INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
Kim, Byung Chun

A CONJUGATE GRADIENT ALGORITHM FOR ANALYSIS OF VARIANCE COMPUTATIONS

Iowa State University

University Microfilms International

300 N. Zeeb Road, Ann Arbor, MI 48106
A conjugate gradient algorithm for analysis of variance computations

by

Byung Chun Kim

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Statistics

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1984
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. PRELIMINARIES</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Notations and Preliminaries</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Minimization with the Conjugate Gradient Algorithm</td>
<td>9</td>
</tr>
<tr>
<td>3. ANALYSIS OF VARIANCE COMPUTATIONS ASSOCIATED WITH THE BALANCED COMPLETE STRUCTURE</td>
<td>14</td>
</tr>
<tr>
<td>3.1 The Basic Algorithm</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Properties and Applications of $M^*$</td>
<td>23</td>
</tr>
<tr>
<td>4. THE METHOD OF CONJUGATE DIRECTIONS</td>
<td>34</td>
</tr>
<tr>
<td>4.1 Basic Properties of Quadratic Functions</td>
<td>35</td>
</tr>
<tr>
<td>4.2 Minimization of a Quadratic Function</td>
<td>41</td>
</tr>
<tr>
<td>4.3 Method of Conjugate Directions (CD-Algorithm)</td>
<td>51</td>
</tr>
<tr>
<td>4.4 Method of Conjugate Gradients (CG-Algorithm)</td>
<td>58</td>
</tr>
<tr>
<td>5. MODIFIED CONJUGATE GRADIENT METHOD</td>
<td>71</td>
</tr>
<tr>
<td>5.1 Conjugate Gradient Algorithms</td>
<td>72</td>
</tr>
<tr>
<td>5.2 Nonnegative Hessians</td>
<td>84</td>
</tr>
<tr>
<td>5.3 Generalized Conjugate Gradient Algorithms</td>
<td>94</td>
</tr>
<tr>
<td>6. COMPUTATIONAL ALGORITHMS FOR UNBALANCED STRUCTURE USING THE CONJUGATE GRADIENT METHOD</td>
<td>105</td>
</tr>
<tr>
<td>6.1 Parameter Estimation from the Normal Equations</td>
<td>105</td>
</tr>
<tr>
<td>6.2 Sum of Squares for Regression and Residual</td>
<td>110</td>
</tr>
<tr>
<td>6.3 Rank of the Design Matrix when Missing Cells are Present</td>
<td>115</td>
</tr>
<tr>
<td>6.4 Weighted Least Squares Estimation</td>
<td>125</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>7.</td>
<td>HYPOTHESES TESTS WITH UNBALANCED DATA</td>
</tr>
<tr>
<td>7.1</td>
<td>Iterative Methods</td>
</tr>
<tr>
<td>7.2</td>
<td>Reparameterizing the Nonfull Rank Model Using $M^*$</td>
</tr>
<tr>
<td>7.3</td>
<td>Direct Computation of Hypothesis Sum of Squares</td>
</tr>
<tr>
<td>7.4</td>
<td>Analysis of Covariance</td>
</tr>
<tr>
<td>8.</td>
<td>COMPUTER PROGRAMMING IMPLEMENTATION</td>
</tr>
<tr>
<td>8.1</td>
<td>Features and Objectives</td>
</tr>
<tr>
<td>8.2</td>
<td>Elementary Algorithms</td>
</tr>
<tr>
<td>8.2.1</td>
<td>The conjugate gradient algorithm</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Iterative rank algorithm</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Balanced factorial decomposition algorithm</td>
</tr>
<tr>
<td>8.2.4</td>
<td>E-P list construction algorithm</td>
</tr>
<tr>
<td>8.3</td>
<td>Logical Components</td>
</tr>
<tr>
<td>8.4</td>
<td>Examples</td>
</tr>
<tr>
<td>9.</td>
<td>BIBLIOGRAPHY</td>
</tr>
<tr>
<td>10.</td>
<td>ACKNOWLEDGEMENTS</td>
</tr>
<tr>
<td>11.</td>
<td>APPENDIX</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

In recent years, the increasing power and the decreasing price of computers, both mainframe and microcomputers, have made them increasingly available for statistical analysis. Powerful, compact microcomputers are now easily purchased by individual groups, including statistics departments. The resulting ease with which data may be gathered and manipulated has led to a corresponding increase in the size of an average statistical problem. To analyze data from such problems, the statistician must have access to algorithms that are sufficiently economical of space requirements that will enable them to run on the available equipment. One area of analysis where there is a need for algorithms that require economical storage is in the fitting of linear models. A nonorthogonal analysis of variance for the linear model is usually performed using regression techniques. Solving the regression problem directly could involve forming and storing a large design matrix. For example, the QR decomposition method of Lawson and Hanson (1974) or the symmetric sweep method of Goodnight (1979) each requires $O(p^2)$ storage to fit a linear model containing $p$ parameters, since they require storage of a triangular structure of dimension $p$. When $p$ is large, it may not be possible to retain these in the memory in which case both the data and the triangular structure must then be placed on a secondary storage medium such as magnetic disk, and retrieved when necessary. This may, of course, cause a considerable loss in efficiency. An algorithm with only $O(p)$
storage requirements may not need to use secondary storage, and consequently may be more efficient. For this reason, we feel that fitting algorithms with $O(p)$ storage requirements should be competitive with standard algorithms in specific situations.

Several such algorithms appear in the statistical literature. Claringbold (1969a,b) and Gower (1969a,b) produce outlines for analysis of variance programs based on tables of marginal means. The storage requirements of these algorithms are fixed and quite substantial, but not $O(p^2)$. James and Wilkinson (1971) and Payne and Wilkinson (1977) describe an algorithm based on the repeated subtraction of means from residuals. Hemmerle (1974, 1976a,b) takes a similar approach. His algorithm (1982) never requires the storage of the design matrix. Only the computation of a sequence of balanced analysis of variance estimates and fits is required. With such algorithms, a piece of software for solving an analysis of variance problem with an orthogonal design can be used repeatedly to solve a nonorthogonal analysis of variance problem. Another important feature of such methods is that they tend to produce shorter and more easily coded algorithms, an advantage when computer storage is at a premium.

On the other hand, very general minimization methods requiring $O(p)$ storage abound in numerical analysis literature. One class of algorithms requiring $O(p)$ storage is the class of conjugate gradient algorithms. These have been used for some time in the
analysis of large, sparse least square problems (Gentleman (1979)).
Recent works by McIntosh (1980) and Golub and Nash (1982) have shown
that conjugate gradient algorithms may also be applied to small, more
standard problems with good results. McIntosh (1982) amplifies and
expands on these investigations. His main focus is on the application
of conjugate gradient algorithms to models associated with common
experimental designs. A major shortcoming of these algorithms is
their inability to carry out computations necessary for testing pre-
specified hypotheses or to produce statistics needed to test standard
hypotheses directly, i.e., without fitting submodels.

In this thesis, we will consider the application of conjugate
gradient algorithms to the linear model. Conjugate gradient algorithms
have the advantage that the storage required to fit a \( p \) parameter
model is of order \( p \). Accordingly, they are well suited to the analysis
of variance problems that are large relative to the amount of computer
memory available. Under appropriate assumptions about balance, we
will show that the number of iterations require to fit a model is
often less than or equal to the number of terms in the model, and
always less than or equal to the rank of the design matrix of the
model.
2. PRELIMINARIES

2.1 Notations and Preliminaries

Throughout this thesis the transpose of a matrix $X$ is denoted by $X'$. A square matrix $X$ is said to be nonsingular if its determinant is not zero. In this case, the inverse of $X$ exists and is denoted by $X^{-1}$. A generalized inverse is denoted by $X^-$, where $X^-$ satisfies the condition $XX^-X = X$. If $X' = X$, then $X$ is said to be symmetric. The Moore-Penrose generalized inverse is denoted by $X^+$ and satisfies the conditions (i) $XX^+X = X$, (ii) $X^+XX^+ = X^+$, (iii) $(X^+X)' = X^+X$, (iv) $(XX^+)' = XX^+$. A matrix $X$ is nonnegative definite if the inequality $\gamma'X\gamma \geq 0$ holds for all vectors $\gamma \in \mathbb{R}^n$ and is positive definite if $\gamma'X\gamma > 0$, whenever $\gamma \neq 0$. A matrix $X$ is nonpositive definite if $-X$ is nonnegative definite and is negative definite if $-X$ is positive definite. In other cases, $X$ is said to be indefinite. $C(X)$ denotes the column space of $X$ and $R(X)$ the row space of $X$.

Consider the fixed effects analysis of variance model

$$y_{ijk} = \mu_{ij} + e_{ijk}, \quad j = 1, 2, \ldots, J,$$

where $E(e_{ijk}) = 0$, $E(e_{ijk}, e_{ikn}) = 0$ for $(i,j,k) \neq (1,m,n)$, $E(e_{ijk}^2) = \sigma^2$, and the $\mu_{ij}$ is some linear combination of crossed terms, nested terms and interaction terms. For example, if
we have the two-way classification with interaction. Letting $X_0$ be the $N \times p$ design matrix for the nonorthogonal model 2.1.1, we write the model 2.1.1 in matrix form as

$$y = X_0 \hat{\beta} + e,$$

(2.1.2)

where $y$ is the $N$-vector of observations, rank $(X_0) = r \leq p$, $\hat{\beta}$ is $p \times 1$ vector of unknown parameters, and $e \sim (0, \sigma^2 I)$. The normal equations for estimating $\hat{\beta}$ are written as

$$X_0'X_0\hat{\beta} = X_0'y.$$

(2.1.3)

Let $X$ be an $n \times p$ design matrix that one would obtain for the model 2.1.1, if there were one and only one observation in each cell. If we compare the unbalanced design matrix $X_0$ to the balanced design matrix $X$, then we can find a matrix $T$ such that $X_0 = TX$. The matrix $T$ has the following form,

$$T_{N \times n} = \begin{pmatrix}
\frac{1}{n_1} & 0 \\
\frac{1}{n_2} & \frac{1}{n_2} \\
\vdots & \vdots \\
0 & \frac{1}{n_n}
\end{pmatrix},$$

where $n_i$ is the number of observations of each cell and $\frac{1}{n_i}$ is a vector with $n_i$ elements 1 for $n_i > 0$ and with elements 0 for $n_i = 0$. We can easily show $TT = D_{n \times n}$, where $D$ is a diagonal
matrix with diagonal elements \( n_i \). Examination of the matrix \( T \) yields some very interesting results.

Let \( D^+ \) be the Moore-Penrose inverse of \( D \). When there are no missing cells, then \( D^+ = D^{-1} \). Define \( Z = T(D^{1/2})^+ \), then we have the following properties,

1. \( ZZ Z = Z \),
2. \( Z Z Z = Z \),
3. \( Z Z \) is symmetric,
4. \( ZZ \) is also symmetric.

Therefore, \( Z \) is the Moore-Penrose inverse of \( Z \).

**Lemma 2.1.1** \((I - ZZ^')X_o = 0\).

**Proof:** Since \( Z \) is the Moore-Penrose inverse of \( Z \), we know that \((I - ZZ^')Z = 0\)

\[
\Rightarrow (I - ZZ^')T(D^{1/2})^+ = 0
\]

\[
\Rightarrow (I - ZZ^')T(D^{1/2})^+D^{1/2}X = 0
\]

\[
\Rightarrow (I - ZZ^')TX = 0
\]

**Lemma 2.1.2** Solving the normal equations \( X'X_o \hat{\beta} = X'o\hat{\gamma} \) is equivalent to solving the equations \( X'DX_o \hat{\beta} = X'D \hat{\gamma} \), where \( \hat{\gamma} \) is the vector of cell means.
Proof: \[ X_o' X_o \hat{\beta} = X_o' y \]
\[ \Rightarrow X T X \hat{\beta} = X T y \]
\[ \Rightarrow X D \hat{\beta} = X \bar{y}_s \text{, where } \bar{y}_s \text{ is the vector of cell sums} \]
\[ \Rightarrow X D \hat{\beta} = X \bar{y} \text{, where } \bar{y} \text{ is the vector of cell means}. \ (2.1.5) \]

It follows that a least squares estimate \( \hat{\beta} \) satisfying the normal equations \( X_o' X_o \hat{\beta} = X_o' y \) has the same form as a solution to the weighted least squares equations given in 2.1.5. Therefore,
\[ \hat{\beta} = (X_o' X_o)^{-1} X_o' y = (X D)^{-1} X D \bar{y}, \quad \text{(2.1.6)} \]
and the regression sum of squares for the model, SSM, is given by
\[ SSM = \hat{\beta}' X D \bar{y} = \hat{\beta}' X \bar{y}_s = \hat{\beta}' X D \bar{y}. \quad \text{(2.1.7)} \]
The residual sum of squares, RSS, is given by
\[ RSS = (y - X \hat{\beta})' (y - X \hat{\beta}) \]
\[ = y'y - \hat{\beta}' X \bar{y} \]
\[ = y'y - \hat{\beta}' X D \bar{y}, \quad \text{(2.1.8)} \]
where \( \hat{\beta} \) is a solution of the normal equations. Golub and Nash (1982) use the weighted least squares method to obtain an expression for the residual sum of squares using the model based on cell means. The cell means model equivalent to 2.1.2 is
\[ \bar{y} = X \bar{\beta} + \delta \quad \text{(2.1.9)} \]
where \( \delta \sim N(0, \sigma^2 D^{-1}) \) and D is the diagonal matrix with elements of
cell frequencies. The normal equations for the above model is

$$(X'DX)\hat{\beta} = X'D\hat{\gamma}$$

and the residual sum of squares is

$$RSS = (\hat{\gamma} - X\hat{\beta})'D(\hat{\gamma} - X\hat{\beta})$$

$$= \hat{\gamma}'D\hat{\gamma} - \hat{\beta}'X'D\hat{\gamma}.$$  \hspace{1cm} (2.1.10)

The residual sum of squares given by 2.1.10 is different from that given by 2.1.8.

If $K$ is a matrix of full row rank $r(\leq \text{rank}(X_o))$ and $K = A X_o$ for some matrix $A$ (i.e., if all elements of $K\hat{\beta}$ are estimable), then the hypothesis

$$H : K\hat{\beta} = m$$  \hspace{1cm} (2.1.11)

can be tested by the statistic

$$F(H) = Q/r\hat{\sigma}^2,$$  \hspace{1cm} (2.1.12)

where

$$Q = (K\hat{\beta} - m)'[K(X_o'X_o)^{-1}K]^{-1}(K\hat{\beta} - m),$$

$$\hat{\sigma}^2 = (\hat{\gamma} - X_o\hat{\beta})'(\hat{\gamma} - X_o\hat{\beta})/(N-r),$$  \hspace{1cm} (2.1.13)

and $\hat{\beta}$ is a solution of the normal equations $X_o'X_o\hat{\beta} = X_o'\hat{\gamma}$. In the special case of $m = 0$,

$$Q = \hat{\beta}'K[K(X_o'X_o)^{-1}K]^{-1}K\hat{\beta}.$$  \hspace{1cm} (2.1.14)
2.2 Minimization with the Conjugate Gradient Algorithm

Let \( \phi(\beta) \) be a function mapping from \( \mathbb{E}^r \) to \( \mathbb{E}^1 \). An alternative to using a direct method for computing \( \hat{\beta} \) is to employ an iterative method when solving the required equations. Starting with an initial approximation \( \hat{\beta}_0 \), an iterative method produces a sequence

\[
\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \ldots
\]

that converges to \( \hat{\beta} \). Given \( \hat{\beta}_k \), a particular next approximation may be computed as

\[
\hat{\beta}_{k+1} = \hat{\beta}_k + \alpha_k \hat{p}_k,
\]

where the search direction \( \hat{p}_k \) is a nonzero vector in \( \mathbb{E}^r \) and the step length \( \alpha_k \) is chosen to produce a reasonable decrease in \( \phi \). The procedure of choosing \( \alpha_k \) is called a linear search; it is said to be exact if \( \alpha_k \) minimizes

\[
\phi^*(\alpha) = \phi(\hat{\beta}_k + \alpha \hat{p}_k).
\]

It will be assumed in all subsequent discussion that \( \phi(\beta) \) has continuous first-order and, as needed, second-order derivatives in the sense that the gradient vector of \( \phi \)

\[
g(\beta) = \begin{bmatrix} \partial \phi / \partial \beta \end{bmatrix} = \begin{bmatrix} \partial \phi / \partial \beta_1, \ldots, \partial \phi / \partial \beta_r \end{bmatrix}^	op
\]

and the Hessian matrix \( G(\beta) = (\gamma_{ij}) \) with

\[
\gamma_{ij} = \frac{\partial^2 \phi(\beta)}{\partial \beta_i \partial \beta_j}
\]

have continuous elements.
Suppose we have a quadratic function,

\[ \phi(\hat{\beta}) = \frac{1}{2} \hat{\beta}' G \hat{\beta} + \hat{b}' \hat{\beta} + c. \]

We assume that \( G \) is positive definite (the nonnegative definite case will be discussed in Chapter 5 and Chapter 6). Now, suppose that we have performed \( k \) steps for choosing a linear independent search direction, resulting in \( \beta_0, \ldots, \beta_{k-1} \). Standard general optimization theory shows that \( \phi \) can be minimized on the manifold

\[ \beta_0 + \text{span}(\beta_0, \ldots, \beta_{k-1}) \]

by taking

\[ \hat{\beta}_k = \beta_0 + P \hat{\alpha}, \]

where

\[ P = (\beta_0 | \beta_1 | \ldots | \beta_{k-1}) \]

and

\[ \hat{\alpha} = -(P' G P)^{-1} P' (G \beta_0 + b). \]

If another search direction is added, and an update of the form 2.2.1 is used, there is no guarantee that \( \hat{\beta}_{k+1} \) will be the minimum of \( \phi \) on the manifold

\[ \beta_0 + \text{span}(\beta_0, \ldots, \beta_k). \]

General optimization theory can be used to show that if an exact line search is used a sufficient condition for this to occur is

\[ P' G \beta_k = 0. \] (2.2.3)

Thus, to use 2.2.1 and ensure at each step that \( \hat{\beta}_k \) is optimal, the search directions should be chosen so that
\[ P_1 G_{i,j} = 0 \text{, for } i \neq j \text{ and } i,j = 0, 1, \ldots, k-1 . \quad (2.2.4) \]

Now, \( G \) defines an inner product, and 2.2.4 represents orthogonality with respect to that product (hence, the name conjugate gradients). A set of \( k \) conjugate search directions could be produced from any set of \( k \) linearly independent vectors via the Gram-Schmidt process. It can be shown, Hestenes and Stiefel (1952), that the vectors \( P_k \) defined by

\[ P_0 = -g_0 \]
\[ P_k = -g_k + a_k P_{k-1} , \quad k = 1, 2, \ldots \]

(2.2.5)

where \( a_k \) is computed according to the one of the formulae presented below, form a set of conjugate directions (more details will be given in Chapter 4 and Chapter 5).

A number of formulae for \( a_k \) appear in the literature. The original algorithm, developed by Hestenes and Stiefel (1952) for quadratic \( \phi \) uses

\[ a_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \]

(2.2.6)

Fletcher and Reeves (1964) proposed the same formula for the general case. Polak and Ribière (1969) suggest

\[ a_k = \frac{g_k (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}} \]

(2.2.7)

In the reference cited above, Hestenes and Stiefel (1952) suggest that the formula
might to be more appropriate in the presence of rounding errors. Beale (1972) proposes the equivalent form

\[ a_k = \frac{\frac{1}{2} g_k^T G_k^{-1} g_k}{g_k^T G_k^{-1} g_k - g_{k-1}^T G_k^{-1} g_{k-1}} \]  

which can be used when \( \phi \) is not quadratic. The formulae 2.2.6, 2.2.7, and 2.2.8 are identical algebraically when \( \phi \) is quadratic and an exact line search is used. In the presence of rounding error, or when \( \phi \) is not quadratic, they may give very different results. Powell (1977) gives reasons why the Polak-Ribière formula is a sounder choice than the formula of Hestenes and Stiefel (1952). Beale (1972) also shows that when \( p_0 \) is chosen arbitrarily, a set of conjugate directions can be obtained as

\[ p_k = -g_k + a_k p_{k-1} + d_k p_0, \quad k = 1, 2, \ldots, \]  

with \( a_k \) is in 2.2.8 and

\[ d_k = \begin{cases} 
0, & \text{when } k = 1, \\
\frac{g_k^T (g_k - g_0)}{g_1^T (g_1 - g_0)}, & \text{when } k = 2, 3, \ldots 
\end{cases} \]  

In this case, formula 2.2.7 is inappropriate. When \( p_0 \) is in fact a multiple of \( g_0 \), it can be shown that \( d_k \) is identically zero, and
2.2.9 reduces to 2.2.5. We will further discuss conjugate gradient algorithms and modified conjugate algorithms in Chapter 4 and Chapter 5.
3. ANALYSIS OF VARIANCE COMPUTATIONS ASSOCIATED WITH
THE BALANCED COMPLETE STRUCTURE

Assume that the form of classificatory model which is to be
utilized reflects completely crossed and/or nested factors and
meaningful interactions. We also assume that the data on hand are
balanced (equal numbers of observations in the subclasses). A general
and efficient method for computing the usual quantities needed in
analysis of variance will be given. This method stems from the work
of many authors, including Yates (1934), Hartley (1956, 1962), Hemmerle
(1964, 1967), Schlater (1965) and Schlater and Hemmerle (1966). It
should be emphasized, again, that the assumption is that the data are
balanced and complete.

