}(A^T r_{k+1}) \)]. Note again that \( A^T A \) is never explicitly formed.

In the absence of rounding error the conjugate-gradient method produces the exact solution in \( t \leq n \) steps, where \( t \) is the number of distinct non-zero singular values of \( A \). However, the method is best regarded as an iterative method, and it can be shown that \( (x_k - x) \|_2 \) and \( r_k \|_2 \) decrease monotonically. For a well-conditioned problem the method may converge to a sufficiently accurate solution in far less than \( n \) iterations.

Paige and Saunders [1978] have given a new method of conjugate gradient type, which although mathematically equivalent to the method above, has proved to have better numerical properties for less well-conditioned least squares problems.

Another iterative method, which on account of its simplicity has been used for the solution of sparse least squares problems, is the SOR method (successive overrelaxation method), see, for example, Ashkenazi [1971]. An implementation of this method, which avoids forming \( A^T A \) is given in Björck [1979]. Again, if \( A \) is partitioned into blocks of columns, a block-SOR method may be used. The SOR method depends for its rapid convergence on \( A^T A \) having the so called "property A". This is usually not the case in least squares applications. An interesting special case is when \( A \) is partitioned into two blocks \( (s = 2) \). Then the SOR theory applies to the block-SOR method, and rapid convergence can be achieved, see Elfving [1980].
2.7 SOFTWARE

Few general purpose programs are available for sparse linear least squares problems. Several subroutines are however available for solving sparse, symmetric positive definite systems, and these can be used to implement the method of normal equations. We mention here three such subroutines or collections of subroutines:

SPARSEPAK, developed by George and Liu [1978], at the University of Waterloo, Canada, embodies a large variety of subroutines. YSMF - Yale Sparse Matrix Package, developed by Eisenstat et al. [1977], is a neat and efficient code. The Harwell subroutine MA31, by Munksgaard [1979], is comparable in efficiency with YSMF, and has an option for using a partial factorization as a preconditioning matrix together with an iterative method.

A program implementing a sparse variant of the Peters-Wilkinson method, see Björck and Duff [1980], is being prepared for the Harwell library. This subroutine is developed especially for weighted, stiff problems with linear equality constraints and includes an updating step for adding non-sparse equations.

Perhaps the most interesting program for very large and sparse problems is the program developed by George, Heath and Flemmons [1980]. This is being developed at Oak Ridge National Laboratory, and will probably become available in the near future.

Among iterative methods we mention here two published codes. The conjugate gradient type method LSQR by Paige and Saunders will appear in ACM Transactions on Mathematical Software in 1981/82. The SSOR-preconditioned version of the conjugate gradient method has appeared in Björck and Elfving [1978].
2.8 REFERENCES


III. NONLINEAR LEAST SQUARES PROBLEMS

3.1 INTRODUCTION

Parameter estimation and curve fitting often give rise to nonlinear least squares problems (NNLS). In the simplest unconstrained case such a problem can be formulated as finding the global minimizer of

\[ \phi(x) = \frac{1}{2} \sum_{i=1}^{m} r_i^2(x) = \frac{1}{2} \|r(x)\|_2^2, \]

\[ x \in \mathbb{R}^n, \quad f \in \mathbb{R}^m, \quad m \geq n. \]

In, for example, curve fitting we want to fit a model of the form \( \eta(x, \theta) \) to given data \((y_i, t_i), i = 1, 2, \ldots, m\). We then have a problem of the form (3.1) with

\[ f_i(x) = \eta(x, t_i) - y_i. \]

The NNLS problem is a special case of minimization of a function of \( n \) variables. We assume for the analysis below that the functions \( f_i \) are twice continuously differentiable and define the Jacobian of \( f(x) \)

\[ J(x) = \left\{ \frac{\partial f_i(x)}{\partial x_j} \right\}, \quad (m \times n \text{ matrix}) \]

and the matrices of second derivatives

\[ G_i(x) = \left\{ \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k} \right\}, \quad (n \times n \text{ matrices}) \]

Then the gradient and Hessian of \( \phi(x) \) have the special form

\[ \phi'(x) = J^T(x) f(x) \]

and

\[ \phi''(x) = J^T(x) J(x) + B(x), \quad B(x) = \sum_{i=1}^{m} f_i(x) G_i(x). \]

This special form of \( \phi'(x) \) and \( \phi''(x) \) is often exploited by algorithms for NNLS problems.
When the second derivatives \( G_i(x) \) are available, Newton's method for minimizing \( f(x) \) can be used. This method constructs a sequence of vectors \( (x_k) \) such that

\[
x_{k+1} = x_k + a_k d_k,
\]

where \( a_k \) is a scalar step length and the search direction \( d_k \) is determined from

\[
[J^T(x_k)J(x_k) + B(x_k)]d_k = -J^T(x_k)f(x_k).
\]

The linear system (3.6) for \( d_k \) should be solved by a method which is stable also when \( J(x_k) \) is ill-conditioned or singular; see Gill and Murray [1978].

The cost of providing the \( mn^2 \) second derivatives needed for the computation of \( B(x_k) \) in (3.6) is often exorbitant, and therefore methods which rely on only using first derivative information are most often used. However, for the curve-fitting problem we note that the values of \( f_i \) and their derivatives can be obtained from a single subroutine for \( n(x,t) \). If \( n(x,t) \) is composed of, for example, simple exponential and trigonometric functions, then second derivatives can sometimes be computed cheaply.

From the discussion above we see that it is convenient to divide methods for the NWLS problem into two categories:

(i) Methods which ignore second derivative terms.

(ii) Methods which explicitly or implicitly take second derivatives into account.

Ignoring the term \( B(x_k) \) in (3.6) leads to the Gauss–Newton method, which will be discussed in section 3.2. In category (ii) fall various types of quasi-Newton methods for general minimization. These are sometimes recommended for large residual problems as are discussed here in section 3.5.
3.2 THE GAUSS-NEWTON METHOD

The simplest way of solving the non-linear problem (3.1) is through approximating by a sequence of linear least squares problems. Let \( x_k \) be the current approximation. Then we compute \( d_k \) as the solution to the problem

\[
\min_{d_k} \| f(x_k) + J(x_k) d_k \|_2, \quad d_k \in \mathbb{R}^n, \tag{3.7}
\]

and take

\[
x_{k+1} = x_k + a_k d_k. \tag{3.8}
\]

The normal equations for the problem (3.7) are

\[
J^T(x_k) J(x_k) d_k = -J^T(x_k) f(x_k),
\]

which equal (3.6) with the matrix \( B(x_k) \) ignored. We remark that (3.7) should preferably be solved by an orthogonalization method and not by forming the normal equations. To emphasize this we write the solution as

\[
d_k = -J^+(x_k) f(x_k),
\]

where \( J^+(x_k) \) is the pseudoinverse of \( J(x_k) \).

The Gauss-Newton method has the following two important properties:

(i) The search direction \( d_k \) in the Gauss-Newton method is invariant under linear transformations of the independent variable \( x \).

(ii) Provided that \( J^T(x_k) f(x_k) \neq 0 \), \( d_k \) is a descent direction i.e.

\[
\| f(x_k + a d_k) \|_2 < \| f(x_k) \|_2,
\]

for \( a > 0 \) sufficiently small.

The second property follows from the following relation

\[
\| f(x_k + a d_k) \|^2 = \| f(x_k) \|^2 - 2a \| J_k^* f(x_k) \|^2 + 0(|a|^2),
\]

where we have introduced the orthogonal projection onto the range of \( J(x_k) \)

\[
P_{J_k} = J(x_k) J^+(x_k).
\]
One can show that $J^T(x_k)r(x_k) \neq 0$ implies $p_k f(x_k) \neq 0$.

To make the Gauss-Newton method into a viable algorithm the steplength $\alpha_k$ must be chosen carefully. Two common ways of choosing $\alpha_k$ are:

(i) Take $\alpha_k$ as the largest of the numbers $1, \frac{1}{2}, \frac{1}{4}, \ldots$, for which

$$\|r(x_k)\|^2 - \|r(x_k + a_k d_k)\|^2 \geq 0.5 \alpha_k \|p_k f(x_k)\|^2.$$ 

This is essentially the Armijo-Goldstein steplength principle, and is used in the Gauss-Newton routine by Wedin [1972]. Note that

$$p_k f(x_k) = -J(x_k) d_k,$$

so this quantity is cheaply available.

(ii) Take $\alpha_k$ as the solution to the one-dimensional minimisation problem

$$\min_{\alpha} \|r(x_k + \alpha d_k)\|^2,$$

i.e., do a line search.