It is normally the case for models of this type that the "usual
restrictions" are included with the model and the constraints in the
same form are employed when obtaining desired solutions. If we adopt
this form of constraints, computations are greatly simplified in this
situation. A few simple rules are, as we shall see, all that are
needed to have a sound basis for an efficient algorithm. One reason
for this is the fact that the restricted model is actually of full
rank.

The following section contains a definition of the "usual restric­
tions" and provides the basic rules which are followed in computations
of many quantities needed for the analysis of models of the above type.
3.1 The Basic Algorithm

In the models with which we shall deal, one and only one subscript will be associated with each main effect symbol and the error symbol. In any given subscripted model term (except e) all subscripts which are present in the model term, and are not associated with a main effect symbol in that term, are called floating subscripts. These definitions are made by Hemmerle (1964). To illustrate these definitions, consider the model

\[
y_{ijklm} = \mu + a_i + b_{ij} + c_{ijk} + d_{i1} + a_{i1} + b_{dij} + e_{ijklm},
\]

\[
i = 1, 2, \ldots, I, \quad j = 1, 2, \ldots, J, \quad k = 1, 2, \ldots, K,
\]

\[
l = 1, 2, \ldots, L, \quad m = 1, 2, \ldots, M
\]

(3.1.1)

The associated and floating subscripts in each model term are shown in Table 3.1.1.

Using these definitions, and ignoring the error term for now, the usual restrictions are obtained by summing each model term over its associated subscripts one at a time and equating each sum to zero. In model Equation 3.1.1, the usual restrictions are seen to be

\[
\frac{1}{I} \sum_{i=1}^{I} a_i = \frac{1}{J} \sum_{j=1}^{J} b_{ij} = \frac{1}{K} \sum_{k=1}^{K} c_{ijk} = \frac{1}{L} \sum_{l=1}^{L} d_{l1} = \frac{1}{I} \sum_{i=1}^{I} a_{i1} = \frac{1}{L} \sum_{l=1}^{L} a_{i1l}
\]

\[
\frac{1}{J} \sum_{j=1}^{J} b_{dij} = \frac{1}{L} \sum_{l=1}^{L} b_{dijl} = 0.
\]

(3.1.2)
Table 3.1.1  Associated and floating subscripts in model 3.1.1

<table>
<thead>
<tr>
<th>Model term</th>
<th>Associated Subscripts</th>
<th>Floating Subscripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_i</td>
<td>i</td>
<td>None</td>
</tr>
<tr>
<td>b_ij</td>
<td>j</td>
<td>i</td>
</tr>
<tr>
<td>c_ijk</td>
<td>k</td>
<td>i,j</td>
</tr>
<tr>
<td>d_l</td>
<td>l</td>
<td>None</td>
</tr>
<tr>
<td>ad_{i,l}</td>
<td>i,l</td>
<td>None</td>
</tr>
<tr>
<td>bd_{i,j,l}</td>
<td>j,l</td>
<td>i</td>
</tr>
<tr>
<td>e_{i,j,k,l}</td>
<td>m</td>
<td>i,j,k,l</td>
</tr>
</tbody>
</table>

Constraints on the solution vector in the form defined by 3.1.2 will be the usual constraints employed to solve the normal equations for this model.

One consequence of using these constraints is that the coefficient matrix in the normal equations is block-diagonal. The blocks correspond to model terms, and the row and column size of each block is the number of degrees of freedom associated with that effect. Thus, the columns of the reduced $X$ matrix which correspond to different effects are orthogonal.
The notation used in stating rules for estimates of parameters, degrees of freedom and sums of squares in analysis of variance tables includes dots in the place of some subscripts on \( y \) to indicate an averaging over that subscript. For example

\[
\overline{y}_{j.k} = \frac{1}{I} \sum_{i=1}^{I} y_{ijk},
\]

where \( I \) is the upper limit of the \( i \) subscript values. These means will be called partial means.

For any given model term let \( F \) denote the set of floating subscripts and \( A \) denote the set of associated subscripts.

**Rule 1:** The degrees of freedom for the model term is

\[
\prod_{i \in F} L_i \prod_{j \in A} (L_j - 1),
\]

where \( L_k \) denotes the upper limit of the \( k \)th subscript in the model.

**Rule 2:** Estimates of the parameters in the model term are specified as linear combinations of partial means using the following device. Expand symbolically the product

\[
\prod_{i \in F} i \prod_{j \in A} (j - 1).
\]

In each term of this symbolic expansion insert a dot in the location of each subscript that appears in the model but not in the term of the symbolic expansion. (A one in the expansion specifies a dot in every position.) The parameters are now estimated as the linear
combination of partial means designated by corresponding coefficients in the symbolic expansion. For example, in a three-subscript model, the expansion \( ij - i \) yields \( ij - i \cdot i \cdot j \), which specifies the linear combination of partial means \( \bar{y}_{ij} - \bar{y}_{i} \).

Rule 3: The sum of squares relative to a model term \( m \) is obtained by summing the square of estimates of the parameters for that term and multiplying this sum of squares by the product \( p \) of the limits of all subscripts in the model which do not appear in the model term \( m \).

Suppose that \( X \) denotes the \( n \times p \) design matrix of the original model (i.e., the model is not restricted by the usual constraints) where \( n \) is the total number of observations and \( p \) the number of parameters. The next rule we discuss allows the direct computation of a generalized inverse of \( X'X \) for a given model. Note that this matrix is a \( p \times p \) symmetric matrix and that each of its rows (columns) can be identified in the model. For example, the rows of the \( X'X \) matrix for the model of the two-way classification with interaction given by

\[
y_{ijk} = \mu + a_i + b_j + (ab)_{ij} + e_{ijk},
\]

\( i=1,2,\ldots,I, \ j=1,2,\ldots,J, \) and \( k=1,2,\ldots,K \) correspond to the parameters \( \mu, a_1, a_2, \ldots, a_I, b_1, b_2, \ldots, b_J, ab_{11}, ab_{12}, \ldots, ab_{IJ} \) respectively.
Rule 4: The particular generalized inverse of $X'X$ that we construct using this rule is a lower triangular matrix denoted by $M^*$. For each term in the model, expand symbolically the product

$$
\sum_{i \in F} \prod_{j \in A} (j - 1).
$$

Each term of this expansion corresponds to the column of $M^*$ in which a nonzero element must appear in a row corresponding to this parameter. The values of these nonzero elements are obtained as follows. In the above symbolic expansion, change the subscripts appearing in each term to their upper limits. For example, $i$, $j$, and $ij$ should be replaced with $I$, $J$, and $IJ$, respectively, where $I$ and $J$ denote the upper limits of the subscripts $i$ and $j$ in the model. Then divide each term by the total number of observations $n$. The terms in this expansion are the values of the element of $M^*$ whose column positions in a row are determined by corresponding terms in the original expansion.

The following example will illustrate the use of Rule 4.

Example 3.1.1 Consider the model

$$
y_{ijk} = \mu + a_i + b_j + a_{ij} + c_k + e_{ijk},
$$

where $i=1,2$, $j=1,2$, and $k=1,2,3$. The generalized inverse of $X'X$ obtained by using Rule 4 is given below. Consider the symbolic expansion which corresponds to the model term $a_{ij}$ namely,
\[(i - 1)(j - 1) = ij - i - j + 1.\]

Replacing \(i\) and \(j\) with their upper limits and dividing by 12, we get the nonzero elements of the row of \(M^*\) which corresponds to \(ab_{ij}\). These are \(4/12, -2/12, -2/12,\) and \(1/12\), respectively. These values are inserted in column position of \(M^*\) corresponding to \(ab_{ij}, a_i, b_j,\) and \(\mu\) of the row of \(M^*\) corresponding to \(ab_{ij}\). All other elements of this row are set to zero.

\[
\begin{array}{cccccccccccc}
\mu & a_1 & a_2 & b_1 & b_2 & ab_{11} & ab_{12} & ab_{21} & ab_{22} & c_1 & c_2 & c_3 \\
\hline
\mu & \frac{1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & -\frac{1}{12} & \frac{2}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2 & -\frac{1}{12} & 0 & \frac{2}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_1 & -\frac{1}{12} & 0 & 0 & \frac{2}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_2 & -\frac{1}{12} & 0 & 0 & 0 & \frac{2}{12} & 0 & 0 & 0 & 0 & 0 & 0 \\
ab_{11} & \frac{1}{12} & -\frac{2}{12} & 0 & -\frac{2}{12} & 0 & \frac{4}{12} & 0 & 0 & 0 & 0 & 0 \\
ab_{12} & \frac{1}{12} & -\frac{2}{12} & 0 & 0 & -\frac{2}{12} & 0 & \frac{4}{12} & 0 & 0 & 0 & 0 \\
ab_{21} & \frac{1}{12} & 0 & -\frac{2}{12} & -\frac{2}{12} & 0 & 0 & \frac{4}{12} & 0 & 0 & 0 & 0 \\
ab_{22} & \frac{1}{12} & 0 & -\frac{2}{12} & 0 & -\frac{2}{12} & 0 & 0 & \frac{4}{12} & 0 & 0 & 0 \\
c_1 & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{12} & 0 & 0 & 0 \\
c_2 & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{12} & 0 & 0 \\
c_3 & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{12} & 0 \\
\end{array}
\]

\(M^* = \begin{array}{cccccccccccc}
\end{array}\)
For a given balanced design model, \( M X \hat{\gamma} \) is a particular solution to the normal equations
\[ X'X\hat{\gamma} = X'\gamma. \]

If we take the model of Example 3.1.1,
\[ M X \hat{\gamma} = \begin{bmatrix} \bar{y} \\ \bar{y}_{i..} - \bar{\bar{y}} \\ \bar{y}_{j..} - \bar{\bar{y}} \\ \bar{y}_{ij} - \bar{y}_{i..} - \bar{y}_{j..} + \bar{\bar{y}} \\ \bar{y}_{..k} - \bar{\bar{y}} \end{bmatrix} \]
where \( i = 1, 2; j = 1, 2; k = 1, 2, 3 \) and bar with \( \bar{\gamma} \) denotes mean.

Since \( M X \hat{\gamma} \) is a solution to the normal equations, we have
\[ X'X M X \hat{\gamma} = X'\gamma, \quad \text{for all } \gamma \]
which is equivalent to
\[ X'X M^* = X' \]
and
\[ X'X M^* X' = X'X. \]

Therefore, \( M^* \) is a generalized inverse of \( X'X \). Since \( M^* \) is non-singular and lower triangular matrix, its inverse, which we will denote by \( M \), can also be easily computed. However, it is more efficient, if \( M \) is constructed directly using a different rule when it is required to be computed.
Rule 5: For each term in the model, expand symbolically the following equation

\[ 1 + \sum_{i \in F} \sum_{j \in A} i (j+1) \]

Each term of this expansion corresponds to the column position of \( M \) in which a nonzero element should appear in a row corresponding to the parameter. To obtain the value of these nonzero elements, change the subscripts appearing in the parameter to their upper limits and form the product. Then divide the total number of observations, \( n \), by the resulting value. The answer is the nonzero value of the element of \( M \) whose column positions in a row are determined by the terms in the above expansion. Thus, every nonzero value in a row will be identical.

Example 3.1.2 will illustrate the use of Rule 5.

Example 3.1.2: Consider the nested model:

\[ y_{ijk} = \beta + a_i + ab_{ij} + c_k + \epsilon_{ijk} \]

where \( i = 1, 2, j = 1, 2, 3 \) and \( k = 1, 2 \). The matrix \( M \) corresponding to \( M^* \), the generalized inverse of \( XX \) computed using Rule 4, is constructed below. Consider the symbolic expansion corresponding to the model term \( ab_{ij} \) which is

\[ 1 + i(j+1) = 1 + i + ij \]

Each of the symbolic terms 1, \( i \) and \( ij \) is the column position of \( M \) in which a nonzero element appear in a row of \( ab_{ij} \). Next, replace
i and j with I and .J in the subscript of this term. Form the product 
(I*J in this case). Find the value \( \frac{n}{(1\times J)} \), \( \frac{12}{(2\times 3)} = 2 \). This value 
is to be inserted in the column position of \( \mu \), \( a_1 \), and \( ab_{1j} \) of the 
row \( ab_{ij} \). All other elements of this row are set to zero. The matrix 
\( M \) of Example 3.1.2 is given below

\[
M = \begin{bmatrix}
\mu & a_1 & a_2 & ab_{11} & ab_{12} & ab_{13} & ab_{21} & ab_{22} & ab_{23} & c_1 & c_2 \\
\mu & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & 6 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2 & 6 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
ab_{11} & 2 & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
ab_{12} & 2 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
ab_{13} & 2 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
ab_{21} & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
ab_{22} & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
ab_{23} & 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\
c_1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\
c_2 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6
\end{bmatrix}
\]

The computer algorithm for Rule 4 will be explained in Chapter 7. 
In the next section, we shall discuss properties and applications of 
\( M^* \).

3.2 Properties and Applications of \( M^* \)

In the previous section, we discussed a classical computational 
method for parameter estimation and obtaining sums of squares and
their degrees of freedom. In addition, a method for the computation of $M^*$ which is a generalized inverse matrix of $XX'$ was given.

We shall now discuss that some properties of the matrix $M^*$ in this section.

Assume that $X$ is an $n \times p$ balanced design matrix that one would obtain if there were one and only one observation in each of $n$ cells. The following properties of the matrix $M^*$ are stated here without proof.

1. $M^*$ is a generalized inverse matrix of $XX'$.

2. $M^*$ is a nonsingular and positive definite lower triangular matrix.

3. $(M^*)' = (M')^{-1}$.

4. $(M^*)'$ is also a generalized inverse matrix of $XX'$.

5. $XM^*X$ is symmetric idempotent and a projection matrix onto $C(X)$.

6. $X(M')^{-1}X = XM^*X$.

7. $(M^*)'X^*M^*$ is a $g_2$-generalized inverse matrix of $XX'$.

Let $D$ be a diagonal matrix whose elements are $\{d_i\}$ where $d_i$'s are nonnegative. The following is a useful result.

**Lemma 3.2.1** If $X'DX$ is of rank, $r$, then the rank of $M^*XDX$ is also $r$. 
Proof: \[ r = \text{rank}(X DX) \]
\[ \geq \text{rank}(X^* X DX) \]
\[ \geq \text{rank}(X^* X MX^* X DX) \]
\[ = \text{rank}(X DX) \]
\[ = r. \]

The following theorem is needed to prove a useful result concerning the characteristic roots of \( X^* X DX \).

**Theorem 3.2.1** If \( A \) and \( B \) are \( p \times p \) nonnegative definite matrices and if \( A \) is symmetric, then the characteristic roots of \( AB \) are nonnegative.

**Proof:** Since \( A \) is symmetric and nonnegative definite of rank \( r \) (\( \leq p \)). Let \( Q \) be a nonsingular matrix such that
\[
Q A Q = \begin{bmatrix}
I & 0 \\
0 & 0
\end{bmatrix},
\]
where \( I \) is the \( r \times r \) identity matrix. The characteristic roots of \( AB \) are the same as the characteristic roots of \( Q^{-1} B Q^{-1} \) and let
\[ C = Q^{-1} B Q^{-1}. \] Then, we have
\[
Q A B Q^{-1} = Q A Q^{-1} B Q^{-1}
\]
\[
= \begin{bmatrix}
I & 0 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]
\[
= \begin{bmatrix}
C_{11} & C_{12} \\
0 & 0
\end{bmatrix}.
\]
So the characteristic roots of $Q ABQ^{-1}$ are the values of $\lambda$ that satisfy
\[
\begin{vmatrix}
C_{11} - \lambda I & C_{12} \\
0 & -\lambda I
\end{vmatrix} = 0,
\]
which reduces to $|\lambda I| |C_{11} - \lambda I| = 0$. Thus, $\lambda$ is either zero or a characteristic roots of $C_{11}$. Next, we should prove that $C_{11}$ is a nonnegative definite matrix. Let
\[
B = \begin{pmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{pmatrix}
\quad \text{and} \quad
Q^{-1} = \begin{pmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{pmatrix}
\]
Since $B$ is nonnegative definite matrix,
\[
\begin{pmatrix}
y_1 \\
y_2
\end{pmatrix}
= \begin{pmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2
\end{pmatrix}
= y_1 B_{11} + y_2 B_{21},
\]
whence
\[
\begin{pmatrix}
y_1 B_{11} \bar{y}_1 + y_2 B_{21} \bar{y}_1 + y_1 B_{12} \bar{y}_2 + y_2 B_{22} \bar{y}_2
\end{pmatrix}
\geq 0, \text{ for } y_1 \text{ and } y_2. \quad (3.2.1)
\]
Since
\[
\begin{vmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{vmatrix}
= \begin{vmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{vmatrix}
\quad \text{and} \quad
\begin{vmatrix}
Q_{11} & Q_{21} \\
Q_{21} & Q_{22}
\end{vmatrix}
= \begin{vmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{vmatrix},
\]
we have
\[
C_{11} = Q_{11} B_{11} Q_{11}^T + Q_{12} B_{21} Q_{11}^T + Q_{11} B_{12} Q_{21}^T + Q_{12} B_{22} Q_{21}^T.
\]
Therefore, by 3.2.1
\[
\xi C_{11} \xi^T = \xi Q_{11} B_{11} Q_{11}^T \xi + \xi Q_{12} B_{21} Q_{11}^T \xi + \xi Q_{11} B_{12} Q_{21}^T \xi
+ \xi Q_{12} B_{22} Q_{21}^T \xi
= z_1 B_{11} z_1 + z_2 B_{21} z_1 + z_1 B_{12} z_2 + z_2 B_{22} z_2 \geq 0
\]
where \( z_1 = Q_{11}t \) and \( z_2 = Q_{21}t \), and \( t \) is an arbitrary vector. So, \( C_{11} \) is a nonnegative definite matrix hence, the characteristic roots of \( AB \) are nonnegative.

**Corollary 3.2.1** If \( A \) and \( B \) are \( p \times p \) symmetric and nonnegative definite matrices, then \( AB \) is also a nonnegative definite matrix.

**Proof:** The necessary and sufficient condition for a nonnegative definite matrix is that the characteristic roots should be greater than or equal to zero. By Theorem 3.2.1, \( AB \) is a nonnegative definite matrix.

The following lemma will prove that \( M^* XDX \) is a nonnegative definite matrix.

**Lemma 3.2.2** The characteristic roots of \( M^* XDX \) are nonnegative.

**Proof:** Since \( M^* \) is a positive definite matrix, it is also a nonnegative definite matrix. We can easily prove that its characteristic roots are greater than or equal to zero by using Theorem 3.2.1.

**Lemma 3.2.3** Let \( \lambda_k \) be a characteristic root of \( M^* XDX \), then \( \lambda_k \leq \max\{d_j\} \) for all \( j \).

**Proof:** Let \( X_r \) be the balanced design matrix whose elements are \(-1, 0, 1\) using usual constraints which were described in Equation 3.2.1. For example, when the model is
\[ y_{ij} = \mu + a_i + b_j + e_{ij}, \quad i = 1,2,3, \quad j = 1,2, \]

the constraints are

\[ a_1 + a_2 + a_3 = 0 \quad \text{and} \quad b_1 + b_2 = 0, \]

and matrix \( X_r \) is

\[
X_r = \begin{pmatrix}
1 & 1 & 0 & 1 \\
1 & 1 & 0 & -1 \\
1 & 0 & 1 & 1 \\
1 & 0 & 1 & -1 \\
1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1
\end{pmatrix}.
\]

We know that \( P_X = X(X'X)^{-1}X' = X_r(X_r'X_r)^{-1}X_r' \), because \( P_X \) is invariant with respect to the above constraints. From Hemmerle (1974), the maximum characteristic root of \( (X_r'X_r)^{-1}X_r'DX_r \) is less than or equal to \( \max\{d_j\} \). Thus,

\[
\max\{\lambda_j\} = \lim_{n \to \infty} \frac{\text{trace}(X'M'DX)^n}{n}^{1/n} \\
= \lim_{n \to \infty} \frac{\text{trace}(XM'XXD)^n}{n}^{1/n} \\
= \lim_{n \to \infty} \frac{\text{trace}(X_rDX_r)^n}{n}^{1/n} \\
= \lim_{n \to \infty} \frac{\text{trace}((X_r'X_r)^{-1}X_r'DX_r)^n}{n}^{1/n} \\
\leq \max\{d_j\}.
\]

If we write the fixed effects nonorthogonal analysis of variance model in matrix form as
\[ y = X_0 \hat{\beta} + e \]  

(3.2.2)

where \( y \) is the \( N \)-vector of observations;

\( X_0 \) is the \( N \times p \) design matrix with elements 0 and 1;

\( \hat{\beta} \) is the \( p \)-vector of parameters;

and

\( e \) is distributed \( N(0, \sigma^2 I) \),

then the normal equations for 3.2.2 are

\[ X_0' X_0 \hat{\beta} = X_0' y . \]  

(3.2.3)

As mentioned in Chapter 2, an equivalent set of equations to 3.2.3 is given by

\[ X D X \hat{\beta} = X D \hat{y} , \]  

(3.2.4)

where \( X \) is the \( n \times p \) balanced design matrix that one would obtain if there were one and only one observation in each cell of the \( n \) cells;

\( D \) is the \( n \times n \) diagonal matrix of cell frequencies \( d_i \);

\( \hat{y} \) is the \( n \)-vector of cell means.