This is computationally more expensive than (i).

One can show that if Gauss-Newton's method converges, then close to the solution we will take undamped steps, $\alpha_k = 1$. A theoretical analysis of these two different steplength principles has been given by Rabe [1979].

We say that $x^*$ is a critical point if

$$\phi'(x^*) = J^T(x^*) f(x^*) = 0 \iff J(x^*) = 0.$$

This is a necessary condition for $x$ to be a local minimum of $\phi(x)$.

We now want to derive a sufficient condition for a critical point to be a local minimum, and follow the analysis by Wedin [1974b]. We assume that $J(x^*)$ has rank $n$ and rewrite (3.4) as
\[ \phi^{(k)}(x^*) = J^T J - G_w = J^T (I - \gamma (J^T G_w J)^+) J, \]

where
\[ G_w = \sum_{j=1}^{m} \omega_j G_j, \quad w = -f/\|f\|_2, \quad \gamma = \|f\|_2, \]

and all quantities are evaluated at the point \( x^* \). The matrix
\[ K = (J^T G_w J)^+ \quad (3.10) \]
is called the normal curvature matrix of the surface \( y = f(x) \). Let the nonzero eigenvalues of \( K \) be
\[ \kappa_1 \geq \kappa_2 \geq \ldots \geq \kappa_n. \quad (3.11) \]
Then we have the result that \( \phi^{(k)}(x^*) \) is positive definite if and only if
\[ 1 - \gamma \kappa_1 > 0. \]
The quantities \( 1/\kappa_i \) are the principal radius of curvature of the \( n \)-dimensional surface \( y = f(x) \) in \( \mathbb{R}^n \).

**Example:** \( n = 1, \quad m = 2. \)

We note that the geometrical interpretation of the problem \( \min_{x} \|f(x)\|_2 \)
is to find a point on the surface \( y = f(x) \) closest to the origin.
3.3 LOCAL CONVERGENCE OF THE GAUSS-NEWTON METHOD

The local convergence of the Gauss-Newton method has been analyzed by Wedin [1974b] and Ransin and Wedin [1977]. One step of the undamped method (i.e. taking \( \alpha = 1 \)) can be written

\[ x_{k+1} = g(x_k), \quad g(x) = x - J^+(x)f(x). \]

Therefore the asymptotic rate of convergence is bounded by the spectral radius \( \rho \) of the matrix \( g^*(x) \) at the solution \( x^* \),

\[ g^*(x) = -J^+(x)[J^+(x)T \sum_{i=1}^n \frac{\partial f_i(x)G_i(x)}{\partial x_i}] = \gamma J^+(J^+)T G_W, \]

see Ortega and Rheinboldt [1970]. But \( g^*(x) \) has the same nonzero eigenvalues as the matrix

\[ K = \gamma (J^+)T G_W J^+, \]

and thus

\[ \rho = \gamma \max(\kappa_1, -\kappa_n). \quad (3.12) \]

In general convergence is linear, but if \( \gamma = ||f(x^*)||^2_2 = 0 \), we have superlinear convergence. From (3.12) it is evident that we can expect the Gauss-Newton method to converge fast when either

(i) residuals \( ||f(x^*)||^2_2 \) are small, or

(ii) \( f(x) \) is close to linear, i.e. \( |\kappa_i|, i = 1, 2, \ldots, n \) are small.

The rate of convergence for the Gauss-Newton method with the steplength chosen by line search has been shown by Ruhe [1979] to be

\[ \rho = \gamma (\kappa_1 - \kappa_n) / \left[ 2 - \gamma (\kappa_1 + \kappa_n) \right]. \quad (3.13) \]

We have \( \rho = \bar{\rho} \) if \( \kappa_n = -\kappa_1 \), and \( \rho < \bar{\rho} \) otherwise. We also have

\[ \bar{\rho} < 1 \iff \gamma \kappa_1 < 1, \]

i.e. with line search we always get convergence close to a local minimum.
Without line search the Gauss-Newton method may fail to converge to a local minimum.

The rate of convergence for the Gauss-Newton method can be estimated during the iterations from

$$
||J_{k+1}^{-1}d_{k+1}||_2 / ||J_k d_k||_2 \leq \rho + O(||x_k - x^*||_2^2).
$$  \hspace{1cm} (3.14)

When the estimated $\rho$ is greater than 0.5 (say) then one should switch to another method - or perhaps evaluate the quality of the underlying model for the problem.

In section 3.2 we interpreted the nonlinear least squares problem in geometrical terms as seeking the point on the n-dimensional surface $y = f(x)$ closest to the origin. An example of an ill-conditioned least squares problem is shown in the figure below.

In this problem the radius of curvature $1/\kappa \ll ||f(x)||_2$. For problems like this it seems reasonable to demand that $f(x)$ is smoothed before one attempts to solve the problem. Wedin [1974b] has shown that the rate of convergence of the Gauss-Newton method is a good confirmation of the quality of the underlying model. Deuflhard and Apostolescu [1980] call problems for which divergence occurs inadequate problems.
3.4 THE LEVENBERG-MARQUARDT METHOD

If there are points with \( \|f(x)\|_2 < \|f(x_0)\|_2 \) for which \( \text{rank}[J(x)] < n \), then the Gauss-Newton method can jam up. This can be avoided either by taking the second derivative term into account (see section 3.6) or by stabilizing the method by the Levenberg-Marquardt technique. In the latter approach we modify the Gauss-Newton method by computing the search direction \( d_k \) as the solution to the problem

\[
\min_d \| f(x_k) + J(x_k)d \|_2^2 + \mu_k \| D_k d \|_2^2, \quad \mu_k > 0,
\]

where \( D_k \) is a diagonal, nonsingular matrix. It can be shown that for some \( \delta_k = \delta(\mu_k) > 0 \) this is equivalent to the problem

\[
\min_d \| f(x_k) + J(x_k)d \|_2, \quad \text{subject to } \| D_k d \|_2 \leq \delta_k. \quad (3.15)
\]

The set \( \{ d : \| D_k d \|_2 \leq \delta_k \} \) can be thought of as a region of trust for the linear model.

A careful implementation and rigorous analysis of the Levenberg-Marquardt method has been given by Moré [1978], who considers an iteration of the following form:

(i) Given \( \delta_k \) find \( d_k \) as the solution of (3.15).

(ii) If \( \| f(x_k + d_k) \|_2 \leq \| f(x_k) \|_2 \), then set \( x_{k+1} = x_k + d_k \), else put \( x_{k+1} = x_k \).

(iii) Determine \( \delta_{k+1} \) and \( D_{k+1} \).

The updating of \( \delta_k \) in step (iii) is based on the objective of keeping the ratio

\[
\frac{\| f(x_k) \|_2^2 - \| f(x_k + d_k) \|_2^2}{\| f(x_k) \|_2^2} \sqrt{\| f(x_k) \|_2^2 - \| f(x_k + d_k) \|_2^2}^2}
\]

close to one. This ratio measures the agreement between the linear model and the nonlinear function.
The purpose of the diagonal matrix $D_k$ is to take into account the scaling of the problem, and one takes

$$D_k = \text{diag}[d_1^{(k)}],$$

where

$$d_1^{(0)} = \left\| \frac{\partial f}{\partial x_1}(x_0) \right\|_2, \quad d_1^{(k)} = \max\{d_1^{(k-1)}, \left\| \frac{\partial f}{\partial x_1}(x_k) \right\|_2\}, \quad k > 1.$$  

An important observation is that this choice of scaling makes the algorithm scale invariant, i.e. the algorithm generates the same iterations if applied to $f(Dx)$ for any nonsingular diagonal matrix $D$.

Moré [1978] proves that if $f(x)$ is continuously differentiable, $f''(x)$ uniformly continuous, and $J(x_k)$ bounded, then the algorithm will converge to a critical point. The main drawback with the Levenberg-Marquardt method is that it may unnecessarily slow down the convergence for problems where the Gauss-Newton method works well.

3.5 METHODS USING SECOND DERIVATIVE INFORMATION

The previous analysis has shown that when $\left\| f'(x^*) \right\|_2^2 \neq 0$ the Gauss-Newton method will only converge linearly, and further in the large residual case convergence will be slow. To get a quadratically convergent method the second term in the Hessian (3.4) has to be taken into account, either explicitly or implicitly. A great number of such algorithms have been suggested and below we briefly review some of the more promising.