Hemmerle's iterative method (Hemmerle 1974, 1976a,b) solves the system of linear Equations 3.2.4 for \( \hat{\beta} \) and uses the matrix \( X_r \). He assumes that he has reparameterized \( X \) using the usual summation restrictions, and that \( X_r \) is \( n \times p_0 \) and of full rank of \( p_0 \). We shall develop the Hemmerle's iterative method using \( M^* \) and \( X \) instead of using \( (X_r X_r)^{-1} \) and \( X_r \). Define

\[ E = (I - \frac{1}{c} M^* X DX) , \]
where $c$ is a constant which must be chosen such that $c > \max\{d_i\}$ and $I$ is the $p \times p$ identity matrix. Let

$$S_k = I + E + E^2 + \ldots + E^k$$

and

$$\hat{\beta}_k = S_k \frac{1}{c} M^* X D\hat{\gamma}^*,$$

where $\hat{\gamma}$ is the $n$-vector of cell means with empty cells represented by corresponding zero elements in this vector. We will show that $\hat{\beta}_k$ converges to a solution of Equation 3.2.4.

From Lemma 3.2.1 and 3.2.2, $\frac{1}{c} M^* X D\gamma$ has $p-r$ characteristic roots equal to zero and $E$ has $p-r$ roots equal to one. Also, if $\lambda$ is a root of $E$ we have that

$$0 < \lambda < 1.$$

Thus, the remaining $r$ roots of $E$ are nonnegative and less than one. We now consider

$$E^* = \lim_{k \to \infty} E^k.$$

Then, since the roots of $E^k$ are those of $E$ taken to the $k$th power, $E^*$ has $r$ characteristic roots of zero and $p-r$ roots of unity.

The following theorem will show the matrix series convergence.

**Theorem 3.2.2** If $E$ is nonnegative definite matrix whose maximum characteristic root is one and define

$$S_k = I + E + E^2 + \ldots + E^k$$
Then

\[ \lim_{k \to \infty} (I-E)S_k(I-E) = I-E . \]

**Proof:**

\[
(I-E)S_k(I-E) = (I-E)(I-E^{k+1}) = (I-E) - (I-E)E^{k+1} .
\]

To prove that \((I-E)E^{k+1}\) goes to zero as \(k \to \infty\), let \(\lambda\) be a characteristic root of \(E\) such that \(0 < \lambda < 1\). Then, \(Eu = \lambda u\), where \(u\) is a characteristic vector.

\[
(I-E)E^{k+1}u = (I-E)\lambda^{k+1}u = \lambda^{k+1}(u - Eu) = \lambda^{k+1}(1-\lambda)u .
\]

Thus, a characteristic root of \((I-E)E^{k+1}\) is \(\lambda^{k+1}(1-\lambda)\). For \(0 \leq \lambda \leq 1\), \(\lambda^{k+1}(1-\lambda)\) becomes zero as \(k \to \infty\). Therefore,

\[ \lim_{k \to \infty} (I-E)E^{k+1} = 0 \]

and

\[ \lim_{k \to \infty} (I-E)S_k(I-E) = I-E . \]

Next, we can prove \(E^*\) and \((I-E)^*\) are idempotent matrices.

*Lemma 3.2.4* \(E^*\) and \((I-E)^*\) are idempotent matrices.
Proof: To prove the lemma, it is enough to prove $E^*$ is idempotent matrix. Because if $E^*$ is idempotent matrix, then $I-E^*$ is also idempotent. Since $E^* = \lim_{k \to \infty} E^k = \lim_{k \to \infty} E^{2k}$,

$$E^* E^* = (\lim_{k \to \infty} E^k)(\lim_{k \to \infty} E^k) = E^k = \lim_{k \to \infty} E^k E^k = \lim_{k \to \infty} E^{2k} = E^*.$$  

Therefore, $E^*$ and $I-E^*$ are idempotent.

Next, we shall prove that $S_k(\frac{1}{c})M^*$ approaches a generalized inverse of $X^*DX$.

**Theorem 3.2.3** $S_k(\frac{1}{c})M^*$ approaches a generalized inverse of $X^*DX$ as $k \to \infty$.

**Proof:** Let $Q_k = S_k(\frac{1}{c})M^*$, so that

$$\frac{1}{c}M^* X^*DXQ_k = \frac{1}{c}M^* X^*DX S_k(\frac{1}{c})M^* DX.$$  

Now, since

$$\frac{1}{c}M^* DX = I-E$$  

and by Theorem 3.2.2,

$$(I-E)S_k(I-E) \to (I-E)$$ as $k \to \infty$,

we have that

$$\frac{1}{c}M^* X^*DXQ_k \to \frac{1}{c}M^* X^*DX$$ as $k \to \infty$.

$$\Rightarrow X^*DXQ_k \to X^*DX$$ as $k \to \infty$.

Since $M^*$ is nonsingular. Thus, $Q_k$ approaches a generalized inverse of $X^*DX$ as $k \to \infty$. 

Therefore, \( \hat{\beta}_k = (\frac{1}{c})S_kX^*D\tilde{y} \) approaches a solution, \( \hat{\beta} \), of the normal Equations 3.2.4. Although we may change the value of \( c \) to speed convergence, this iterative method converges more slowly than the conjugate gradient method in a larger parameter case (Golub and Nash 1982). We will introduce the conjugate and the modified conjugate gradient methods in Chapter 4 and Chapter 5.
4. THE METHOD OF CONJUGATE DIRECTIONS

The conjugate direction algorithm for minimizing a quadratic function was proposed by Hestenes and Stiefel (1952). One of the first algorithms which used conjugate gradient for solution of the general unconstrained minimization problem was published by Fletcher and Reeves (1964). This algorithm, with some relatively minor modifications, is still in use today. While the quasi-Newton methods require only first derivatives, which makes them easier to use than Newton's method, they do use an approximate Hessian matrix which will require a large amount of computer memory for storage when sizable problems are solved. The conjugate gradient method, which will be discussed later, is much less demanding in this respect and can be used when solving large problems more efficiently.

We will investigate the properties of the basic conjugate direction method and consider several modifications of the basic method for minimizing quadratic functions. One important modification features use of gradient directions which are designed to be mutually conjugate. We will introduce the basic properties of the conjugate direction method in the first two sections. A conjugate direction algorithm will be discussed in Section 3. The conjugate gradient method and the associated algorithm will be discussed in the last section of this chapter. Most of the theorems which appear in this chapter are proved in Hestenes (1980).
4.1 Basic Properties of Quadratic Functions

Assume $n$ is a fixed integer and $F$ is the quadratic function

$$F(x) = \frac{1}{2} x^T A x - h^T x + c$$

(4.1.1)

for $x \in \mathbb{R}^n$, where $A$ is a real symmetric positive definite $n \times n$-dimensional matrix, $h$ is a fixed $n$-dimensional vector, and $c$ is a scalar. The positive definiteness of $A$ ensures that a unique minimum exists for $F$ and is given by

$$x_o = A^{-1} h.$$

The point $x_o$ is the common center of a 1-parameter family of $(n-1)$-dimensional ellipsoids

$$F(x) = \gamma \ (\gamma > F(x_o)).$$

The following is a basic property of the positive definite function $F$.

**Theorem 4.1.1** The minimum points of $F$ on parallel lines in $\mathbb{R}^n$ lie on an $(n-1)$-plane $\widehat{\Pi}_{n-1}$ through the minimum point $x_o$ of $F$. The $(n-1)$-plane $\widehat{\Pi}_{n-1}$ is defined by the equation

$$p^T (Ax - h) = 0$$

(4.1.2)

where $p$ is a direction vector for these parallel lines. The vector $Ap$ is normal to $\widehat{\Pi}_{n-1}$.

**Proof:** Let the points $x_2$ and $\bar{x}_2$ be, respectively, the minimum points of $F$ on two parallel lines $L$ and $\bar{L}$ as shown in Figure 4.1.1.
The direction of these lines can be represented by a nonnull vector \( p \). At the minimum point \( x^* \) of \( F \) on \( L \) the gradient \( F'(x^*) = \nabla F(x^*) = -h \) of \( F \) is orthogonal to \( L \) and also to \( p \). The point \( x^* \) therefore satisfies Equation 4.1.2. Similarly \( \bar{x}_2 \) satisfies this equation. Equation 4.1.2 represents an \((n-1)\)-plane \( \mathbb{P}_{n-1} \) whose normal is \( A_p \). Inasmuch as \( Ax_o = h \) the minimum point \( x_o \) of \( F \) lies in \( \mathbb{P}_{n-1} \). Since \( \mathbb{P}_{n-1} \) is uniquely determined by the direction vector \( p \), it follows that the minimum points of \( F \) on all lines with direction \( p \) must lie in the \((n-1)\)-plane \( \mathbb{P}_{n-1} \).
As remarked above the vector \( Ap \) is orthogonal to \( \Pi_{n-1} \). We express this fact by saying that \( p \) is conjugate \((A\text{-orthogonal})\) to \( \Pi_{n-1} \) and that \( \Pi_{n-1} \) is conjugate to \( p \). In other words, the relation
\[
\langle p, Aq \rangle = 0 \quad (4.1.3)
\]
holds for every vector \( q \) in \( \Pi_{n-1} \). Since the vector \( q = \bar{x}_2 - \bar{x}_2 \) exhibited in Figure 4.1.1 is in \( \Pi_{n-1} \) we have the following result.

**Theorem 4.1.2** Given a nonnull vector \( p \), let \( \bar{x}_2 \) and \( \bar{x}_2 \) be, respectively, the minimum points of \( F \) on two lines \( L \) and \( \bar{L} \) whose direction is \( p \). The vector \( q = \bar{x}_2 - \bar{x}_2 \) is conjugate to \( p \) in the sense that the relation 4.1.3 holds.

The concept of conjugate directions \( p \) and \( q \) plays a very important role in minimization algorithms. When \( A \) is the identity matrix we have the usual orthogonality condition.

Referring to Figure 4.1.1 observe that the line \( L \) is given parametrically in the form \( x = x_1 + \alpha p \), where \( x_1 \) is a point on \( L \) and \( \alpha \) is a parameter ranging from \( -\infty \) to \( \infty \). If we set
\[
\tau_1 = -F'(x_1) = h - Ax_1,
\]
then, along the line \( x = x_1 + \alpha p \), we have
\[
F(x_1 + \alpha p) = F(x_1) - \alpha p^* \tau_1 + \frac{\alpha^2}{2} p^* A p \quad (4.1.4)
\]
This function of \( \alpha \) has a minimum value when \( \alpha = \frac{p^* \tau_1}{p^* A p} \).
Theorem 4.1.3  The minimum point $x_0$ of $F$ on the line $x = x_1 + \alpha p$ is given by the formula

$$x_2 = x_1 + \alpha p,$$

where

$$a = \frac{c}{d}, \quad c = p' r_1, \quad d = p' A p, \quad r_1 = -F'(x_1) = h - A x_1.$$

Proof: Setting $\alpha = 2a$ in formula 4.1.4, we obtain the relation

$$F(x_1 + 2ap) = F(x_1) - 2a(c - ad) = F(x_1).$$

The point $x_2 = x_1 + \alpha p$ is the midpoint of the line segment joining $x_1$ to $\hat{x}_1 = x_1 + 2ap$. Since $F(\hat{x}_1) = F(x_1)$, this line segment is a chord of the $(n-1)$-dimensional ellipsoid $F(x) = F(x_1)$ and $x_2$ is the midpoint of this chord.

As a dual of Theorem 4.1.1 we have

Theorem 4.1.4  The minimum point of $F$ on parallel $(n-1)$-planes lie on a line $L$ conjugate to these hyperplanes and passing through the minimum point $x_0$ of $F$. In other words, if $q$ is a given nonnull vector, then for every real number $\rho$ the minimum point $x_1$ of $F$ on the $(n-1)$-plane

$$\Pi_{n-1} : q \cdot x = \rho$$

lies on the line

$$L : x = x_0 + \alpha A^{-1} q.$$
passing through the minimum point $x_0$ of $F$ in the direction $p = A^{-1}q$. The vector $p$ or equivalently the line $L$ is conjugate to $\Pi_{n-1}$.

**Proof:** The gradient $F'(x_1) = Ax_1 - h$ at the minimum point $x_1$ of $F$ on $\Pi_{n-1}$ is orthogonal to $\Pi_{n-1}$ and must be a multiple of the normal $q$ of $\Pi_{n-1}$. There is accordingly a number $\alpha_1$ such that

$$Ax_1 - h = \alpha_1 q \quad \text{or} \quad x_1 = A^{-1}h + \alpha_1 A^{-1}q = x_0 + \alpha_1 A^{-1}q,$$

the last equality holding since $A^{-1}h$ is the minimum point $x_0$ of $F$. It follows that $x_1$ is on the line $L$. The direct vector $p = A^{-1}q$ has the property that $Ap = q$ is orthogonal to $\Pi_{n-1}$. The vector $p$ is therefore conjugate to $\Pi_{n-1}$, as was to be proved.

As an extension of Theorem 4.1.4, we have the following theorem.

**Theorem 4.1.5** The minimum points of $F$ on parallel $k$-planes lie on $(n-k)$-plane conjugate to these $k$-planes and passing through the minimum point $x_0$ of $F$. That is, given a set of $n-k$ linearly independent vectors $q_1, \ldots, q_{n-k}$, then for every set of real
numbers $\rho_1, \ldots, \rho_{n-k}$ the minimum point $x_1$ of $F$ on the $k$-plane

$\Pi_k : q_i x = \rho_i \quad (i = 1, \ldots, n-k)$

lies on the $(n-k)$-plane

$\hat{\Pi}_{n-k} : x = x_0 + \alpha_1 A^{-1} q_1 + \ldots + \alpha_{n-k} A^{-1} q_{n-k}$

passing through the minimum point $x_0$ of $F$. The vectors

$\mathbf{p}_1 = A^{-1} q_1', \ldots, \mathbf{p}_{n-k} = A^{-1} q_{n-k}'$ are conjugate to $\Pi_k$ so that $\hat{\Pi}_{n-k}$ is conjugate to $\Pi_k$.

**Proof:** The proof is similar to that of Theorem 4.1.4. At the minimum point $x_1$ of $F$ on $\Pi_k$ the gradient $F'(x_1) = Ax_1 - h$ is orthogonal to $\Pi_k$ and is a linear combination

$Ax_1 - h = \alpha_1 q_1 + \ldots + \alpha_{n-k} q_{n-k}$

of the normals $q_1, \ldots, q_{n-k}$ of $\Pi_k$. Since $x_0 = A^{-1} h$ it follows

$x_1 = x_0 + \alpha_1 A^{-1} q_1 + \ldots + \alpha_{n-k} A^{-1} q_{n-k}$

is in the $(n-k)$-plant $\hat{\Pi}_{n-k}$. The vectors $\mathbf{p}_1 = A^{-1} q_1', \ldots, \mathbf{p}_{n-k} = A^{-1} q_{n-k}'$ have the property that the vectors $A\mathbf{p}_1 = q_1', \ldots, A\mathbf{p}_{n-k} = q_{n-k}'$ are orthogonal to $\Pi_k$ so that $\mathbf{p}_1, \ldots, \mathbf{p}_{n-k}$ are conjugate to $\Pi_k$.

Since $\mathbf{p}_1, \ldots, \mathbf{p}_{n-k}$ generate the vectors in $\hat{\Pi}_{n-k}$, the $(n-k)$-plane
is conjugate to \( \Pi_k \), as was to be proved.

We will discuss the conjugate direction method to minimize the quadratic function in the next section.

4.2 Minimization of a Quadratic Function

We have discussed general properties of the positive quadratic function

\[ F(x) = \frac{1}{2} x' A x - h' x + c. \]

we showed that on each k-plane \( \Pi_k \) the function \( F \) has a unique minimum point. For reasons that will become apparent as we proceed, We designate the minimum point of \( F \) on \( \Pi_k \) by \( x_{k+1} \). The (n-k)-plane \( \Pi_{n-k} \) through \( x_{k+1} \) conjugate to \( \Pi_k \) contains the minimum point \( x_0 \) of \( F \). The present section is devoted mainly to obtaining a simple formula for the minimum point \( x_0 \) of \( F \).

**Definition 4.2.1** A set of nonnull vectors \( p_1, \ldots, p_k \) in \( \Pi_k \) is said to be mutually conjugate with respect to \( A \) if

\[ p_i' A p_j = 0 \quad (i \neq j, i = 1, 2, \ldots, k). \quad (4.2.1) \]

Note that

\[ d_i = p_i' A p_i > 0 \quad (i = 1, 2, \ldots, k) \quad (4.2.2) \]

because \( p_i \neq 0 \) and \( A \) is positive definite. That such a set of vectors exists is apparent because eigenvectors of \( A \) have this property. When \( A \) is the identity matrix, we have the usual orthogonal condition. A set of nonnull mutually conjugate vectors constitute a conjugate system.

We begin with a point \( x_1 \) and construct nonnull mutually conjugate vectors which in turn define a k-plane.
\[ \Pi_k : \quad x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k \]

Observe that we continue to designate a k-plane by the formula for its points. As a first result we have

**Theorem 4.2.1** Let

\[ \Pi_k : \quad x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k \]

be the k-plane through a selected point \( x_1 \) determined by a conjugate system \( p_1, \ldots, p_k \). The minimum point \( x_{k+1} \) of \( F \) on \( \Pi_k \) is given by the formula

\[ x_{k+1} = x_1 + a_1 p_1 + \ldots + a_k p_k \quad (4.2.3) \]

where

\[ a_i = \frac{c_i}{d_i}, \quad c_i = p_i r_i, \quad d_i = p_i A p_i \quad (i=1, \ldots, k) \quad (4.2.4) \]

and

\[ r_1 = -F'(x_1) = h - Ax_1 \quad (4.2.5) \]

is the residual of \( F \) at \( x_1 \). The residual \( r_{k+1} = -F'(x_{k+1}) \) of \( F \) at \( x_{k+1} \) is given by

\[ r_{k+1} = r_1 - a_1 A p_1 - \ldots - a_k A p_k \quad (4.2.6) \]

and is orthogonal to \( \Pi_k \) so that

\[ p_i r_{k+1} = 0 \quad (i = 1, \ldots, k) \quad (4.2.7) \]

The minimum value of \( F \) on \( \Pi_k \) is

\[ F(x_{k+1}) = F(x_1) - \frac{1}{2} (a_1 c_1 + \ldots + a_k c_k) \quad (4.2.8) \]

**Proof:** Since the minimum point

\[ x_{k+1} = x_1 + a_1 p_1 + \ldots + a_k p_k \]
of $F$ on $\Pi_k$ is characterized by the fact that its residual $r_{k+1} = - F'(x_{k+1})$ is orthogonal to $\Pi_k$ and hence, to the vectors $p_1, \ldots, p_k$ in $\Pi_k$, as stated in Equation 4.2.7. Inasmuch as

$$r_{k+1} = h - Ax_{k+1} = h - A(x_1 + a_1 p_1 + \ldots + a_k p_k)$$

the formula (4.2.6) for $r_{k+1}$ holds. Combining 4.2.6 and 4.2.7, with the conjugacy relations 4.2.1 we find that, for $i = 1, \ldots, k$

$$0 = p_i r_{k+1} = p_i r_1 - \sum_{j=1}^{k} a_j p_i A p_j = c_i - a_i d_i$$

so that $a_i$ is given by formula 4.2.4. To obtain 4.2.8 we use the identity

$$F(x + p) - F(x) = p^T F'(x) + 1/2 p^T A p$$

with $x = x_{k+1}$ and

$$p = x_1 - x_{k+1} = - a_1 p_1 - \ldots - a_k p_k.$$ 

By (4.2.7), $p$ is orthogonal to $F'(x_{k+1}) = - r_{k+1}$, so that

$$F(x_1) - F(x_{k+1}) = p^T F'(x_{k+1}) + 1/2 p^T A p$$

$$= 1/2 \sum_{i,j=1}^{k} a_i a_j p_i A p_j$$

$$= 1/2 \sum_{i=1}^{k} a_i^2 d_i = 1/2 \sum_{i=1}^{k} a_i c_i$$

by virtue of the conjugacy relations 4.2.1. This completes the proof of Theorem 4.2.1.
Corollary 4.2.1  

The minimum point $x_{k+1}$ of $F$ on the $k$-plane

$$
\Pi_k : \quad x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k
$$

is the point of intersection of $\Pi_k$ with the $(n-k)$-plane

$$
\hat{\Pi}_{n-k} : \quad \bar{P}_j F'(x) = \bar{P}_j (Ax - h) = 0 \quad (j = 1, \ldots, k).
$$

Proof:  
This follows because if $x$ is in $\Pi_k$, the equation

$$
\bar{P}_j F'(x) = \bar{P}_j \left[ A(x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k) - h \right]
$$

$$
= - \bar{P}_j x_1 + \alpha_j \bar{d}_j = 0
$$

holds if and only if $\alpha_j = a_j$, where $a_j$ is given by Equation 4.2.4.