The simplest alternative is to use a general quasi-Newton optimization routine (e.g. the well known VA09A in the Harwell library). Many of these are known to possess superlinear convergence. These compute search directions from

$$S_k d_k = -\phi(x_k) = -J^T(x_k)f(x_k),$$

where $S_k$ is an approximation to the Hessian chosen to satisfy the so-called quasi-Newton relations.
\[ S_k(x_k - x_{k-1}) = J^T_k(x_k) f(x_k) - J^T_{k-1}(x_{k-1}) f(x_{k-1}) , \]  
(3.16)

and developed from \( S_{k-1} \) by a change of small rank. The user needs to assign a starting value to \( S_0 \) and the choice

\[ S_0 = J^T(x_0) f(x_0) \]

is usually recommended.

The general quasi-Newton routines do not take the special structure of the nonlinear least squares problem into account, and have the unsatisfactory feature that they form the search direction essentially by solving the normal equations. However, several special algorithms using a quasi-Newton approach have been developed; for a survey see Dennis [1977] and Nazareth [1980]. In particular we mention here the algorithm NL280L by Dennis, Gay and Welsch [1980]. This algorithm uses an approximation to the Hessian of the form

\[ J^T_k J_k + S_k , \]

where \( S_0 = 0 \) and \( S_k \) satisfies the quasi-Newton relations (3.16). The algorithm is very sophisticated and includes an automatic scaling of \( S_k \).

Ramsin and Wedin [1977] gave the following recommendations on the choice between a Gauss-Newton (GN) and a quasi-Newton (QN) method.

(i) For \( \rho \leq 0.5 \) GN is better.

(ii) For globally simple problems QN is better for \( \rho > 0.5 \).

(iii) For globally difficult problems GN is much faster for \( \rho \leq 0.7 \).

For larger values, \( \rho > 0.7 \), QN is safer.

Nazareth [1980] has suggested a hybrid between the Gauss-Newton and quasi-Newton methods, where the search directions are computed from

\[ \beta_k J^T_k J_k + (1-\beta_k) S_k \]  

and \( g_k \) is an adaptively chosen parameter.
A different approach is taken by Gill and Murray [1976]. They suggest regarding \( J^T_k J_k \) as a good estimate of the Hessian in the subspace corresponding to the large eigenvalues of \( J^T_k J_k \). In the complementary subspace they use a finite difference approximation to the second derivatives. Gill and Murray [1978] develop a stable algorithm for solving equations with coefficient matrix \( J^T_k J_k + S_k \), where \( J_k \) may be singular and \( S_k \) indefinite, which is of interest also for quasi-Newton methods.

Ruhe [1979] has developed a nonlinear conjugate gradient acceleration of the Gauss-Newton method, which implicitly obtains information about the second-order derivatives. This method achieves quadratic convergence. Ruhe [1979] compares three algorithms: Gauss-Newton without line search, Gauss-Newton with line search and Gauss-Newton with line search and conjugate gradient acceleration. He concludes that for large residual problems the conjugate gradient acceleration gives a much better rate of convergence and in general it never increases the number of iterations required. When line search is used he therefore recommends that conjugate gradient acceleration is used, since it amounts to a negligible amount of extra work. However, in well-formulated problems, where the residual is small, line search is a waste of time and then the simplest Gauss-Newton method is superior.

3.6 CONSTRAINED PROBLEMS

In a more general formulation of the nonlinear least squares problem, the solution \( x \) may be subject to nonlinear equality constraints. Such a problem can be stated as

\[
\min \frac{1}{2} \| r(x) \|_2^2, \quad x \in \mathbb{R}^n, \quad r \in \mathbb{R}^m,
\]

subject to

\[
h_k(x) = 0, \quad k = 1, 2, \ldots, p, \quad p \leq n, \quad m + p \geq n.
\]

As in the Gauss-Newton method we can obtain a search direction by
linearizing at the current approximation \( x_k \), giving \( d_k \) as the solution to the problem

\[
\min_d |f(x_k) + J(x_k)d|^2, \quad \text{subject to} \quad h(x_k)^* C(x_k)d = 0.
\]

This is a linear least squares problem with linear equality constraints and can be solved by the types of method described in section 1.7. The search direction \( d_k \) computed like this can be shown to be a descent direction to the penalty function

\[
\phi(x,\nu) = ||f(x)||^2_2 + \nu ||h(x)||^2_2
\]

at the point \( x_k \) provided \( \nu \) is large enough.

With a suitable active set strategy such an algorithm can be extended to handle problems which also have nonlinear inequality constraints. An algorithm based on this approach has been developed by Lindström [1980].

Implicit curve fitting problems, where a model with parameters \( x, h(t,y,x) = 0 \),

is to be fitted to a set of points \((t_i,x_i^1, y_i^1), i = 1,2, \ldots, m\), can be formulated as a constrained least squares problem:

\[
\min_{x,z} ||z - y||^2_2, \quad \text{subject to} \quad h(t_i, z_i, x) = 0, \quad 1 \leq i \leq m.
\]

This is a special case of (3.17). It has many unknowns, \( x \in \mathbb{R}^n \) and \( z \in \mathbb{R}^m \), but the Jacobian matrices are sparse, which may be utilized in the implementation (see Lindström [1980]).

Another particular case, which may be formulated as a constrained nonlinear least squares problem, is the problem of orthogonal regression with an implicit model. Lindström [1980] considers the ellipse problem to determine the two-dimensional curve

\[
x^T U x + b^T x + c = 0,
\]

which fits a measured points in the orthogonal regression sense and gives numerical results for this problem.
3.7 SEPARABLE NONLINEAR PROBLEMS

Assume that in the nonlinear least squares problem
\[ \min_{x} \| f(x) \|_2^2, \quad x \in \mathbb{R}^n, \]  
(3.18)
we can partition the variables \( x \) as
\[ x = (y, z), \quad w \in \mathbb{R}^p, \quad z \in \mathbb{R}^q, \quad p+q = n, \]
so that \( f \) is linear in the variables \( w \) and
\[ f(w, z) = g - F(z)w. \]  
(3.19)
Then for fixed \( z \) the subproblem
\[ \min_{w} \| g - F(z)w \|_2^2 \]  
(3.20)
is a linear least squares problem. A solution to this problem is given by
\[ w(z) = P^*(z)g, \]
where \( P^*(z) \) is the pseudoinverse of \( F(z) \). We can now replace the \( n \)-dimensional original problem (3.18) by the \( q \)-dimensional one
\[ \min_{z} \| g - F(z)w(z) \|_2^2, \quad w(z) = P^*(z)g. \]  
(3.21)
Separable problems often occur in curve fitting. Suppose we want to fit the model
\[ y(t) = \sum_{j=1}^{p} w_j \eta_j(z, t), \]
to given data \( (y_i, t_i), \quad i = 1, 2, \ldots, m \). If we put
\[ f_i(w, z) = y_i - \sum_{j=1}^{p} w_j \eta_j(z, t_i), \]
then \( f(w, z) \) is of the form (3.19) with
\[ \left[ F(z) \right]_{i,j} = \eta_j(z, t_i). \]
A special instance of this occurs in exponential fitting where
\[ \eta_j(z,t) = e^{Z_j^t}. \]

An algorithm for separable problems of the form (3.19) was first developed by Golub and Pereyra [1973]. Kaufman [1975] proposed a simplification of this algorithm, which computationally proved to be more efficient. Ruhe and Wedin [1980] give a more general analysis of different algorithms for separable problems. They show that the Gauss-Newton algorithm applied to (3.18) and (3.21) gives comparable rates of convergence; both converge almost quadratically for almost compatible problems. This is in contrast to some simpler algorithms, which always converge linearly. They also prove that the simplified algorithm of Kaufman has roughly the same asymptotic convergence rate as the one proposed by Golub and Pereyra.

We note that we can write (3.21) as

\[ \min_{z} \left\| (I - F(z))z \right\|^2, \quad F(z) = F(z)^*F(z), \]

where \( F(z) \) is the orthogonal projector on the linear space spanned by the columns of the matrix \( F(z) \). This explains why algorithms based on (3.21) are often called variable projection algorithms.