The following theorem is basic in the development of the conjugate gradient algorithm introduced in the next section.

Theorem 4.2.2  
Let $p_1, \ldots, p_m$ be a conjugate system. For a given point $x_1$, let $x_2, x_3, \ldots, x_{m+1}$ be the points defined recursively by the condition that for each $k$, $1 \leq k \leq m$, the point $x_{k+1}$ minimizes $F$ on the line

$$
x = x_k + \alpha p_k. \quad (4.2.9)
$$

Then $x_{k+1}$ and the residual $r_{k+1} = - F'(x_{k+1})$ are given by

$$
x_{k+1} = x_k + a_k p_k, \quad r_{k+1} = r_k - a_k A p_k, \quad (4.2.10)
$$

where $r_k = - F'(x_k)$ and

$$
a_k = \frac{c_k}{d_k}, \quad c_k = \bar{P}_k r_k, \quad d_k = \bar{P}_k A p_k \quad (4.2.11)
$$
The point \( x_{k+1} \) minimizes \( F \) on the \( k \)-plane \( \Pi_k : \quad x = x_1 + \alpha_1 p_1 + \cdots + \alpha_k p_k \), and we have the relations

\[
P_k r_j = c_k \quad (j \leq k \leq m) , \quad p_k r_j = 0 \quad (k, j \leq m+1). \quad (4.2.12)
\]

**Proof:** Since \( x_{k+1} \) minimizes \( F \) on the 1-plane \( 4.2.9 \), it follows from Theorem 4.2.1, with \( x_k \) playing the role of \( x_1 \), that \( x_{k+1} \) and \( r_{k+1} \) are given by \( 4.2.10 \) and \( 4.2.11 \) and that

\[
P_k r_{k+1} = 0 \quad (k = 1, \ldots, m).
\]

In view of the conjugacy relations \( p_k A p_j = 0 \) \((k \neq j)\) we have

\[
P_k r_{j+1} = p_k (r_j - a_j A p_k) = p_k r_j \quad (j \neq k, j = 1, 2, \ldots, m).
\]

Consequently,

\[
P_k r_1 = P_k r_2 = \cdots = p_k r_{k-1} = p_k r_k = c_k \quad (k \leq m),
\]

\[
P_k r_{m+1} = \cdots = p_k r_{k+2} = p_k r_{k+1} = 0 \quad (k < m).
\]

This proves relations \( 4.2.12 \). Since \( c_j = p_j r_1 \), it follows from Theorem 4.2.1 that \( x_{k+1} \) minimizes \( F \) on the \( k \)-plane \( \Pi_k \).

As a converse of Theorems 4.2.1 and 4.2.2, we have

**Theorem 4.2.3** Let \( p_1, \ldots, p_m \) be linearly independent vectors. For a selected initial point \( x_1 \), let \( x_2, x_3, \ldots, x_{m+1} \) be points defined by the formulas

\[
x_{k+1} = x_k + a_k p_k = x_1 + a_1 p_1 + \cdots + a_k p_k \quad (k=1,\ldots,m) \quad (4.2.13)
\]
where \( a_1, \ldots, a_m \) are nonzero real numbers. If, for \( k = 1, \ldots, m \), the point \( x_{k+1} \) minimizes \( F \) on the \( k \)-plane

\[
\Pi_k : \quad x = x_1 + a_1 p_1 + \ldots + a_k p_k,
\]

then the vectors \( p_1, \ldots, p_m \) are mutually conjugate and the relations described in Theorems 4.2.1 and 4.2.2 hold.

Proof: Because \( x_{k+1} \) minimizes \( F \) on the \( k \)-plane \( \Pi_k \), the residual

\[
F_{k+1}' = -F'(x_{k+1})
\]

is orthogonal to \( \Pi_k \) and hence, to \( p_1, \ldots, p_k \) so that

\[
p_j r_{k+1} = 0 \quad (j \leq k \leq m).
\]

Hence, if \( j < k \), we have \( p_j r_k = 0 \) and

\[
0 = p_j r_{k+1} = p_j (r_k - a_k p_k) = -a_k p_j p_k.
\]

Since \( a_k \neq 0 \) we have \( p_j p_k = 0 \) \((j < k \leq m)\). The vectors \( p_1, \ldots, p_m \) are therefore mutually conjugate, as was to be proved.

Suppose that the vectors \( p_1, \ldots, p_m \) described in Theorem 4.2.3 are mutually conjugate. Suppose further that, in computing the minimum point

\[
x_{k+1} = x_1 + a_1 p_1 + \ldots + a_k p_k = x_k + a_k p_k
\]

of \( F \) on \( \Pi_k \), we make an error in evaluating \( a_k \) and obtain instead a value

\[
\hat{a}_k = a_k + e_k.
\]
This yields the point

$$\hat{x}_{k+1} = x_k + \hat{a}_k p_k + \ldots + \hat{a}_k p_k = \hat{x}_k + \hat{a}_k p_k = x_{k+1} + \epsilon_{k+1}$$

where $\hat{x}_1 = x_1$ and

$$\epsilon_{k+1} = e_1 p_1 + \ldots + e_k p_k = \epsilon_k + e_k p_k , \quad \epsilon_1 = 0 .$$

Observe that the residual $r_{k+1} = - F (x_{k+1})$ and $\hat{r} = - F (\hat{x}_{k+1})$ are connected by the formula

$$\hat{r}_{k+1} = r_{k+1} + A \epsilon_{k+1} = r_{k+1} - e_k p_k - \ldots - e_k p_k .$$

Since $p_k r_{k+1} = 0$ and $p_k p_j = 0$ ($j < k$), we have

$$p_k \hat{r}_{k+1} = - e_k d_k , \quad d_k = p_k p_k , \quad e_k = - p_k \hat{r}_{k+1} / d_k .$$

We have accordingly the correction formulas

$$\hat{x}_1 = x_1 , \quad \hat{x}_{k+1} = \hat{x}_k + \hat{a}_k p_k , \quad \hat{r}_{k+1} = \hat{r}_k - \hat{a}_k \epsilon_k ,$$

$$\epsilon_1 = 0 , \quad \epsilon_{k+1} = \epsilon_k + e_k p_k , \quad d_k = p_k p_k , \quad e_k = - p_k \hat{r}_{k+1} / d_k ,$$

$$x_{k+1} = \hat{x}_{k+1} - \epsilon_{k+1}$$

for obtaining the true minimum point $x_{k+1}$ of $F$ on $\Pi_k$.

The results given in Theorem 4.2.1 and 4.2.2 are illustrated in the following example for the case $n = 4$.

**Example 4.2.1**

Consider the case in which

$$A = \begin{pmatrix}
  1 & 2 & -1 & 1 \\
  2 & 5 & 0 & 2 \\
 -1 & 0 & 6 & 0 \\
  1 & 2 & 0 & 3
\end{pmatrix}$$
\[ h = \begin{bmatrix} 0 \\ 2 \\ -1 \\ 1 \end{bmatrix}, \quad x_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \tau_1 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \]

where \( \tau_1 = h - Ax_1 = -F(x_1) \). The minimum point \( F \) is

\[ x_0 = (-65, 24, -11, 6). \]

Let \( P_1, P_2, P_3, P_4 \) be the column vectors of the matrix

\[ P = \begin{bmatrix} -1 & -6 & -30 & -20 \\ 0 & 2 & 12 & 10 \\ 0 & -1 & -6 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \]

In view of the computations

\[ AP = \begin{bmatrix} -1 & 0 & 0 & 0 \\ -2 & 0 & 0 & 10 \\ 1 & 0 & -6 & 20 \\ -1 & 1 & -6 & 0 \end{bmatrix}, \quad P'AP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 36 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, \]

the off-diagonal elements \( p_{jk} \) of \( P'AP \) are zero so that the vectors \( P_1, P_2, P_3, P_4 \) are mutually conjugate. The numbers \( d_i = P_i'AP_i, \quad c_i = P_i'\tau_1, \quad a_i = c_i/d_i \quad (i = 1, 2, 3, 4) \) are

\[ d_1 = 1, \quad c_1 = 1, \quad a_1 = 1; \]
\[ d_2 = 1, \quad c_2 = 6, \quad a_2 = 6; \]
\[ d_3 = 36, \quad c_3 = 30, \quad a_3 = 5/6; \]
\[ d_4 = 100, \quad c_4 = 20, \quad a_4 = 1/5. \]
By a simple computation it is seen that the points
\[ x_{k+1} = x_1 + a_1 p_1 + \ldots + a_k p_k = x_k + a_k p_k \ (k=1,2,3,4) \]
described in Theorem 4.2.1 are
\[
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix},
\begin{bmatrix}
-36 \\
12 \\
6 \\
6 \\
6
\end{bmatrix},
\begin{bmatrix}
-61 \\
22 \\
-6 \\
6 \\
-6
\end{bmatrix},
\begin{bmatrix}
-65 \\
24 \\
-65 \\
24 \\
24
\end{bmatrix} = x_0.
\]
The corresponding residuals \( r_k = -F(x_k) = h - Ax_k \) are
\[
\begin{bmatrix}
0 \\
2 \\
-1 \\
1
\end{bmatrix},
\begin{bmatrix}
0 \\
2 \\
-1 \\
-5
\end{bmatrix},
\begin{bmatrix}
0 \\
2 \\
4 \\
0
\end{bmatrix},
\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}.
\]

It is easily verified that for \( k = 1, 2, 3, 4 \)
\[ c_k = p_k \cdot r_j = p_k \cdot r_{k-1} \quad (j \leq k), \]
\[ p_j r_{k+1} = 0 \quad (j \leq k). \]

Because \( r_2 \) is orthogonal to \( p_1 \), the point \( x_2 \) minimizes \( F \) on the line
\[ \Pi_1 : \quad x = x_1 + \alpha p_1. \]

The three plane \( \hat{\Pi}_3 \) through \( x_2 \) conjugate to \( p_1 \) is
\[ \hat{\Pi}_3 : \quad \hat{p}_1 A (x-x_2) = -x_{(1)} - 2x_{(2)} + x_{(3)} + x_{(4)} = 0 \]
and contains the points \( x_3, x_4, \) and \( x_5 \). The minimum point \( x_3 \)
of \( F \) on the line \( x = x_2 + \alpha p_2 \) also minimizes \( F \) on the 2-plane
\[ \Pi_2 : \quad x = x_1 + \alpha_1 p_1 + \alpha_2 p_2. \]
because \( \pi_3 \) is orthogonal to \( p_1 \) and \( p_2 \). The 2-plane \( \hat{\Pi}_2 \) through \( x_3 \) conjugate to \( p_1 \) and \( p_2 \) is

\[
\hat{\Pi}_2 : \quad p_1 A(x - x_3) = 0, \quad p_2 A(x - x_3) = 0,
\]

that is,

\[
\hat{\Pi}_2 : \quad -x(1) - 2x(2) + x(3) - x(4) = 0, \quad x(4) - 6 = 0.
\]

The 2-plane \( \hat{\Pi}_2 \) passes through \( x_4 \) and \( x_5 \). The residual \( \pi_4 \) of the minimum point \( x_4 \) of \( F \) on the line \( x = x_3 + \alpha p_3 \) is orthogonal to \( p_1 \) and \( p_2 \) as well to \( p_3 \) so that \( x_4 \) minimizes \( F \) on the 3-plane

\[
\pi_3 : \quad x = x_1 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3.
\]

The conjugate 1-plane \( \hat{\Pi}_1 \) to this 3-plane through \( x_4 \) is the line

\[
\hat{\Pi}_1 : \quad p_i (x - x_4) = 0 \quad (i = 1, 2, 3),
\]

so that

\[
\hat{\Pi}_1 : \quad -x(1) - 2x(2) + x(3) - x(4) = 0, \quad x(4) - 6 = 0,
\]

\[
-6x(3) - 6x(4) - 30 = 0.
\]

This line is given parametrically by \( x = x_4 + \alpha p_4 \). The minimum point \( x_5 \) of \( F \) on this line also minimizes \( F \) on the 4-plane

\[
\pi_4 : \quad x = x_1 + \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \alpha_4 p_4,
\]
which is the complete 4-dimensional space $E^4$. Observe that we obtained the minimum point $x_5$ of $F$ by successive minimization of $F$ along lines. First, we minimized $F$ on the line $x = x_1 + \alpha p_1$ to obtain the point $x_2$; then along the line $x = x_2 + \alpha p_2$ to obtain the point $x_3$; next along the line $x = x_3 + \alpha p_3$ to obtain the point $x_4$; and finally along the line $x = x_4 + \alpha p_4$ to obtain the minimum point $x_5 = x_0$ of $F$. This process is called the method of conjugate directions and will be discussed in the next section.

4.3 Method of Conjugate Directions (CD-Algorithm)

We now turn to specific computational procedures for minimizing a positive definite quadratic function

$$ F(x) = \frac{1}{2} x' A x' - h' x + c. $$

These procedures consist of minimizing $F$ successively along lines. If these lines are mutually conjugate, the procedure is called a conjugate direction method (CD-method) for finding the minimum point $x_0 = A^{-1}h$ of $F$. By virtue of Theorem 4.2.2, a CD-method terminates in $m \leq n$ steps, if no round off errors occur. This fact also follows from Theorem 4.1.1, as can be seen from the following geometrical description of a CD-method.

Select a point $x_1$ and a line $L_1$ through $x_1$ in a direction $p_1$. Find the minimum point $x_2$ of $F$ on $L_1$. Construct the $(n-1)$-plane $\hat{n}_{n-1}$ through $x_2$ of $F$ is in $\hat{n}_{n-1}$. Consequently, our next search can be limited to $\hat{n}_{n-1}$ so that we have reduced the
dimensionality of our space of search by one. We now repeat the process, restricting ourselves to the \((n-1)\)-plane \(\hat{\Pi}_{n-1}\). We select a line \(L_2\) in \(\hat{\Pi}_{n-1}\) through \(x_2\) in a direction \(p_2\) and obtain the minimum point \(x_3\) of \(F\) on \(L_2\). Next construct the \((n-2)\)-plane \(\hat{\Pi}_{n-2}\) in \(\hat{\Pi}_{n-1}\) through \(x_3\) and conjugate to \(p_2\). By Theorem 4.1.1 with \(\hat{\Pi}_{n-1}\) playing the role of \(E^n\), the minimum point \(x_0\) of \(F\) is in the \((n-2)\)-plane \(\hat{\Pi}_{n-2}\). So that we can limit our search to \(\hat{\Pi}_{n-2}\).

Again, the dimension of our space of search has been reduced by one. Through \(x_3\) we pass a line \(L_3\) in \(\hat{\Pi}_{n-2}\) in a direction \(p_3\) and find the minimum point \(x_4\) of \(F\) on \(L_3\). Because \(x_0\) is in the \((n-3)\)-plane \(\hat{\Pi}_{n-3}\) in \(\hat{\Pi}_{n-2}\) through \(x_4\) conjugate to \(p_3\), we next find the minimum point \(x_5\) on a line \(L_4\) in \(\hat{\Pi}_{n-3}\) through \(x_4\) in a direction \(p_4\). Proceeding in this manner we reduce the dimensionality of our space by one at each step. At the \(n\)th step our space of search is a line \(\hat{\Pi}_1\) through \(x_0\) so that the minimum point \(x_{n+1}\) of \(F\) on \(\hat{\Pi}_1\) is the minimum point \(x_0\). Of course, on rare occasions we have \(x_{m+1} = x_0\) at an \(m\)th step \((m < n)\), in which case we can terminate in \(m < n\) steps.

The algorithm just described can be put in another form which does not involve the \((n-k)\)-planes \(\hat{\Pi}_{n-k}\) \((k = 1, \ldots, n-1)\) explicitly. These \((n-k)\)-planes are used to generate a set of mutually conjugate vectors \(p_1, \ldots, p_n\). The requirement that \(p_{k+1}\) be in \(\hat{\Pi}_{n-k}\) is equivalent to the requirement that \(p_{k+1}\) be conjugate to \(p_1, \ldots, p_k\). Accordingly our algorithm can be restated as follows:
Select a point \( x_1 \) and a direction \( p_1 \). Find the minimum point \( x_2 \) of \( F \) on the line \( x = x_1 + \alpha p_1 \). Next, select a direction \( p_2 \) conjugate to \( p_1 \) and determine the minimum point \( x_3 \) of \( F \) on the line \( x = x_2 + \alpha p_2 \). Having obtained the point \( x_k \) select a direction \( p_k \) conjugate to \( p_1, \ldots, p_{k-1} \) and find the minimum point \( x_{k+1} \) of \( F \) on the line \( x = x_k + \alpha p_k \). The point \( x_{n+1} \) obtained in this manner minimize \( F \).

This algorithm, which we call the method of conjugate directions (CD-method) can be formalized as follows:

Algorithm 4.3.1  Select an initial estimate \( x_1 \) of the minimum point \( x_o \) of \( F \), compute \( \varepsilon_1 = \nabla^T x_1 = h - Ax_1 \) and select an initial direction \( p_1 \neq 0 \).

1. \( c_k = p_k \varepsilon_k \),
2. \( d_k = p_k^T A p_k \),
3. \( a_k = c_k / d_k \),
4. \( x_{k+1} = x_k + a_k p_k \),
5. \( \varepsilon_{k+1} = \varepsilon_k - a_k A p_k \),
6. Select a nonnull vector \( p_{k+1} \) conjugates to \( p_1, \ldots, p_k \) such that \( p_j^T A p_{k+1} = 0 \) \( (j = 1, \ldots, k) \).

Terminate at \( m^{th} \) step if \( \varepsilon_{m+1} = -\nabla^T (x_{m+1}) = 0 \). Then, \( m \leq n \) and \( x_{m+1} \) is the minimum point \( x_o \) of \( F \).
The formula for $r_{k+1}$ given in 5 in Algorithm 4.3.1 can be replaced by the equivalent formula

$$r_{k+1} = F'(x_{k+1}) = x \Delta x_{k+1}.$$ 

In the Algorithm 4.3.1, the scale factor $a_k$ is chosen so that

$$p_k x_{k+1} = p_k x_k - a_k p_k A p_k = 0.$$  \hspace{1cm} (4.3.1)

This equation signifies that the gradient $F'(x_{k+1}) = -r_{k+1}$ is orthogonal to $p_k$. Consequently, $x_{k+1}$ minimizes $F$ on the line $x = x_k + a p_k$. By Theorem 4.2.2, the point $x_{k+1}$ also minimize $F$ on the k-plane $\Pi_k: x = x_k + a_1 p_1 + \cdots + a_k p_k$

There is a first integer $m$ such that the $m$-plane $\Pi_m$ contains $x_o$. In this case, $x_o$ as well as $x_{m+1}$ minimizes $F$ on $\Pi_m$ so that $x_{m+1} = x_o$ and $r_{m+1} = -F'(x_{m+1}) = 0$. The conjugate direction Algorithm 4.3.1 therefore terminates in $m \leq n$ steps. If $m < n$ and the algorithm is continued we have $x_{m+1} = x_{m+2} = \cdots = x_{n+1} = x_o$.

If roundoff errors occur, the residual $r_{n+1} = -F'(x_{n+1})$ may not be zero or nearly zero as it should be if $x_{n+1}$ is to be accepted as a good estimate of the minimum point $x_o$ of $F$. If it turns out that $x_{n+1}$ is not a good estimate of $x_o$, the algorithm can be repeated with $x_{n+1}$ as the new initial point $x_{n+1}$. Normally a repetition of the algorithm will result in a satisfactory estimate of $x_o$.
The basic relations in the CD-method are given in the following theorem.