The basic idea is now to apply some method for nonlinear least squares to the problem (3.21), using the coupling between the \( w \) and \( z \) variables. Here we just give a heuristic derivation of the Kaufman algorithm. Put \( r(z) = g - F(z)w(z) \). Then

\[ r(z + \delta z) = g - F(z)w(z + \delta z) = B(z)\delta z + O(\left| \delta z \right|^2), \]

where

\[ B = F^*(z)w(z) = \left( \frac{\partial F}{\partial z_1} w, \ldots, \frac{\partial F}{\partial z_q} w \right) \quad (3.22) \]

is an \( m \times q \) matrix. We now split \( r(z + \delta z) \) into two components, one in the range of \( F(z) \) and one orthogonal to that. Then, neglecting quantities of order \( |\delta z|^2 \), we have \( \left\| r(z + \delta z) \right\|^2 \approx I + \Pi \), where
\[ I = \left\| F(z)(g - B\delta z) - F(z)w(z + \delta z) \right\|_2^2, \]
\[ II = \left\| (I - P_F(z))(g - B\delta z) \right\|_2^2. \]

Now, for any \( \delta z \) we can make the term I zero by taking \( w(z + \delta z) \) to be the solution to the least squares problem
\[
\min_{\delta z} \left\| (g - B\delta z) - F(z)w \right\|_2^2.
\]
But up to quantities of second order in \( \delta z \), this is equivalent to
\[
\min_{\delta z} \left\| g - F(z + \delta z)w \right\|_2^2,
\]
which is just the functional relationship \( (3.21) \) between \( w \) and \( z \).

Therefore, to minimize \( \left\| r(z + \delta z) \right\|_2^2 \), we want to determine \( \delta z \) as the solution to
\[
\min_{\delta z} \left\| (I - P_{F(z)})g - (I - P_{F(z)})B \delta z \right\|_2^2, \tag{3.23}
\]
which is a linear least squares problem.

Briefly, an outline of the computation is as follows. (For more details see Hoss and Wedin [1980], pp. 329-331.)

(i) Compute the QR decomposition of \( P(z_k) \), where \( z_k \) is the current approximation and for simplicity we assume that the rank of \( P(z_k) \) is \( p \):
\[
Q_k^T F(z_k) = \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \quad Q_k^T g = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix}
\]
where \( R_k = p \times p \) matrix, \( g_1 = p \times 1 \) matrix. Then, \( w_k = R_k^{-1} g_1 \).

(ii) Compute the \( m \times q \) matrix \( B_k = F'(z_k)w_k \) and perform the transformations
\[
Q_k^T B_k = \begin{pmatrix} B_{1k} \\ B_{2k} \end{pmatrix}, \quad B_{1k} = p \times q \text{ matrix}.
\]

(iii) Find \( \delta z_k \) by solving the linear least squares problem
\[
\min_{\delta z} \left\| g_2 - B_{2k} \delta z \right\|_2^2, \tag{3.24}
\]
by a QR decomposition of \( B_{2k} \). Take \( z_{k+1} = z_k + \delta z_k \).
Note that in substituting (3.24) for (3.23) we have used the invariance of the norm under orthogonal transformations.

One advantage with variable projection algorithms is that no starting values for the linear parameters have to be provided. Krogh [1974] reports that the variable projection algorithm has solved several problems which other nonlinear least squares methods could not solve. Kaufman [1975] mentions that common methods for nonlinear least squares problems such as Gauss-Newton, Levenberg-Marquardt and quasi-Newton have required less time to solve (3.21) than (3.18).

Golub and LeVeque[1979] have extended the variable projection method for solving problems in which it is desired to fit more than one data vector with the same nonlinear parameters (though possibly different linear parameters) for each right-hand side. They also give further modifications for dealing with the total least squares problem, cf. section 1.1. Kaufman and Pereyra [1978] have extended the method to nonlinearly constrained problems.

We end with an example from Ruhe and Wedin [1980].

**Example:** Consider the curve-fitting problem with \( p = q = 1 \) and
\[
y(t) = v \exp(-st^2).\]

Data was artificially generated as \( y_i = y(t_i) + \epsilon_i, \ i = 1,2,\ldots,m, \)
\( t_i = i/m, \) where \( \epsilon_i \) are taken from a sequence of pseudo-random numbers having a roof-top distribution in the interval \([-1,1]\).

For \( m = 50, \) and different values of the perturbation \( \epsilon \) the observed rate of convergence for the Kaufman algorithm (K) and a simpler separable algorithm (S), ignoring the functional relationship between \( w \) and \( z \), is given below.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>( \epsilon )</th>
<th>0</th>
<th>0.1</th>
<th>0.4</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>quadr.</td>
<td>0.0005</td>
<td>0.036</td>
<td>0.088</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.41</td>
<td>0.43</td>
<td>0.39</td>
<td>0.45</td>
</tr>
</tbody>
</table>
3.8 SOFTWARE

There exists a large number of programs for solving nonlinear least squares problems; some generally available, some in the experimental stage. They all require the user to provide a subroutine to calculate the function values of \( f(x) \). Some also require a subroutine for first-order derivatives, i.e. the elements of the Jacobian matrix of \( f(x) \). A few also require second-order derivatives.

The NAG library contains two subroutines for nonlinear least squares. E04JAF only requires function values. It generates from an initial estimate a set of at least \((n+1)\) points from which the Jacobian matrix is estimated. Apart from this it essentially is a Gauss-Newton method. The other subroutine E04UBF also requires first derivatives and uses a Marquardt modification of the Gauss-Newton method.

IMSL has the subroutine ZXSSQ, which only requires function values. It uses a Levenberg-Marquardt algorithm with a finite difference approximation of the Jacobian matrix, see IMSL FORTRAN Library 3 Manual, edition 6, Chapter 2.

The NFL Algorithm Library (National Physical Laboratory, England) contains three subroutines based on the work by Gill and Murray. These are:

- **LSQSDN** Uses Newton's method and requires both first- and second-order derivatives.

- **LSQPDF** Uses a finite difference approximation of second-order derivatives in a subspace. Only requires first-order derivatives from the user.

- **LSQPDQ** Uses a quasi-Newton algorithm and requires only first-order derivatives.
MINPACK-1 is a package of FORTRAN subprograms for the numerical solution of nonlinear equations and nonlinear least squares problems documented in Moré et al. [1980]. MINPACK-1 contains three subprograms for nonlinear least squares all based on the Levenberg-Marquardt algorithm as developed by J.J. Moré. It also has easy-to-use driver routines with a simplified calling sequence when full flexibility is not required. The three subprograms are:

**LMSEF**
Requires only function values and uses a forward difference approximation to the Jacobian matrix.

**LMBER**
Requires a subroutine also for the calculation of the Jacobian matrix.

**LMSTR**
Requires a subroutine which computes the row of the Jacobian matrix, one row per call.

The MINPACK-1 package comprises approximately 28,000 card images and is available either from IMSL or from National Energy Software Center, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

NLSSOL is a quasi-Newton subroutine, which implements the algorithm of Dennis, Gay and Welsch [1980]. It requires first derivatives from the user. It maintains a scaled secant approximation for the second-order part of the Hessian and adaptively decides when to use it. It uses a Levenberg-Marquard scheme for the Gauss-Newton model and a similar scheme for the augmented model. The subroutine comprises 2360 FORTRAN statements and is available through IMSL.

LSE is a program developed by Wedin and Lindström and documented in Lindström [1980]. This has the unusual feature that it solves the nonlinear least squares problem where the solution $x$ is subject to the nonlinear constraints

$$h_i(x) = 0, \quad i = 1, 2, \ldots, p, \quad h_i(x) \geq 0, \quad i = p+1, \ldots, q.$$  

The program uses a generalized Gauss-Newton method with step length control. It is planned to include line search and conjugate gradient acceleration at a later stage. The subroutine is informally available.
from its author at Umeå University, Department of Numerical Analysis, S-901 87 Umeå, SWEDEN.

Two other recently developed subroutines are NLSQA (requires derivatives) and NLSQN (only function values required). These are based on an underrelaxed Gauss-Newton method, i.e. they use a special steplength strategy. These subroutines are developed by Deufihard and Apostolovcu and are available from the Institut für Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 294, D-6900 Heidelberg.

For separable linear/nonlinear problems the VARPRO subroutine, based on the research of Golub and Pereyra [1973], [1976] and Kaufman [1975], is available from Stanford University, Computer Science Department, Stanford, CA 94305. VARPRO needs no initial guesses for the linear parameters. It also computes the covariance matrix of the solution.

Note added in proof: In a recent paper by Hiebert [1981] mathematical software that solves nonlinear least squares problems is evaluated. The evaluation is based on a comparison of 12 readily available FORTRAN codes. Theoretical and software aspects of the codes, as well as their performance on a set of test problems, are used in the comparison,
3.9 REFERENCES


P.Å. Wedin (1972, 1974a). The nonlinear least squares problem from a numerical point of view: I and II. Dept. of Comp. Sciences, Lund University, Sweden.