**Theorem 4.3.1** The directions \( p_1, p_2, \ldots, p_n \) are mutually conjugate. The negative gradient \( r_k = -F'(x_k) \) of \( F \) at \( x_k \) is orthogonal to \( p_1, \ldots, p_{k+1} \) and the inner product of \( p_k \) with each of the residual \( r_1, \ldots, r_k \) is the same. That is

\[
\begin{align*}
 p_j p_k &= 0 \quad (j \neq k) , \\
p_j r_k &= 0 \quad (j = 1, \ldots, k-1) , \\
p_k r_j &= c_k \quad (j = 1, \ldots, k) .
\end{align*}
\]

This result is an immediate consequence of Theorem 4.2.2. In view of 4.3.4, we have \( c_k = p_k r_k \) so that step 1 in Algorithm 4.3.1 can be replaced by

\[
c_k = p_k r_k .
\]

When this formula for \( c_k \) is used, the estimates \( x_2, x_3, \ldots, x_{m+1} \) can be obtained without computing the residual \( r_2, r_3, \ldots, r_m \). Observe further that if we use \( x_j \) as the point \( x_1 \) in Theorem 4.2.2, we obtain the relation

\[
F(x_j) - F(x_{k+1}) = 1/2(a_j c_j + \ldots + a_k c_k) \geq 0 \quad (4.3.5)
\]

provided that \( j \leq k \). If \( j = k \) we have

\[
F(x_k) - F(x_{k+1}) = \frac{a_k c_k}{2} \geq 0 \quad (4.3.6)
\]
The geometric properties of the CD-algorithm are summarized in the following:

**Theorem 4.3.2** The point $x_{k+1}$ minimizes $F$ on the $k$-plane

$$
\Pi_k : \quad x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k .
$$

The $k$-plane $\Pi_k$ cuts the $(n-1)$-dimensional ellipsoid $F(x) = \gamma$ ($\gamma > F(x_1)$) in a $(k-1)$-dimensional ellipsoid $E_{k-1}$ whose center is $x_{k+1}$. The points $x_1, x_2, \ldots, x_{k+1}$ are in $\Pi_k$. The points $x_k+2 = x_{k+3} = \ldots = x_{n+1} = x_0$ are in the $(n-k)$-plane

$$
\Pi_{n-k} : \quad p_j A(x - x_{k+1}) = 0 \quad (j = 1, \ldots, k)
$$

through $x_{k+1}$ conjugate to $\Pi_k$.

**Proof:** Let $x_1$ be a fixed point in $\Pi_k$. If $x$ is another point in $\Pi_k$, $x$ is expressible in the form

$$
x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k
$$

(4.3.7)

for a suitable choice of parameters $\alpha_1, \ldots, \alpha_k$. Let $P$ be the matrix whose column vectors are $p_1, \ldots, p_k$. Then 4.3.7 can be put in the form

$$
x = x_1 + Py ,
$$

where $y$ is $k$-dimensional column vector whose components are $\alpha_1, \ldots, \alpha_k$. On the $k$-plane $\Pi_k$ the function $F$ is a quadratic function

$$
G(y) = F(x_1 + Py) = 1/2 y^T B y - g^T y + F(x_1) ,
$$
where
\[ B = P A P', \quad g = -P F'(x_1). \]

Since \( A \) is positive definite, so also is \( B \). Inasmuch as
\[ \hat{x} = B^{-1}g \]

affords a strict minimum to \( G \), the point
\[ \hat{x} = x_1 + P\hat{p} \]

affords a strict minimum to \( F \) on \( \Pi_k \). By virtue of the relation
\[ 0 = G'(\hat{x}) = P F'(x_1 + P\hat{p}) = P F'(x) \]

the minimum point \( \hat{x} \) of \( F \) on \( \Pi_k \) is characterized by the relations
\[ P_i F'(\hat{x}) = 0 \quad (i = 1, \ldots, k) \]

and hence, by the condition that \( F'(\hat{x}) \) be orthogonal to \( \Pi_k \).

Finally, since \( \hat{x} \) is the center to ellipsoid \( G(y) = \gamma \) \((\gamma > G(\hat{x}))\) in \( y \)-space, the point \( \hat{x} \) is the center of the \((k-1)\)-dimensional ellipsoid \( E_{k-1} \) in which \( \Pi_k \) intersects the \((n-1)\)-dimensional ellipsoid \( F(x) = \gamma \). If \( i > k+1 \) the vector
\[ x_i - \hat{x} = a_{k+1} p_{k+1} + \ldots + a_{i-1} p_{i-1} \]

is conjugate to vectors \( p_1, \ldots, p_k \). Consequently, \( x_1 \) lies in the \((n-k)\)-plane \( \Pi_{n-k} \), as was to be proved.

In the next section we will modify the conjugate direction method to obtain the conjugate directions from the gradient vector at end of intermediate linear minimization steps.
4.4 Method of Conjugate Gradients (CG-Algorithm)

In the description of the conjugate direction algorithm given in Section 4.3, there is a noticeable absence of an explicit procedure for generating a conjugate system of vectors $p_1$, $p_2$, $p_3$, ... In this section, we will describe a method for generating mutually conjugate vectors which is conceptually appealing as a minimization process as well as effective from a computational point of view. As an initial description of the conjugate gradient algorithm, we modify the initial description of the CD-algorithm, given in Section 4.3, as follows.

After selecting an initial point $x_1$, we compute the steepest descent vector $p_1 = -F'(x_1)$ of $F$ at $x_1$ and obtain the minimum point $x_2$ of $F$ on the line $L_1$ through $x_1$ in the direction $p_1$. The $(n-1)$-plane $\Pi_{n-1}$ through $x_2$ conjugate to $p_1$ contains the minimum point $x_0$ of $F$, so that our space of search can be diminished by one. We repeat the process restricting ourselves to the $(n-1)$-plane $\Pi_{n-1}$. We select a steepest descent vector $p_2$ of $F$ at $x_2$ in $\Pi_{n-1}$ and obtain the minimum point $x_3$ of $F$ on the line $L_2$ through $x_2$ in the direction $p_2$. The $(n-2)$-plane $\Pi_{n-2}$ in $\Pi_{n-1}$ through $x_3$ and conjugate to $p_2$ contains $x_0$, so that at the next step we limit our search to $\Pi_{n-2}$, a space of one lower dimension. This process is continued, decreasing the dimension of our space of search by one in each step. In the $k^{th}$ step we select a steepest descent vector $p_k$ at $x_k$ in an $(n-k+1)$-plane $\Pi_{n-k+1}$ and obtain
the minimum point \( x_{k+1} \) of \( F \) on the line \( L_k \) in \( \mathbb{R}^{n-k+1} \) through \( x_k \) in the direction \( p_k \). Our next space of search containing \( x_0 \) is the \((n-k)\)-plane \( \mathbb{R}^{n-k} \) in \( \mathbb{R}^{n-k+1} \) through \( x_{k+1} \) and conjugate to \( p_k \). After \( m \leq n \) steps we obtain a point \( x_{m+1} \) which coincides with the minimum point \( x_o \) of \( F \).

The description of the conjugate gradient method just given is somewhat involved. In applications we need not determine the planes \( \mathbb{R}^{n-1} \), \( \mathbb{R}^{n-2} \), ... explicitly. All we need is the formula

\[
P_{k+1} = -F'(x_{k+1}) + b_k p_k, \quad b_k = \frac{|F'(x_{k+1})|^2}{|F'(x_k)|^2}, \quad (4.4.1)
\]

for the direction \( p_{k+1} \) of steepest descent of \( F \) at \( x_{k+1} \) in the \((n-k)\)-plane \( \mathbb{R}^{n-k} \) through \( x_{k+1} \) conjugate to the vector \( p_1, \ldots, p_k \) previously chosen. This formula will be justified presently. Accepting formula 4.4.1 for \( p_{k+1} \) we restate the CG-algorithm in a form that is easily extended to the case when \( F \) is nonquadratic. In this form we initially select a point \( x_1 \) and the vector \( p_1 = -F'(x_1) \). Then for \( k = 1, 2, 3, \ldots \) we determine \( x_{k+1} \) and \( p_{k+1} \) from \( x_k \) and \( p_k \) by the rules:

1. Find the minimum point \( x_{k+1} = x_k + a_k p_k \) of \( F \) on the line \( x = x_k + a p_k \).

2. Determine the next direction \( p_{k+1} \) by the formula

\[
p_{k+1} = \xi_{k+1} + b_k p_k, \quad \xi_{k+1} = -F'(x_{k+1}), \quad b_k = \frac{|\xi_{k+1}|^2}{|\xi_k|^2}. \quad (4.4.2)
\]
We terminate at the \( m \)-th step if \( r_{m+1} = -F'(x_{m+1}) = 0 \).

At the minimum point \( x_{k+1} = x_k + a_k p_k \) of \( F \) on the line \( x = x_k + \alpha p_k \), the residual \( r_{k+1} = -F(x_{k+1}) \) is orthogonal to \( p_k \), that is,

\[
p_k r_{k+1} = 0 \quad (k = 1, \ldots, m). \quad (4.4.3)
\]

Recalling that

\[
F_{k+1} = h - Ax_{k+1} = h - A(x_k + a_k p_k) = r_k - a_k A p_k
\]

the relations

\[
0 = P_k r_{k+1} = P_k (r_k - a_k A p_k) = P_k r_k - a_k P_k A p_k
\]

yield the formula

\[
a_k = \frac{c_k}{d_k}, \quad c_k = P_k r_k, \quad d_k = P_k A p_k \quad (k=1,\ldots,m) \quad (4.4.4)
\]

for the scalar \( a_k \) determining \( x_{k+1} = x_k + a_k p_k \). The scalar \( c_k \) can be computed by the alternative formula

\[
c_k = |r_k|^2 \quad (k = 1, \ldots, m). \quad (4.4.5)
\]

This is clearly true for \( k = 1 \) since \( P_1 = 1 \). For \( k > 1 \) we have

\[
P_k r_k = 0, \quad \text{by 4.4.3, so that}
\]

\[
c_k = P_k r_k = (r_k + b_k-1 P_{k-1} r_{k-1}) = |r_k|^2.
\]

Recall that in the initial description of the CG-algorithm, the vector \( P_{k+1} \) was required to be conjugate to \( P_1, \ldots, P_k \). In particular, \( P_{k+1} \) must be conjugate to \( P_k \), so that

\[
P_k A p_{k+1} = 0 \quad (k = 1, \ldots, m). \quad (4.4.6)
\]
By the computations

\[ P_k A^k (r_{k+1} + b_k P_k) = P_k A_{k+1} + b_k d_k \]

we obtain a first formula

\[ b_k = \frac{-P_k A r_{k+1}}{d_k} \quad (k = 1, \ldots, m-1) \quad (4.4.7) \]

for \( b_k \). By use of the relation

\[ r_{k+1} = (r_k - a_k \epsilon_k) r_{k+1} + c_k b_k \]

we find a second formula

\[ b_k = \frac{\|r_{k+1}\|^2 - r_k r_{k+1}}{c_k} \quad (k = 1, \ldots, m-1) \quad (4.4.8) \]

for \( b_k \). But, as we shall see in a moment,

\[ r_{k-1} r_{k+1} = (r_k - a_k \epsilon_k) r_{k-1} - a_k d_k = 0 \quad (k = 1, \ldots, m-1) \quad (4.4.9) \]

so that \( b_k \) is given by

\[ b_k = \frac{\|r_{k+1}\|^2}{c_k} \quad (4.4.10) \]

as stated in 4.4.2. As noted in (Hestenes and Stiefel, 1952), formula 4.4.8 can be used in place of 4.4.10 as a correction for roundoff errors which arise due to inexact computations. Since \( p_1 = r_1 \) and

\[ P_{k+1} A_p_{k+1} = P_{k+1} A (r_{k+1} + b_k P_k) = P_{k+1} A r_{k+1} \]

we have
\[ d_k = \beta_k A r_k \quad (k = 1, \ldots, m) \quad (4.4.11) \]

as an alternative formula for \( d_k \).

We will now give a formulation of the conjugate gradient algorithm (CG-algorithm) which is readily adaptable to programming.

**Algorithm 4.4.1** Initially select a point \( x_1 \) and compute

\[ P_1 = F_1 = - F'(x_1) = b - Ax_1 \]

1. \( c_k = P_k r_k \) or \( c_k = |r_k|^2 \)
2. \( d_k = P_k A r_k \)
3. \( a_k = \frac{c_k}{d_k} \)
4. \( x_{k+1} = x_k + a_k P_k \)
5. \( r_{k+1} = r_k - a_k A P_k \)
6. \( b_k = \frac{P_k A r_{k+1}}{d_k} \) or \( b_k = \frac{|r_{k+1}|^2}{c_k} \)
7. \( P_{k+1} = P_k + b_k P_k \)

Terminate at the \( m \)th step if \( r_{m+1} = 0 \). Then \( m < n \) and \( x_{m+1} = x_0 \), the minimum point of \( F \).

If \( r_1 = 0 \), then \( x_1 \) minimizes \( F \) and the algorithm terminates at the initial step. Suppose that \( r_1 \neq 0 \). As will be seen presently, the residuals \( r_1, r_2, \ldots \), generated by CG-algorithm 4.4.1, are mutually orthogonal. There is accordingly a first integer \( m \leq n \).
such that \( r_{m+1} = 0 \). The algorithm terminates in \( m \leq n \) steps. For \( k \leq m \) we have \( c_k = p_k^T r_k = |r_k|^2 > 0 \) so that \( p_k \neq 0 \) and
\[
d_k = p_k^T A p_k > 0
\]
since \( A \) is positive definite. It follows that the scalars \( a_1, \ldots, a_n \) are positive and that the points \( x_1, \ldots, x_{m+1} \) are distinct. By writing step 7 in CG-algorithm 4.4.1, with
\[
b_k = c_{k+1} / c_k,
\]
in the form
\[
p_{k+1} = c_{k+1} \left( \frac{p_k}{c_k} + \frac{r_{k+1}}{c_{k+1}} \right)
\]
we see that
\[
p_1 = r_1
\]
\[
p_2 = c_2 \left[ \frac{p_1}{c_1} + \frac{r_2}{c_2} \right] = c_2 \left[ \frac{r_1}{c_1} + \frac{r_2}{c_2} \right]
\]
\[
p_3 = c_3 \left[ \frac{p_2}{c_2} + \frac{r_3}{c_3} \right] = c_3 \left[ \frac{r_1}{c_1} + \frac{r_2}{c_2} + \frac{r_3}{c_3} \right]
\]
and in general, that
\[
p_k = c_k \left[ \frac{r_1}{c_1} + \frac{r_2}{c_2} + \ldots + \frac{r_k}{c_k} \right] \quad (k = 1, 2, \ldots, m), \quad (4.4.12)
\]
where \( c_j = |r_j|^2 \) \((j = 1, 2, \ldots, k)\). Thus, \( p_k \) is a linear combination to the residuals \( r_1, \ldots, r_k \). Conversely, for \( k > 1 \), the residual \( r_k \) is in the 2-space generated by \( p_k \) and \( p_{k+1} \), as can be seen from the relations
\[
r_1 = p_1 \quad r_k = p_k - b_{k-1} p_{k-1} \quad (k=2,\ldots,m). \quad (4.4.13)
\]

Basic properties of the CG-algorithm are given in the following theorem.
Theorem 4.4.1

The direction vectors \( p_1, p_2, \ldots \), generated by CG-algorithm 4.4.1 are mutually conjugate and the points \( x_1, x_2, \ldots \) are distinct. CG-algorithm 4.4.1 is a conjugate direction method in which the residuals \( r_1, r_2, \ldots \) are mutually orthogonal. Accordingly, we have the orthogonality relations
\[
    r_j^\top r_k = 0 \quad (j \neq k) \quad (4.4.14)
\]
in addition to the CD-relations
\[
    p_j^\top A p_k = 0 \quad (j \neq k) \quad (4.4.15)
\]
\[
    p_k^\top r_j = p_k^\top r_k \quad (j \leq k) \quad (4.4.16)
\]
\[
    r_j^\top r_k = 0 \quad (j < k) \quad (4.4.17)
\]

The negative gradient \( r_{k+1} \) of \( F \) at \( x_{k+1} \) is orthogonal to the direction vectors \( p_1, \ldots, p_k \), signifying that \( x_{k+1} \) minimizes \( F \) on the \( k \)-plane
\[
    \Pi_k: \quad x = x_1 + a_1 p_1 + \ldots + a_k p_k.
\]

Proof: The last conclusion in the theorem follows from Theorem 4.3.2 once we have shown that Algorithm 4.4.1 is a CD-algorithm. To show that Algorithm 4.4.1 is a CD-algorithm observe that, because the scalars \( a_1, a_2, \ldots \), appearing in Algorithm 4.4.1 are positive, it follows from Theorem 4.3.3 that relations 4.4.15 and 4.4.16 are a consequence of relations 4.4.17. Theorem 4.4.1, therefore, can be established by showing that the relations
hold. These will be established by induction. We have seen already that
\[ r_j r_k = 0, \quad p_j r_k = 0 \quad (j < k) \]
(4.4.18)

by virtue of 4.4.9 and 4.4.3. Hence, 4.4.18 holds when \( k = 2 \) and for \( k > 2 \) we can limit ourselves to the case \( j < k-1 \). Suppose that relations 4.4.18 hold when \( k \leq i \). If \( r_{i+1} = 0 \), the algorithm terminates and the theorem is established. Suppose that \( r_{i+1} \neq 0 \).

Then \( p_{i+1} \neq 0 \) and the algorithm continues. Our induction will be complete when we have shown that
\[ r_j r_{i+1} = 0, \quad p_j r_{i+1} = 0 \quad (j < i) \]
(4.4.19)

To this end we use the fact that, by Theorem 4.3.3, relations 4.4.17 with \( j < k \leq i \) imply the conditions 4.4.15 and the conditions
\[ p_j a_{i-1} = 0 \quad (j < i) \]
(4.4.20)

As a consequence we have
\[ r_j a_{i-1} = 0 \quad (j < i) \]
(4.4.21)
as can be seen by the computations \( r_j a_{i-1} = r_{i-1} a_{i-1} = 0 \) and
\[ r_j a_{i-1} = (p_j - b_{j-1} p_{j-1}) a_{i-1} = 0 \quad (1 < j < i) \]

Combining 4.4.20 and 4.4.21 with the relations \( p_j r_i = r_j r_i = 0 \) for \( j < i \), we obtain the desired relations
\[ E_{j+1} = E_{j} - a_{j}A_{j} \quad (j < i) \]
\[ P_{j+1} = P_{j} - a_{j}A_{j} \quad (j < i) \]
given in 4.4.19. This proves Theorem 4.4.1.

Since \( x_{k+1} \) minimizes \( F \) on the line \( x = x_{k} + \alpha p_{k} \), we have
\[ F(x_{k+1}) > F(x_{k+1}) \quad (k = 1, \ldots, m) \]  
so that \( F \) is diminished in each step of CG-algorithm 5.4.1. The distance from the minimum point \( x_{o} \) of \( F \) is also diminished in each step, that is
\[ |x_{o} - x_{k}| > |x_{o} - x_{k+1}| \quad (k = 1, \ldots, m) \]  
Because \( a_{k} > 0 \) and
\[ |x_{o} - x_{k}|^{2} = |x_{o} - x_{k+1} + a_{k}p_{k}|^{2} \]
\[ = |x_{o} - x_{k+1}|^{2} + 2a_{k}p_{k}(x_{o} - x_{k+1}) + |a_{k}p_{k}|^{2}, \]
equality 4.4.23 will hold if we show that
\[ p_{k}(x_{o} - x_{k+1}) > 0 \quad (k = 1, \ldots, m). \]  
since \( x_{o} = x_{n+1} \), this inequality holds when \( k = m \). If \( k < m \), we have
\[ x_{o} = x_{m+1} = x_{k+1} + a_{k+1}p_{k+1} + \ldots + a_{m}p_{m}. \]
Consequently,
\[ p_{k}(x_{o} - x_{k+1}) = a_{k+1}p_{k+1} + \ldots + a_{m}p_{m} > 0 \]
because, for \( j > k \), we have
Theorem 4.4.2  The vectors $P_1$, ..., $P_m$ generated by CG-algorithm 4.4.1 satisfy the equations

$$P_2 = (1 + b_1)P_1 - a_1 A P_1 \quad (4.4.25)$$

$$P_{k+1} = (1 + b_k)P_k - a_k A P_k - b_{k-1} P_{k-1} \quad (k > 1) \quad (4.4.26).$$

Similarly, the residuals $r_1$, ..., $r_m$ are connected by the relations

$$r_2 = r_1 - a_1 A r_1 \quad (4.4.27)$$

$$r_{k+1} = (1 + b_{k-1})r_k - a_k A r_k - b_{k-1} r_{k-1} \quad (b > 1), \quad (4.4.28)$$

where

$$b_{k-1} = \frac{a_k b_{k-1}}{a_k - 1}. \quad \text{Proof:} \quad \text{Relation 4.4.26 is obtained by eliminating } r_k \text{ and } r_{k+1} \text{ from the equations}$$

$$P_{k+1} = r_{k+1} + b_k P_k ,$$

$$P_k = r_k - a_k A r_k ,$$

$$P_k = r_k + b_{k-1} P_{k-1} .$$
Relation 4.4.25 follows similarly, since \( p_1 = r_1 \). In the same manner Equation 4.4.28 is obtained by eliminating \( A_{p_k} \) and \( A_{p_{k-1}} \) the relations

\[
\begin{align*}
\tau_{k+1} &= \tau_k - a_k A_{p_k}, \\
A_{p_k} &= A_{\tau_k} + b_{k-1} A_{p_{k-1}}, \\
\tau_k &= \tau_{k-1} - a_{k-1} A_{p_{k-1}}.
\end{align*}
\]

Equation 4.4.27 holds because \( p_1 = \tau_1 \).

In the initial description of the CG-algorithm it was stated that the vector \( p_{k+1} \) \((1 \leq k \leq m-1)\) is in the direction of steepest descent at \( x_{k+1} \) on the conjugate \((n-k)\)-plane

\[
\hat{\Pi}_{n-k}: \quad p_j A(x - x_{k+1}) = 0 \quad (j = 1, ..., k).
\]

We show now that this is indeed true, and so justify the terminology "conjugate gradient." Observe that the direction of steepest descent for \( F \) at \( x_{k+1} \) on \( \hat{\Pi}_{n-k} \) is given by the orthogonal projection \( p \) of the negative gradient \( \tau_{k+1} = -F'(x_{k+1}) \) of \( F \) onto \( \hat{\Pi}_{n-k} \).

Since the vectors \( A_{p_1}, ..., A_{p_k} \) generate the normals to \( \hat{\Pi}_{n-k} \), this orthogonal projection is of the form

\[
\begin{align*}
p &= \tau_{k+1} + \beta_1 A_{p_1} + \cdots + \beta_k A_{p_k}.
\end{align*}
\]

In view of Theorem 4.4.2, we can express \( p \) as a linear combination

\[
\begin{align*}
p &= \tau_{k+1} + \gamma_1 p_1 + \cdots + \gamma_k p_{k+1}
\end{align*}
\]

of \( \tau_{k+1}, p_1, ..., p_{k+1} \). Since \( p \) is orthogonal to the normals \( A_{p_1}, ..., A_{p_k} \) or \( \hat{\Pi}_{n-k} \) we have
0 = \sum_{j=1}^{k} \gamma_j d_j \quad (j = 1, \ldots, k).

Recall that \( \sum_{j=1}^{k} r_j = \sum_{j=1}^{k} A r_{j+1} + \gamma_j d_j \), so that

\[
\frac{\partial}{\partial \theta} F(\mathbf{x}_{k+1}) = \mathbf{r}_{k+1}^T \sum_{j=1}^{k} \gamma_j A = \mathbf{r}_{k+1} \sum_{j=1}^{k} \gamma_j A = \mathbf{r}_{k+1} \gamma_{k+1} A.
\]

Consequently, \( \gamma_j = 0 \quad (j < k) \). On the other hand, \( \gamma_k = b_k \) by step 6 in CG-algorithm 4.4.1 for \( b_k \). Hence,

\[
\mathbf{r} = \mathbf{r}_{k+1} + b_k \mathbf{r}_k + \gamma_{k+1} \mathbf{r}_{k+1} = (1 + \gamma_{k+1}) \mathbf{r}_{k+1} = \sigma \mathbf{r}_{k+1}
\]

since the directional derivative

\[
F'(\mathbf{x}_{k+1}, \mathbf{r}) = \nabla F(\mathbf{x}_{k+1}) = - \sigma \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} = - \sigma |\gamma_{k+1}|^2
\]

of \( F \) at \( \mathbf{x}_{k+1} \) in the direction \( \mathbf{r} \) is negative it follows that \( \sigma > 0 \). Hence, \( \mathbf{r}_{k+1} \) is in the direction of steepest descent of \( F \) on \( \mathcal{P}_{n-k} \) at \( \mathbf{x}_{k+1} \).

The essential properties of the vectors \( \mathbf{r}_1, \mathbf{r}_2, \ldots \) in a CG-algorithm are their directions and not their lengths. In applications it is often convenient to introduce a scale factor \( \rho_k \) for \( \mathbf{r}_k \). For example, we may wish to scale \( \mathbf{r}_k \) so that it is a unit vector or so that it has some other convenient property. When a positive scale factor \( \rho_k \) for \( \mathbf{r}_k \) is introduced in CG-algorithm 4.4.1, we obtain a scaled CG-algorithm defined by the following relations:

**Algorithm 4.4.2** Initially \( \mathbf{r}_1 = - F(\mathbf{x}_1) \) and \( \mathbf{p}_1 = \rho_1 \mathbf{r}_1 \) where \( \rho_1 > 0 \) and \( \mathbf{x}_1 \) is arbitrary.
1. \( c_k = \frac{p_k r_k}{d_k} \) or \( c_k = \rho_k |x_k|^2 \),

2. \( d_k = p_k A p_k \),

3. \( a_k = c_k/d_k \),

4. \( x_{k+1} = x_k + a_k p_k \),

5. \( r_{k+1} = r_k - a_k A p_k \)

6. \( b_k = -\frac{p_k A r_{k+1}}{d_k} \) or \( b_k = \frac{|r_{k+1}|^2}{c_k} \)

7. \( p_{k+1} = \rho_{k+1} (r_{k+1} + b_k p_k) \), \( \rho_{k+1} > 0 \).

Terminate at the mth step if \( r_{m+1} = 0 \). Then, \( m \leq n \) and \( x_{m+1} = x_0 \), the minimum point of \( F \).
5. MODIFIED CONJUGATE GRADIENT METHOD

A form of the conjugate gradient method was given in Section 4 of Chapter 4. In the present chapter we shall investigate the conjugate gradient method in depth, giving several alternative versions of the method. In the first section we shall present a modification of the conjugate gradient method by changing the scalar factor of $p_k$, and show that the number of steps required to obtain the minimum point of a quadratic function $F$ is bounded by the number of distinct eigenvalues of the Hessian $A$ of $F$.

In the second section, we will introduce a modification of the conjugate gradient method which can obtain a critical point of a quadratic function $F$ whose Hessian $A$ is nonnegative definite, and give the planar conjugate gradient method as defined by Hestenes (1980). The planar conjugate gradient method can obtain critical points of successively on mutually conjugate 2-planes, whereas, in the standard conjugate gradient method, we restrict ourselves to successively locating critical points of $F$ on mutually conjugate lines. Consideration of the planar conjugate gradient method leads to a better understanding of modifications to the standard conjugate gradient method that are given in the third section of this chapter.

In the last section, we will introduce a generalized conjugate gradient method which involves use of a generalized gradient $HF$ of $F$. This modification of the standard conjugate gradient method can be obtained by changing the direction as well as the length of the conjugate gradient vectors $p_k$. The generalized conjugate gradient method is useful
when the Hessian $A$ of $F$ is sparse and nonnegative definite. Finally, we will develop a form of the generalized conjugate gradient method which is used in Chapters 6 and 7 to solve analysis of variance problems.

## 5.1 Conjugate Gradient Algorithms

In Section 4, Chapter 4, we introduced a conjugate gradient method which minimizes a positive definite quadratic function

$$F(x) = \frac{1}{2} x' A x - h' x + \text{const}$$

in $m \leq n$ steps, where $n$ is the rank of $A$. This conjugate gradient method, modified so as to admit a positive scale factor $\rho_k$ for the conjugate gradient $p_k$, is given by the following algorithm.

**Algorithm 5.1.1** Initially $x_1$ is arbitrary, $r_1 = - F'(x_1) = h - Ax_1$ and $p_1 = \rho_1 r_1$. For each $k = 1, 2, \ldots$, compute

1. $c_k = p_k r_k$ or $c_k = \rho_k |r_k|^2$,
2. $d_k = p_k A p_k$ or $d_k = \rho_k r_k A p_k$,
3. $a_k = c_k / d_k$,
4. $x_{k+1} = x_k + a_k p_k$,
5. $r_{k+1} = r_k - a_k A p_k$,
6. $b_k = \frac{r_{k+1} A r_{k+1}}{d_k}$ or $b_k = \frac{|r_{k+1}|^2}{c_k}$,
7. $p_{k+1} = \rho_{k+1} (r_{k+1} + b_k p_k)$.

This algorithm terminates at the $m$th step if $r_{m+1} = 0$ in which case $x_{m+1}$ is the minimum point of $F$. If no roundoff errors occur when using a computer, the algorithm terminates in $m \leq n$ steps. When roundoff
errors occur, \( x_{n+1} \) is normally an acceptable estimate of the minimum point \( x_0 \) of \( F \). If not, the algorithm should be repeated with
\( x_{n+1} \) as the new \( x_1 \).

The scalars \( a_k \) and \( b_k \) in Algorithm 5.1.1 are determined by the constraints

\[
P_k r_{k+1} = p_k (r_k - a_k A r_k) = 0, \\
P_k A r_{k+1} = p_k (r_k - a_k A r_k + b_k p_k) = 0, \\
(5.1.1)
\]

so that, as in Section 4, Chapter 4,

\[
a_k = c_k / d_k, \quad c_k = p_k r_k, \quad d_k = p_k A r_k, \quad b_k = -\frac{p_k A r_{k+1}}{d_k}.
\]

In view of steps 6 and 7 in Algorithm 5.1.1 we have, as before,

\[
c_{k+1} = p_{k+1} r_{k+1} = p_{k+1} (r_k + b_k p_k) \quad r_{k+1} = p_{k+1} |r_{k+1}|^2, \\
d_{k+1} = p_{k+1} A r_{k+1} = p_{k+1} (r_k + b_k p_k) A r_{k+1} \\
= p_{k+1} (r_k + b_k p_k) A r_{k+1}.
\]

This justifies the formulas \( c_k = \rho_k |r_k|^2 \) and \( d_k = \rho_k r_k A r_k \).

Since

\[
\rho_k r_k r_{k+1} = \rho_k r_k (r_k - a_k A r_k) = c_k - a_k d_k = 0
\]

we have \( r_k r_{k+1} = 0 \). Hence,

\[
|r_{k+1}|^2 = r_{k+1} (r_k - a_k A r_k) = a_k p_k A r_{k+1} = a_k d_k b_k = c_k b_k.
\]
so that \( b_k = \frac{\|r_{k+1}\|^2}{c_k} \). This justifies the various formulas for \( a_k, b_k, c_k, \) and \( d_k \) given in Algorithm 5.1.1.

The scaling of \( p_k \) in Algorithm 5.1.1 alters its length but not its direction. The quantities

\[
\hat{p}_k = \frac{p_k}{\rho_k}, \quad \hat{a}_k = a_k \rho_k, \quad \hat{b}_k = b_k \rho_k = \frac{\|r_{k+1}\|^2}{\|r_k\|^2},
\]

\[
\hat{c}_k = \frac{c_k}{\rho_k} = \frac{\|r_k\|^2}, \quad \hat{d}_k = \frac{d_k}{\rho_k^2} = \frac{p_k A p_k}{\rho_k^2}
\]

are the values of \( p_k, a_k, b_k, c_k, d_k \) for the choice \( \rho_k = 1 \) and do not depend upon the scale factor \( \rho_k \) of \( p_k \). For a given initial point \( x_1 \), the point \( x_1, x_2, x_3, \ldots \) are generated by Algorithm 5.1.1 and their residuals \( r_1, r_2, r_3, \ldots \) are the same for all choices of the scale factors \( \rho_1, \rho_2, \rho_3, \ldots \). The relations

\[
r_{j} r_k = 0 \quad (j \neq k), \quad p_{j} A p_k = 0 \quad (j \neq k), \quad (5.1.2)
\]

\[
p_k r_j = p_k r_k = c_k \quad (j \leq k), \quad p_j r_k = 0 \quad (j < k) \quad (5.1.3)
\]

hold when \( \rho_k = 1 \), hence, they hold for all admissible choice of \( \rho_k \).

Equation 5.1.2 state that the residuals \( r_1, \ldots, r_m \) are mutually orthogonal and that the direction vectors \( p_1, \ldots, p_m \) are mutually conjugate. For \( k \leq m \), the vector \( p_k \) is the linear combination

\[
p_k = c_k \left[ \frac{r_1}{\gamma_1} + \ldots + \frac{r_k}{\gamma_k} \right], \quad \gamma_j = \|r_j\|^2, \quad (5.1.4)
\]

of the residuals \( r_1, \ldots, r_k \). This result holds when \( k = 1 \). Since \( c_{k+1} = \rho_{k+1} \gamma_{k+1} \) and \( b_k = \gamma_{k+1}/c_k \), \( p_{k+1} \) can be put in the form
Hence, if 5.1.4 holds for \( k < m \), we have

\[
 p_{k+1} = c_{k+1} \left[ \frac{r_k}{c_k} + \frac{r_{k+1}}{\gamma_{k+1}} \right].
\]

as was to be proved. Formula 5.1.4 also follows from the corresponding formula 4.4.12 for the case \( \rho_k = 1 \) (\( k = 1, 2, \ldots, m \)).

Ortega and Rheinboldt (1970) show that the following characterization of a CG-algorithm that terminates in exactly \( n \) steps.

**Theorem 5.1.1**  Let \( p_1, \ldots, p_m \) be mutually conjugate vectors and let \( x_1 \) be an initial point such that the conjugate direction method

\[
x_{k+1} = x_k + a_k p_k, \quad r_{k+1} = r_k - a_k A p_k, \quad r_1 = -F'(x_1),
\]

\[
a_k = c_k / d_k, \quad c_k = p_k r_k, \quad d_k = p_k A p_k
\]

terminates in exactly \( n \) steps. Replace \( p_k \) by \(-p_k\) if necessary so that \( a_k \geq 0 \), and so that \( c_k \geq 0 \). If the residuals \( r_1, \ldots, r_n \) are mutually orthogonal, there exist positive number \( \rho_1, \ldots, \rho_n, b_1, \ldots, b_{n-1} \) such that

\[
p_1 = \rho_1 r_1, \quad p_{k+1} = \rho_{k+1} (r_{k+1} + b_k p_k)
\]

and the conjugate direction method is a conjugate gradient method.
Next, we will discuss the modification of Algorithm 5.1.1.

In our previous discussion it was shown that the point $x_{k+1}$ in the conjugate Algorithm 5.1.1 minimize $F$ on the $k$-plane

$$\Pi_k: \quad x = x_1 + \alpha_1 p_1 + \ldots + \alpha_k p_k.$$  

The $k$-plane $\Pi_k$ contains the points $x_1, \ldots, x_{k+1}$ and is uniquely determined by these points. The direction $p_{k+1}$ in Algorithm 5.1.1 is in the direction of steepest descent of $F$ at $x_{k+1}$ relative to the $(n-k)$-plane

$$p_j A(x - x_{k+1}) = 0 \quad (j = 1, \ldots, k).$$

This $(n-k)$-plane is conjugate to $\Pi_k$ and contains the minimum point $x_0$ of $F$. Hestenes (1980) shows an alternative interpretation of $p_{k+1}$. That is as follows:

**Theorem 5.1.2** The vector $p_{k+1}$ obtained in Algorithm 5.1.1 is in the direction of steepest descent of $F$ at a point $\hat{x}_{k+1}$ on the $k$-plane $\Pi_k$ through the points $x_1, \ldots, x_{k+1}$. The point $\hat{x}_{k+1}$ is the point on $\Pi_k$ whose residual $\hat{x}_{k+1}$ is of minimum length so that

$$|F'(x)| > |F'(\hat{x}_{k+1})|$$

for every point $x \neq \hat{x}_{k+1}$ on $\Pi_k$. Specifically,

$$p_{k+1} = \sigma_{k+1} F' (\hat{x}_{k+1}) = \sigma_{k+1} \hat{p}_{k+1}, \quad \sigma_{k+1} = \frac{c_{k+1}}{\gamma_{k+1}}, \quad (5.1.8)$$

where
\[
\frac{1}{\gamma_{k+1}} = \frac{1}{\gamma_1} + \ldots + \frac{1}{\gamma_{k+1}}, \quad \gamma_j = |r_j|^2 \quad (j=1, \ldots, k+1) \quad (5.1.9)
\]

\[
\hat{x}_{k+1} = \gamma_{k+1} \left( \frac{x_1}{\gamma_1} + \ldots + \frac{x_{k+1}}{\gamma_{k+1}} \right), \quad (5.1.10)
\]

\[
\hat{r}_{k+1} = \gamma_{k+1} \left( \frac{r_1}{\gamma_1} + \ldots + \frac{r_{k+1}}{\gamma_{k+1}} \right). \quad (5.1.11)
\]

**Proof:** The point \( \hat{x}_{k+1} \) defined by 5.1.9 and 5.1.10 is a convex linear combination of the points \( x_1, \ldots, x_{k+1} \) and lies in the \( k \)-plane \( \Pi_k \) through these points. In fact, since the coefficients in 5.1.10 are positive, the point \( \hat{x}_{k+1} \) lies within the \( k \)-simplex having \( x_1, \ldots, x_{k+1} \) as its vertices. The residual of \( F \) at \( \hat{x}_{k+1} \) is

\[
\hat{r}_{k+1} = h - A\hat{x}_{k+1} = \gamma_{k+1} \left( \sum_{j=1}^{k+1} \frac{h - Ax_j}{\gamma_j} \right) = \gamma_{k+1} \left( \sum_{j=1}^{k+1} \frac{r_j}{\gamma_j} \right),
\]

as stated in 5.1.11. Comparing 5.1.11 and 5.1.4 we see that 5.1.8 holds, so that \( P_{k+1} \) is in the direction of steepest descent of \( F \) at \( \hat{x}_{k+1} \). It remains to show that \( \hat{r}_{k+1} \) is the residual of \( F \) of minimum length on \( \Pi_k \) or equivalently that \( \hat{x}_{k+1} \) minimizes the auxiliary function

\[
\hat{F}(x) = \frac{1}{2} |F'(x)| = \frac{1}{2} |h - Ax|^2 \quad (5.1.12)
\]
on $\Pi_k$. The gradient of $\hat{F}$ at $\hat{x}_{k+1}$ is

$$\hat{F}'(x) = -A\hat{x}_{k+1} = -\frac{1}{c_{k+1}} A\hat{p}_{k+1}.$$  \hspace{1cm} (5.1.13)

Since $\hat{p}_{k+1}$ is conjugate to $p_1, \ldots, p_k$ it follows from 5.2.6 that the gradient $\hat{F}'(\hat{x}_{k+1})$ is orthogonal to $p_1, \ldots, p_k$ and to $\Pi_k$. Consequently, $\hat{x}_{k+1}$ minimizes $\hat{F}$ on $\Pi_k$, as was to be proved.

Hestenes shows that the vector $\hat{x}_{k+1}$ is a convex linear combination of $x_{k+1}$ and $\hat{x}_k$, as stated in the following:

**Theorem 5.1.3**  Relative to the Algorithm 5.1.1, the points

$\hat{x}_1 = x_1$, $\hat{x}_{2}, \ldots, \hat{x}_3, \ldots$, their residuals $\hat{r}_1 = r_1$, $\hat{r}_{2}, \hat{r}_3, \ldots$, and the scalars $\sigma_1, \sigma_2, \sigma_3, \ldots$, described in Theorem 5.1.2 are generated by the following relations

$$\hat{x}_1 = x_1, \hat{x}_{k+1} = \frac{x_{k+1} + b_k \sigma_k \hat{x}_k}{1 + b_k \sigma_k},$$  \hspace{1cm} (5.1.14)

$$\hat{r}_1 = r_1, \hat{r}_{k+1} = \frac{r_{k+1} + b_k \sigma_k \hat{r}_k}{1 + b_k \sigma_k},$$  \hspace{1cm} (5.1.15)

$$\sigma_1 = \rho_1, \sigma_{k+1} = \rho_{k+1}(1 + b_k \sigma_k).$$  \hspace{1cm} (5.1.16)

The proof is given in Hestenes (1980).

Relations 5.1.14, 5.1.15, and 5.1.16 can be combined with Algorithm 5.1.1 to obtain following algorithm, which is called the augmented conjugate gradient algorithm with positive scale factors $\rho_k$ for $p_k$. 

Algorithm 5.1.2
Initially $x_1$ is arbitrary and compute $r_1 = -F'(x_1) = b - Ax_1$, $p_1 = \rho_1 r_1$, $\sigma_1 = \rho_1$ and $\hat{x}_1 = x_1$. For each $k = 1, 2, \ldots$, compute

1. $c_k = p_k r_k = \rho_k |r_k|^2$,
2. $d_k = p_k A p_k = \rho_k r_k A p_k$,
3. $a_k = c_k / d_k$,
4. $x_{k+1} = x_k + a_k p_k$,
5. $r_{k+1} = r_k - a_k A p_k$,
6. $b_k = -\frac{p_k A r_{k+1}}{d_k} = \frac{|\tilde{r}_{k+1}|^2}{c_k}$,
7. $p_{k+1} = \rho_{k+1} (r_{k+1} + b_k p_k)$,
8. $\sigma_{k+1} = \rho_{k+1} (1 + b_k \sigma_k)$,
9. $\hat{x}_{k+1} = \frac{x_{k+1} + b_k \sigma_{k+1}}{1 + b_k \sigma_k}$.

The point $\hat{x}_{k+1}$ minimizes $F$ and the point $x_{k+1}$ minimizes $|F'(x)|$ on the $k$-plane $\Pi_k$ through the points $x_1, x_2, \ldots, x_{k+1}$. The algorithm terminates when $r_{m+1} = 0$, in which case we have $x_{m+1} = \hat{x}_{m+1}$ as the minimum point of $F$. The residual $\tilde{r}_k$ of $F$ at $\hat{x}_k$ is $p_k / \sigma_k$. 
If we select $\rho_k = 1$ for all values of $k$, we obtain the standard conjugate gradient algorithm introduced in Section 4, Chapter 4. If we select

$$
\rho_1 = 1, \quad \rho_{k+1} = \frac{1}{1 + b_k}
$$

so that $\sigma_k = 1$, we obtain the normalized conjugate gradient algorithm. It consists of the following steps.

**Algorithm 5.1.3** (normalized conjugate gradient algorithm).

Initially choose a point $x_1$ and compute $r_1 = -F(x_1)$, $p_1 = r_1$ and $\hat{x}_1 = x_1$. For each $k = 1, 2, \ldots$, compute

1. $c_k = p_k^T r_k = |p_k|^2$,

2. $d_k = p_k^T A p_k$,

3. $a_k = \frac{c_k}{d_k}$,

4. $x_{k+1} = x_k + a_k p_k$,

5. $r_{k+1} = r_k + a_k A p_k$,

6. $b_k = -\frac{p_k^T r_{k+1}}{d_k} = \frac{|r_{k+1}|^2}{c_k}$,

7. $p_{k+1} = \frac{r_{k+1} + b_k p_k}{1 + b_k}$,

8. $\hat{x}_{k+1} = \frac{x_{k+1} + b_k \hat{x}_k}{1 + b_k}$. 
Terminate at the mth step if \( r_{m+1} = 0 \). Then \( m \leq n \) and \( x_{m+1} \) is the minimum point of \( F \).

Of course the points \( \hat{x}_1, \hat{x}_2, \hat{x}_3, \ldots \), need not be computed if we are concerned with the minimum of \( F \).

Algorithm 5.1.3 follows Algorithm 5.1.2 by selecting

\[
\rho_1 = 1, \quad \rho_{k+1} = \frac{1}{1 + \beta_k}.
\]

By step 8 in Algorithm 5.1.2, we have \( \zeta_k = 1 \) for all values of \( k \).

This implies, by 5.2.1, that \( p_k = -F'(\hat{x}_k) = \hat{z}_k \) and that

\[
c_k = \hat{y}_k = |\xi_k|^2 = |p_k|^2,
\]

as stated in step 1 in Algorithm 5.1.2. It is of interest to note that

\[
a_k = \frac{|p_k|^2}{p_k^T \beta_k}
\]

is the reciprocal of the Rayleigh quotient of \( A \) at \( p_k \). Recall that the point

\[
\hat{x}_{k+1} = \frac{x_{-k+1} + b_k \hat{x}_k}{1 + b_k}
\]

minimizes the length of the residual \( r = -F'(x) \) on the \( k \)-plane \( \Pi_k \) through the point \( x_1, \ldots, x_{k+1} \) and hence, also on the line segment

\[
x = \frac{x_{k+1} + \hat{x}_k}{1 + \beta} \quad (0 < \beta < \infty).
\]

Since \( p_k = \hat{z}_k \), the residual \( r = -F'(x) \) at a point \( x \) on this line segment lies on the residual line segment
The shortest vector \( r \) on the residual line segment 5.1.17 is perpendicular to \( p_k - r_{k+1} \). Its parameter \( \beta \) therefore satisfies the relation

\[
(p_k - r_{k+1})' (r_{k+1} + \beta p_k) = \beta |p_k|^2 - |r_{k+1}|^2 = 0 ,
\]

so that \( \beta = |r_{k+1}|^2 / |p_k|^2 = b_k \). It follows, as was proved earlier, that the vector

\[
p_{k+1} = r_{k+1} = \frac{r_{k+1} + b_k p_k}{1 + b_k}
\]

is the shortest residual of the form 5.1.17.

Hestenes (1980) establishes the following form of the normalized conjugate gradient algorithm, which we call the method of shortest residuals.
Algorithm 5.1.4

1. Select a point $x_1$ and compute $p_1 = -F'(x_1)$.

2. Iterate as follows: Find the minimum point $x_{k+1} = x_k + a_k p_k$ of $F$ on the line $x = x_k + a p_k$. Compute $r_{k+1} = -F'(x_{k+1})$.

   Next find the shortest vector $p_{k+1}$ of the form
   
   $$p = \frac{r_{k+1} + \beta p_k}{1 + \beta}.$$

   The vector $p_{k+1}$ is obtained by selecting $\beta = |r_{k+1}|^2/|p_k|^2$.

   The algorithm terminates when $r_{m+1} = 0$.

   The vector $p_{k+1}$ is also the shortest residual in the k-simplex whose vertices are $r_1, \ldots, r_{k+1}$.

   The method of shortest residuals is applicable to a nonquadratic function $F$ by introducing a search routine for minimizing $F$ along lines. The routine should be restarted after $N \geq n$ steps. The algorithm is terminated when $r_{m+1}$ is so small that $x_{m+1}$ is an acceptable estimate of the minimum point of $F$. The method of shortest residuals is a special case of a general algorithm developed in (Wolfe, 1975) for minimizing a convex function which may be non-differentiable.
5.2 Nonnegative Hessians

As was seen in Chapter 4, a quadratic function

\[ F(x) = \frac{1}{2} x^T A x - x^T h + \text{const.} \]

possessing a minimum point \( x_o \) has a nonnegative Hessian \( F''(x) = A \).

At a minimum point \( x_o \) of \( F \) we have \( F'(x_o) = A x_o - h = 0 \), so that \( x_o \) is a solution of the linear equation

\[ A x = h. \quad (5.2.1) \]

Conversely, if \( A \) is nonnegative, every solution \( x_o \) of Equation 5.2.1 is a minimum point of \( F \). If \( x_o \) and \( x_1 \) satisfy Equation 5.2.1 then \( z = x_1 - x_o \) has \( A z = 0 \) and is accordingly a null vector of \( A \). If follows that if \( x_o \) minimizes \( F \), then every minimum point of \( F \) differs from \( x_o \) by a null vector of \( A \).

If \( A \) is a nonsingular nonnegative matrix, then \( A \) is positive definite and \( x_o = A^{-1} h \) is the unique minimum point of \( F \). In this section we consider the case in which \( A \) is a singular nonnegative matrix. Then Equation 5.2.1 may fail to have a solution, that is, \( F \) may fail to have a minimum point. First, we state a well-known result.

**Lemma 5.2.1** The matrix equation \( A x = h \) is consistent, i.e., has a solution for \( x \), if and only if \( C^\perp(A, h) \) is the same as \( C^\perp(A) \), i.e., if and only if \( b^T h = 0 \) for every vector \( b \) such that \( b^T A = 0 \), where \( C^\perp(X) \) denotes the orthogonal null column space \( X \).
Proof: \( b' h = 0 \) for every vector \( b \) such that \( b' A = 0 \)

\[\iff \quad z' (I - AA') h = 0 \quad \text{for all vector } z\]

\[\iff \quad (I - AA') h = 0\]

\[\iff \quad AA' h = h\]

\[\iff \quad Ax = h \quad \text{has a solution for } x .\]

where \( A' \) is a generalized inverse of \( A \) such that \( AA' A = A \).

Therefore, Equation 5.2.1 has a solution if only if \( h \) is orthogonal to the null vectors of \( A \). Hence, \( F \) possesses a minimum point if and only if \( h \) is orthogonal to the null space of \( A \).

Although \( F \) may fail to have a minimum point, the associated quadratic form

\[\hat{F}(x) = \frac{1}{2} |F'(x)|^2 = \frac{1}{2} |h - Ax|^2\]

always possesses a minimum point \( \hat{x}_o \). Such a point \( \hat{x}_o \) is a solution of the equation

\[A^2 \hat{x} = A \hat{h}\]

and is called a least square solution of Equation 5.2.1. Every least square solution of Equation 5.2.1 differs from \( \hat{x}_o \) by a null vector of \( A \). There is a unique least square solution of Equation 5.2.1 orthogonal to the null space of \( A \). It is the shortest least squares solution of Equation 5.2.1.
The conjugate gradient algorithm developed in preceding sections is applicable to a quadratic function $F$ whose Hessian $A$ is non-negative. If $F$ possesses a minimum point, the algorithm terminates at the minimum point $x_{m+1}$ of $F$. If $F$ fails to have a minimum point, the conjugate gradient algorithm terminates when $d_m = 0$ in which case the vector $\hat{x}_m$ obtained by the augmented conjugate algorithm is a least square solution of $Ax = h$. Observe that in the augmented conjugate gradient Algorithm 5.2.1 with $\rho_k$ as a positive scale factor: We can observe that $p_k \neq 0$ whenever $r_k \neq 0$. The algorithm terminates at the $m$th step if either $d_m = 0$ or else if $d_m > 0$ and $r_{m+1} = 0$. If $r_m = 0$, then $x_{m+1}$ is a minimum point of $F$. Every other minimum point $F$ is of the form $x_{m+1} + z$, where $z$ is a null vector of $A$. If $d_m = p_mA_p = 0$, then, because $A$ is nonnegative, we have $A_p = 0$, so that $p_m$ is a null vector of $A$. $p_m \neq 0$ since $r_m \neq 0$. Since $p_m = \sigma_m (h - A \hat{x}_m)$ we have

$$A_p = \sigma_m (Ah - A^2 \hat{x}_m) = 0.$$  

Consequently, $\hat{x}_m$ minimizes $F(x)$ and is accordingly a least square solution of $Ax = h$.

There is a modification of conjugate gradient algorithm which enable us to obtain the critical point $x_0$ of a quadratic function

$$F(x) = \frac{1}{2} \langle A x, x \rangle + h^T x + \text{const}$$
whose Hessian \( A \) is nonsingular. The matrix \( A \) may be definite or indefinite. If \( A \) is definite, then \( x_0 \) is an extreme point of \( F \). If \( A \) is indefinite, then \( x_0 \) is a saddle point of \( F \). The critical point of \( F \) is the point \( x_0 = A^{-1}h \) at which
\[
F'(x) = Ax - h = 0.
\]
The point \( x_0 \) is the minimum point of the associated quadratic function
\[
\hat{F}(x) = \frac{1}{2}F'(x)^2 = \frac{1}{2}h - Ax^2.
\]
It is clear that the conjugate gradient algorithm can be applied to \( \hat{F} \) to obtain its minimum point \( x_0 \), the critical point of \( A \).

When the original conjugate gradient algorithm is applied directly to \( F \), the algorithm may fail if \( A \) is indefinite. In the conjugate gradient algorithm we obtain successively critical points \( x_2, x_3, \ldots \) of \( F \) on mutually conjugate lines \( x = x_k + \alpha p_k \) (\( k = 1, 2, 3, \ldots \)). If in the \( k \)th step we encounter the situation in which \( r_k = -F'(x_k) = 0 \) and \( d_k = p_k A p_k = 0 \), then \( F \) has no critical point on the line \( x = x_k + \alpha p_k \) and the algorithm terminates prematurely. However, in this situation we can continue by finding the critical point of \( F \) on the 2-plane \( x = x_k + \alpha p_k + \beta \lambda p_k \). This suggests that we can modify the conjugate algorithm to obtain a new algorithm that is effective in the indefinite as well as in the definite case. This is done by finding critical points of \( F \) successively on mutually conjugate lines and 2-planes in an appropriate
method. The new algorithm is termed a planar conjugate algorithm by Hestenes (1980) to emphasize that 2-planes play a significant role. In the following description of our algorithm we give alternative formulas for some of the scalars appearing in the routine.

Algorithm 5.2.1 (planar conjugate gradient algorithm)

Initial step. Select an initial point $x_1$, set $\epsilon = 1/2$, and compute

$$\xi_1 = - \nabla F(x_1), \quad p_1 = \xi_1, \quad q_1 = A p_1.$$ (5.2.3)

Iterative steps. Having obtained $x_k$, $\xi_k$, $p_k$, $q_k$, compute

$$A p_k + A q_k, \quad d_k = p_k A p_k, \quad \delta_k = p_k A q_k, \quad e_k = q_k A q_k$$ (5.2.4a)

$$\Delta_k = d_k e_k - \delta_k^2, \quad c_k = p_k - \xi_k.$$ (5.2.4b)

If $|\Delta_k| < \epsilon \delta_k^2$ then go to 5.2.5a, else go to 5.2.6a

$$a_k = \frac{c_k}{d_k}, \quad x_{k+1} = x_k + a_k p_k, \quad \xi_{k+1} = \xi_k - a_k \xi_k.$$ (5.2.5a)

If $\xi_{k+1} = 0$ terminate, else compute

$$p_{k+1} = \xi_{k+1} + b_k p_k, \quad b_k = - \frac{\nabla^2 F(x_{k+1})}{d_k} = \frac{|\xi_{k+1}|^2}{c_k}.$$ (5.2.5b)

$$q_{k+1} = A p_{k+1} + \beta_k p_k, \quad \beta_k = - \frac{\nabla^2 F(x_{k+1})}{d_k} = - \frac{q_k A p_{k+1}}{d_k}.$$ (5.2.5c)

Increase the index $k$ by 1 and go to 5.2.4a.
\[ c_k = \frac{c_k e_k - \delta_k d_k}{\Delta_k}, \quad d_k = \frac{d_k e_k - \delta_k c_k}{\Delta_k} \] (5.2.6a)

\[ x_{k+2} = x_k + \hat{c}_k p_k + \hat{d}_k q_k, \quad r_{k+2} = r_k + \hat{c}_k a r_k - \hat{d}_k a q_k. \] (5.2.6b)

If \( r_{k+2} = 0 \) terminate, else compute

\[ p_{k+2} = r_{k+2} + \frac{b_k}{\Delta_k} (d_k q_k - \delta_k p_k), \quad b_k = -q_k a r_{k+2} \] (5.2.6c)

\[ q_{k+2} = A p_{k+2} + \frac{\beta_k}{\Delta_k} (d_k q_k - e_k p_k), \quad \beta_k = -q_k a A p_{k+2} \] (5.2.6d)

Increase the index \( k \) by 2 and go to 5.2.4a.

Termination. At termination the last \( x \)-vector computed is the critical point \( x_o \) of \( F \). If the algorithm does not terminate early, the point \( x_{n+1} \) is the critical point of \( F \), unless significant roundoff errors. If due to roundoff errors the point \( x_{n+1} \) or \( x_{n+2} \) is an unsatisfactory estimate of \( x_o \), restart the algorithm with \( x_{n+1} \) or \( x_{n+2} \) as the new initial point \( x_1 \).

The formulas for scalars in Algorithm 5.2.3-5.2.6 have been chosen so that they are applicable when a scale factor \( \rho_k \) is introduced we have the optional formulas

\[ c_k = \frac{|r_k|^2}{d_k}, d_k = g_k \hat{r}_k \] (5.2.7a)

\[ \hat{c}_k = \delta_k \frac{e_k c_k - \delta_k d_k}{\Delta_k}, \quad \hat{d}_k = \delta_k \frac{d_k^2 - \delta_k c_k}{\Delta_k}, \quad \hat{b}_k = \frac{|\hat{r}_{k+2}|^2}{\hat{d}_k} \] (5.2.7b)
The new formula for \( \hat{\mathbf{c}}_k \) remains valid under scaling of \( \mathbf{p}_k \). It should be noted that formulas 5.2.6a for \( \hat{\mathbf{c}}_k \) and \( \hat{\mathbf{d}}_k \) can be put in the form

\[
\hat{\mathbf{c}}_k = \hat{\mathbf{p}}_k - \mathbf{r}_k, \quad \hat{\mathbf{d}}_k = \hat{\mathbf{q}}_k - \mathbf{r}_k,
\]

where

\[
\hat{\mathbf{p}}_k = \frac{\mathbf{e}_k \mathbf{p}_k - \mathbf{\delta}_k \mathbf{q}_k}{\Delta_k}, \quad \hat{\mathbf{q}}_k = \frac{\mathbf{d}_k \mathbf{q}_k - \mathbf{\delta}_k \mathbf{p}_k}{\Delta_k}.
\]

The vectors \( \hat{\mathbf{p}}_k \) and \( \hat{\mathbf{q}}_k \) have the property that

\[
\mathbf{p}_k \hat{\mathbf{a}}_k = 1, \quad \mathbf{p}_k \hat{\mathbf{d}}_k = \mathbf{q}_k \hat{\mathbf{a}}_k = 0, \quad \mathbf{q}_k \hat{\mathbf{a}}_k = 1,
\]

\[
\mathbf{p}_k = \mathbf{d}_k \hat{\mathbf{p}}_k + \mathbf{\delta}_k \hat{\mathbf{q}}_k, \quad \mathbf{q}_k = \mathbf{e}_k \hat{\mathbf{p}}_k + \mathbf{\delta}_k \hat{\mathbf{q}}_k.
\]

In terms of these new vectors, formulas 5.2.6b, 5.2.6c, and 5.2.6d can be rewritten in the form

\[
\mathbf{x}_{k+2} = \mathbf{x}_k + \mathbf{c}_k \hat{\mathbf{p}}_k + \mathbf{d}_k \hat{\mathbf{q}}_k, \quad \mathbf{r}_{k+2} = \mathbf{r}_k - \mathbf{c}_k \hat{\mathbf{a}}_k - \mathbf{d}_k \hat{\mathbf{d}}_k,
\]

\[
\mathbf{p}_{k+2} = \mathbf{r}_{k+2} + \mathbf{b}_k \hat{\mathbf{a}}_k, \quad \mathbf{q}_{k+2} = \mathbf{A} \mathbf{p}_{k+2} + \mathbf{\beta} \mathbf{q}_k.
\]

when \( \mathbf{d}_k = 0 \) we have the simplified formulas

\[
\mathbf{x}_{k+2} = \mathbf{x}_k + \mathbf{c}_k \hat{\mathbf{p}}_k, \quad \mathbf{r}_{k+2} = \mathbf{r}_k - \mathbf{c}_k \hat{\mathbf{a}}_k, \quad \mathbf{q}_k = \frac{\mathbf{p}_k}{\mathbf{\delta}_k}.
\]

Step 5.2.5 is a standard conjugate gradient step which determines \( \mathbf{x}_{k+1} \) as the critical point of \( F \) on the line \( \mathbf{x} = \mathbf{x}_k + \alpha \mathbf{p}_k \). It can always be used when \( \mathbf{d}_k \neq 0 \) although, for numerical reasons, we restrict its use to the case when \( |\mathbf{\Delta}_k| \leq \varepsilon |\mathbf{\delta}_k|^2 \). We have the relations
It follows that, when $\Delta_k \neq 0$ as well as $d_k \neq 0$. We have

$$E_{k+1} = -\frac{a_k \Delta_k}{d_k} q_k.$$  \hfill (5.2.13)

If we use the familiar scaling

$$E_{k+1} = \frac{r_k + b_k p_k}{1 + b_k}$$

for $E_{k+1}$, then, in view of 5.2.12, we have alternative formula

$$E_{k+1} = E_k - \alpha_k g_k,$$

$$\alpha_k = \frac{d_k}{\Delta_k}$$

for $E_{k+1}$. In this event the scalar $\beta_k$ appearing in 5.2.5c, satisfies the relation $d_k \beta_k = \Delta_k$. On the other hand when we use the original scaling for $E_{k+1}$ we have $d_k^2 \beta_k = a_k \Delta_k$.

Step 5.2.6 is a new conjugate gradient step which determines the critical point $x_{k+2}$ of $F$ on the 2-plane

$$\Pi_2: \quad x = x_k + \alpha p_k + \beta g_k.$$  

This step can be used whenever $\Delta_k \neq 0$ but we restrict its use to the case in which $|\Delta_k| \geq \varepsilon \delta_k^2$. Observe that if $d_k = 0$ or $\rho_k = 0$, then $\Delta_k = -\delta_k^2$, so that step 5.2.4 is applied. It should
be noted that, when \(d_k \neq 0\) and \(\Delta_k \neq 0\), step 5.2.6 is equivalent to two applications of step 5.2.5. For in this case the vectors \(p_k\) and
\[
p_{k+1} = \frac{a_k}{d_k} (\delta_k p_k - d_k q_k) = r_{k+1} + b_k p_k
\]
are mutually conjugate and form an alternative basis for \(\Pi_2\). Consequently, the critical point \(x_{k+2}\) of \(F\) on \(\Pi_2\) is given by the formula
\[
x_{k+2} = x_k + a_k p_k + a_{k+1} p_{k+1} = x_{k+1} + a_{k+1} p_{k+1},
\]
where, as in 5.2.5, \(a_k = c_k/d_k\) and \(a_{k+1} = c_{k+1}/d_{k+1}\). The point \(x_{k+1}\) is critical point of \(F\) on the line \(x = x_k + \alpha p_k\). The point \(x_{k+2}\) is the critical point of \(F\) on the line \(x = x_{k+1} + \alpha p_{k+1}\).

It follows that the planar conjugate gradient algorithm whenever the linear conjugate algorithm does not terminate prematurely. In particular, this is the case which \(A\) is definite.

The switching parameter \(\epsilon = 1/2\) is used to ensure that when \(d_k\) or \(\Delta_k\) is a divisor, it is not excessively small in magnitude. Excessively small divisors tend to make an algorithm numerically unstable. The value of the switching parameter \(\epsilon\) is at our disposal. It must lie between 0 and 1. We selected the value \(\epsilon = 1/2\). When \(A\) is positive definite and \(\epsilon\) is small, then step 5.2.4 will be used except possibly for the last step.

The planar conjugate gradient algorithm 5.2.1 can be used to minimize \(F(x)\) on an \((n-N)\)-plane
where $B$ is an $(n \times N)$-dimensional matrix of rank $N < n$. We assume that $p^T A p > 0$ for all vectors $p \neq 0$ having $B^T p = 0$. This implies that the matrix

$$\hat{A} = \begin{pmatrix} A & B \\ B & 0 \end{pmatrix}$$

is nonsingular. The Lagrangian function associated with $F$ is

$$L(x, \rho) = F(x) + \rho^T G(x),$$

where $\rho$ is a Lagrange multiplier. A point $x_0$ minimizes $F(x)$ subject to $G(x) = 0$ if and only if there is a multiplier $\rho_0$ such that $(x_0, \rho_0)$ is a critical point of $L$, that is, if $(x_0, \rho_0)$ solves the linear equations

$$Ax + B\rho = h,$$
$$B^T x = k.$$  

The matrix $\hat{A}$ of coefficients is a nonsingular indefinite symmetric matrix. The planar conjugate gradient algorithm 5.2.1 applied to $L$ yields the critical point $(x_0, \rho_0)$ of $L$ and hence, the minimum point $x_0$ of $F$ on $G(x) = 0$ with $\rho_0 = 0$ as the corresponding Lagrange multiplier.

An algorithm similar to Algorithm 5.2.1 has been devised by Luenberger (1969). That algorithm is switched from linear to planar...
minimizations whenever \( d_k \) is zero, so that small errors arise when planar minimizations are used when \( d_k \) is small, but not zero. This difficulty is avoided in Algorithm 5.2.1.

### 5.3 Generalized Conjugate Gradient Algorithms

Conjugate gradient Algorithm 5.2.1 can be modified in several ways. For example, we can modify the length of the conjugate gradient \( p_k \) without changing its direction. This modification was introduced in Section 5.1 by a positive scale factor \( \rho_k \) for \( p_k \).

In a conjugate gradient algorithm we can modify the direction of \( p_k \) as well as its length. This can be done by replacing the recursion formulas

\[
E_{k+1} = E_k + \omega \quad \quad \quad (5.2.1)
\]

by a new set of recursion formulas

\[
P_1 = \rho_1 E_1 \quad \quad P_{k+1} = \rho_{k+1} (E_{k+1} + b_k p_k)
\]

This is termed a generalized conjugate gradient algorithm and consists of the following steps. We assume that the Hessian \( A \) of \( F \) is positive definite.

**Algorithm 5.3.1** (generalized conjugate gradient algorithm)

Initial step. Select a nonnegative symmetric matrix \( H \), an initial point \( x_1 \), a positive scale factor \( \rho_1 \) and compute
Iterative steps. Perform the iteration defined by the formulas

\[ d_k = p_k s_k, \quad c_k = p_k r_k \quad \text{or} \quad c_k = \rho_k \gamma_k, \quad a_k = \frac{c_k}{d_k}, \tag{5.3.3} \]

\[ x_{k+1} = x_k + a_k p_k, \quad r_{k+1} = r_k - a_k s_k. \tag{5.3.4} \]

\[ \gamma_{k+1} = r_{k+1}^T H r_{k+1}, \quad b_k = \frac{\gamma_{k+1}}{c_k} \quad \text{or} \quad b_k = \frac{-s_k^T H r_{k+1}}{d_k}. \tag{5.3.5} \]

\[ p_{k+1} = \rho_{k+1} (H r_{k+1} + b_k p_k), \quad s_{k+1} = A p_{k+1}. \tag{5.3.6} \]

where \( \rho_{k+1} \) is a positive scale factor.

Termination. If \( H \) is nonsingular, terminate at the \( m \)th step if

\[ r_{m+1} = 0. \] Then \( m \leq n \) and \( x_{m+1} \) is the minimum point of \( F \). If \( H \) has rank \( N \), terminate at the \( m \)th step if \( r_{m+1} = 0. \) Then \( m \leq N \) and \( x_{m+1} \) minimizes \( F \) on the \( N \)-plane through \( x_1 \) orthogonal to the null space of \( H \).

The augmented generalized conjugate gradient algorithm is obtained by adjoining the computations

\[ \sigma_1 = \rho_1, \quad \sigma_{k+1} = \rho_{k+1} (1 + b_k \sigma_k). \tag{5.3.7} \]

\[ \hat{x}_1 = x_1, \quad \hat{x}_{k+1} = \frac{x_{k+1} + b_k \sigma_k \hat{x}}{1 + b_k \sigma_k}. \tag{5.3.8} \]

The point \( \hat{x}_{k+1} \) minimizes the auxiliary quadratic function
\[ F(x) = \frac{1}{2} \mathbf{r}'H\mathbf{r} = \frac{1}{2}(\mathbf{r} - \mathbf{A}x)'H(\mathbf{r} - \mathbf{A}x) \quad (5.3.9) \]

on the k-plane

\[ \Pi_k: \quad x = x_1 + \alpha_1p_1 + \ldots + \alpha_kp_k \]

through \( x_1 \).

Algorithm 5.3.1 is applicable also to the case in which the Hessian \( A \) of \( F \) is nonnegative and singular. In this case Algorithm 5.3.1 terminates at an \( m \)th step either when \( d_m > 0 \) and \( Hr_{m+1} = 0 \) or else when \( d_m = 0 \). If \( d_m = p_mA_p = 0 \), then

\[ A_p = 0. \]

By formula 5.5.3

\[ 0 = A_p = \sigma_m AH(\mathbf{r} - \mathbf{A}x_m) = -\sigma_m F'(\mathbf{x}) \]

so that \( \mathbf{x}_m \) minimizes \( \hat{F} \) and is accordingly a weighted least square solution of \( A\mathbf{x} = \mathbf{h} \). It should be noted that when \( h \) is orthogonal to the null space of \( A \), the situation \( d_m = 0 \) cannot arise. For suppose that \( d_m = p_mA_p = 0 \). Then \( p_m \) is a null vector of \( A \) so that

\[ p_m^r = p_m(h - \mathbf{A}x_m) = 0. \]

We then have

\[ 0 = r_{m-m}^r = \sigma_m r_m^r(Hr_m + b_m - l_{m-1}p_{m-1}) = \sigma_m r_m^r Hr_m \]

and hence, \( Hr_m = 0 \), which is not the case. It follows that, when \( h \) is orthogonal to the null space of \( A \), Algorithm 5.3.1 yields a
minimizes \( \mathbf{x}_{m+1} \) of \( F \) on the \( N \)-plane through \( \mathbf{x}_1 \) orthogonal to the null space of \( H \).

Algorithm 5.3.1 can also be obtained from the Algorithm 5.2.1 by means of a transformation of coordinates

\[
\mathbf{x} = \mathbf{x}_1 + U\mathbf{y},
\]

where \( U \) is a matrix such that \( H = U U' \). From this result, we conclude that we can minimize \( F \) on a prescribed \( N \)-plane

\[
\Pi_N: \quad \mathbf{x} = \mathbf{x}_1 + a_{1u_1} + \ldots + a_{Nu_N}
\]

by selecting \( H = U U' \), where \( U \) is the matrix whose column vectors are \( u_1, \ldots, u_N \). Obviously, \( \Pi_N \) is the \( N \)-plane through \( \mathbf{x}_1 \) orthogonal to the null space of \( H \).

The formula for \( p_{k+1} \) in Algorithm 5.5.1 can be put in the form

\[
p_{k+1} = \rho_{k+1}(g_{k+1} + b_k p_k), \quad g_{k+1} = Hr_{k+1}
\]

If \( H \) is positive definite, then, by setting \( M = H^{-1} \), we obtain an alternative set of equations

\[
p_{k+1} = \rho_{k+1}(g_{k+1} + b_k p_k), \quad Mg_{k+1} = r_{k+1}
\]

which determines \( p_{k+1} \). This alternative set is useful when the equation \( Mg = r \) is easily solvable. This equation is easily solvable by a conjugate gradient routine if \( M \) has a small number \( N \) of distinct eigenvalues. At most \( N \) conjugate gradient steps are
need in this case. A block diagonal matrix with identical blocks has a small number of eigenvalues. If \( M \) is a good estimate of \( A \), then \( H = M^{-1} \) is close to \( A^{-1} \). Then the eigenvalues of \( HA \) will be clustered about \( \lambda = 1 \) so that relatively few steps of Algorithm 5.3.1 with \( H = M^{-1} \) or modified by 5.3.11 will yield a good estimate of the solution \( x_0 \) of \( Ax = h \). Now, we will discuss this new modified conjugate gradient method.

To solve the linear system equations

\[
Ax = h \tag{5.3.12}
\]

under consistency, it is frequently desirable to rewrite 5.3.12 as

\[
Mx = Nx + c,
\]

where \( M \) is positive definite (not necessarily symmetric), \( M-N \) is a symmetric and nonnegative definite matrix. It is a much simpler computational task to solve the system

\[
Mz = d \tag{5.3.13}
\]

then to solve 5.3.12.
Now, consider an iterative formula of the form

$$x_{k+1} = x_{k-1} + w_k (\alpha_k z_k + x_k - x_{k-1}), \quad (5.3.14)$$

where

$$Mz_k = c - (M-N)x_k. \quad (5.3.15)$$

Many iterative methods can be derived by 5.3.14; e.g., the Chebyshev semi-iterative method and Richardson second order method (Golub and Varga, 1961). The generalized conjugate gradient method is also of this form.

From Equations 5.3.14 and 5.3.15, it follows that

$$Mz_{k+1} = Mz_{k-1} - w_{k+1} \{\alpha_k (M-N)z_k + M(z_{k-1} - z_k)\}. \quad (5.3.16)$$

For the generalized conjugate gradient method the parameters \{\alpha_k, w_{k+1}\} are computed so that

$$Mz_p = 0 \quad (5.3.17)$$

for \(p \neq q\) and \(p,q = 0, 1, 2, \ldots, n-1\). Since \(M\) is \(n \times n\) positive definite matrix, Equation 5.3.17 implies that for some \(k \leq n\)

$$z_k = 0$$

and hence,

$$x_k = x. \quad (5.3.18)$$

That is, the iteration converges in no more than \(n\) steps.
Theorem 5.3.1  If \( \alpha_k = \frac{z_k^M z_k}{z_k^M (M-N) z_k} \) and \( w_k = \left[ 1 - \alpha_k \frac{z_k^N z_k}{z_k (M-N) z_{k-1}} \right]^{-1} \)

are parameters in the Equation 5.6.5, then there exists \( z_p \), \( p = 0, 1, \ldots, n-1 \), such that \( z_p^M z_q = 0 \) for \( p \neq q \) and \( p, q = 0, 1, \ldots, k \), where \( k < n-1 \). Then, from Equation 5.3.16

\[

z_{k+1}^M z_k = z_k^M z_{k+1} - w_{k+1} \left\{ \alpha_k z_k (M-N) z_k + z_k^M (z_{k-1} - z_k) \right\}
\]

\[

= 0 - w_{k+1} \left\{ z_k^M z_k - z_k^M z_{k-1} \right\}
\]

\[

= 0 \quad (5.3.19)
\]

and

\[

z_{k-1}^M z_{k+1} = z_{k-1}^M z_k - w_{k+1} \left\{ \alpha_k z_k (M-N) z_k + z_k^M (z_{k-1} - z_k) \right\}
\]

\[

= z_{k-1}^M z_k - w_{k+1} \left( z_{k-1}^M z_k - \alpha_k z_{k-1} z_k \right)
\]

\[

= z_{k-1}^M z_{k-1} - z_{k-1}^M z_{k-1} - z_{k-1}^M z_{k-1}
\]

\[

= 0 .
\]

To simplify the expression for \( w_{k+1} \), from Equation 5.3.16

\[

M z_k = M z_k - w_k \left\{ \alpha_k (M-N) z_k + M (z_k - z_k) \right\}
\]

and premultiply \( z_k \).
\begin{align*}
\hat{z}_k^{Mz} &= -w_k^{\alpha_k}z_k^{(M-N)z_k-1} \\
&= -w_k^{\alpha_k}z_{k-1}^{(M-N)z_k-1}.
\end{align*}

That is
\begin{align*}
\hat{z}_k^{Mz} &= w_k^{\alpha_k}z_{k-1}^{-(M-N)z_k},
\end{align*}

it follows
\begin{align*}
w_{k+1} &= \left( 1 - \frac{\alpha_k}{\alpha_{k-1}} \frac{\hat{z}_k^{Mz}}{\hat{z}_{k-1}^{Mz}} \frac{1}{w_k} \right)^{-1}.
\end{align*}

From 5.3.16, for \( j < k-1 \),
\begin{align*}
\hat{z}_j^{Mz+1} &= \alpha_k w_{k+1} z_{j-1}^{(M-N)z_j}.
\end{align*}

But,
\begin{align*}
\hat{z}_j^{Mz+1} &= \hat{z}_j^{Mz-1} - w_{j+1}^{\alpha_j(M-N)z_j - M(z_{j-1} - z_j)},
\end{align*}

so that
\begin{align*}
0 &= \hat{z}_k^{Mz-1} - w_{j+1}^{\alpha_j z_k^{(M-N)z_j - M(z_{j-1} - z_j)}} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz} \\
0 &= \hat{z}_j^{Mz}.
\end{align*}

Thus \( \hat{z}_j^{Mz_{k+1}} = 0 \) for \( j < k-1 \)

Hence, by induction, we obtain 5.3.17 and 5.3.18.
The modified generalized conjugate gradient method is summarized as follows.

Algorithm 5.3.2 Let \( x_0 \) be a given vector and arbitrary define \( x_{-1} \). For \( k = 0, 1, \ldots \)

1. Solve \( Mz_k = c - (M-N)x_k \).

2. Compute

\[
\alpha_k = \frac{z_k^T Mz_k}{z_k^T (M-N)z_k},
\]

\[
w_{k+1} = \begin{cases} 1 - \frac{\alpha_k z_k^T Mz_k}{\alpha_k z_{k-1}^T Mz_{k-1}} \cdot \frac{1}{w_k} \end{cases}, \quad (k \geq 1),
\]

\( w_1 = 1 \).

3. Compute

\[
x_{k+1} = x_{k-1} + w_{k+1} (\alpha_k z_k + x_k - x_{k-1}).
\]

Note that the algorithm can be viewed as an acceleration of the underlying first order iteration \( (w_{k+1} \equiv 1), x_{k+1} = x_k + \alpha_k z_k \). As with other higher order methods, the storage requirements of the algorithm are greater than those of the underlying first order iteration being accelerated.

The algorithm presented above is given primarily for expository purposes. For actual computation, the following equivalent form can be more efficient in terms of storage (Reid, 1971).
**Algorithm 5.3.3 (modified generalized conjugate gradient algorithm)**

Let $x_0$ be a given vector and arbitrary define $p_{-1}$. For $k = 0, 1, ...$

1. Solve $Mz_k = c - (M-N)x_k$

2. Compute
   
   $p_k = \frac{z_k^TMz_k}{z_{k-1}^TMz_{k-1}}, \quad k \geq 1$

   $\rho_0 = 0$

   $p_k = z_k + p_k p_{k-1}$

3. Compute

   $a_k = \frac{z_k^TMz_k}{p_k^T(M-N)p_k}$

   $x_{k+1} = x_k + a_k p_k$

These algorithms are the same as the algorithms of Concus, Golub, and O'Leary (1976). But, their algorithm has a restriction that matrices $M$ and $N$ should be symmetric.

If we choose $M = I$ and $N = I - A$, we obtain the basic conjugate gradient Algorithm 5.1.1, for which

$$z_k = r_k = b - Ax_k$$

is simply the residual at the $k$th step. When Algorithm 5.3.3 is used for solving the linear system Equations 5.3.12, $M$ is chosen at the beginning of the algorithm such that $M^{-1}$ is closed to $A^-$. The
application of Algorithm 5.3.3 will be discussed in the next chapter to solve the analysis of variance problems.
6. COMPUTATIONAL ALGORITHMS FOR UNBALANCED STRUCTURE

USING THE CONJUGATE GRADIENT METHOD

As we have seen, the conjugate gradient algorithm is one of the methods available for minimizing a quadratic function and for solving linear equations under the consistency condition. In this chapter, we shall discuss some applications of the conjugate gradient method, particularly the modified generalized conjugate gradient algorithm. In Section 6.1 and 6.2, we shall discuss algorithms for parameter estimation and for computation of regression sum of squares and residual sum of squares, respectively. A useful monotonicity property of the regression sum of squares will be also discussed in Section 6.2. One of the important components in analysis of variance computations with missing cells is degrees of freedom of the model. With missing cells, Rule 1 given in Chapter 3 cannot be applied for this purpose. The rank of the design matrix and degrees of freedom of the model will be discussed in Section 6.3. In the model

\[ y = X\hat{\beta} + \epsilon, \quad \text{var}(\epsilon) = \sigma^2 \mathbb{I}, \]

the best linear unbiased estimates have to be obtained by solving Aitken equations. We shall use the conjugate gradient method for estimating parameters and computing sum of squares for this model in Section 6.4.

6.1 Parameter Estimation from the Normal Equations

Recall that we are solving the linear system equations

\[ X' DX\hat{\beta} = X' D\hat{\epsilon} \]

(6.1.1)

for \( \hat{\beta} \). By Lemma 5.2.1, the normal equations are consistent even though
missing cells exist in the analysis of variance model. In order to apply the modified generalized conjugate gradient method which was introduced in Chapter 5, we choose

$$M - N = XD^T X, \quad c = XD^T \tilde{y},$$

where $M$ is the inverse matrix of $M^*$ which is a generalized inverse of $X'X$ the construction of which was discussed in Section 3.1. If Algorithm 5.3.3 is restated with these choices, we obtain the following:

**Algorithm 6.1.1** An initial guess $\hat{\beta}_0$ is given. For $k = 0, 1, 2, ...$

1. Solve $M \zeta_k = X D^T \tilde{y} - X D^T X \hat{\beta}_k$ for $\zeta_k$,

2. $c_k = \frac{\zeta_{k-1} M \zeta_k}{\zeta_{k-1} M \zeta_{k-1}}$, $c_0 = 0$

3. $\delta_k = \zeta_k + c_k \delta_{k-1}$

4. $a_k = \frac{\zeta_k M \zeta_k}{\delta_k (X' D X) \delta_k}$

5. $\hat{\beta}_{k+1} = \hat{\beta}_k + a_k \delta_k$

Because we do not wish to specify the design matrix $X$ for our model, we will iterate using two corresponding sets of vectors. The first set of vectors represented by lower-case Greek letters will correspond directly to the vectors in Algorithm 6.1.1. The second set represented by uppercase Roman letters will be used in iterative formulas.
for vectors multiplied by the matrix $X$. The Greek and Roman letters will correspond so that the algorithm is easier to follow (for example, $R = X^\alpha$, $B = X^\beta$, etc.).

In the first step in Algorithm 6.1.1, it is necessary to solve

$$M^*_{\alpha_k} = X D^\alpha_{\alpha_k} - X D X^\beta_{\beta_k},$$

for $\xi_k$.

But, then $\xi_k = E[G_k]$, where

$$E = M^* X,$$

$G_k = D^\alpha_{\alpha_k} - D B_k$, and

$$B_k = X^\beta_{\beta_k}.$$

Here, $E$ may be viewed as a solution operator for a balanced analysis of variance with design matrix $X$. When applying to any $n$-vector $v$, operator $E$ will produce balanced analysis of variance estimates for that $v$. For example, when

$$X = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and $v = \begin{bmatrix} v_{11} \\ v_{12} \\ v_{21} \\ v_{22} \end{bmatrix}$

then

$$X^\alpha v = \begin{bmatrix} v_{1.} \\ v_{2.} \\ v_{11} \\ v_{12} \end{bmatrix}, \quad \text{and} \quad E(v) = M^* X^\alpha v = \begin{bmatrix} v_{1.} - v_{1} \\ v_{2.} - v_{2} \\ v_{11} - v_{1} \\ v_{12} - v_{2} \end{bmatrix}.$$
where \( v_{..} = v_{11} + v_{12} + v_{21} + v_{22}, \ v_{i.} = v_{i1} + v_{i2}, \ v_{.j} = v_{1j} + v_{2j}, \ \bar{v}_{..} = \frac{v_{..}}{4}, \ \bar{v}_{i.} = \frac{v_{i.}}{2}, \) and \( \bar{v}_{.j} = \frac{v_{.j}}{2}. \) These estimates are seen to be in the standard form which can be easily computed.

Further, since \( X_k = M^* M G_k, \)

\[
X_k^T X_k = (M^* M G_k)^T (M^* M G_k)
\]

\[= G_k^T X^T(M^* X G_k), \] since \( M^{-1} = M^{*} \)

\[= G_k^T P_k G_k, \]

\[= (P_k G_k)^T (P_k G_k). \]

Here, \( P_X = X^T(M^* X)^T \) is the projection matrix onto the column space \( X \) for the corresponding balanced analysis of variance. Thus, we can consider \( X^T(M^* X)^T \) as a projection operator. For example, for the \( X \) matrix and \( v \) vector given above we have

\[
P_{X \cdot \cdot} = X M^* X v = \begin{bmatrix}
-\bar{v}_{1.} & + & \bar{v}_{.1} - & \bar{v}_{..} \\
-\bar{v}_{1.} & + & \bar{v}_{.2} - & \bar{v}_{..} \\
-\bar{v}_{2.} & + & \bar{v}_{1.} - & \bar{v}_{..} \\
-\bar{v}_{2.} & + & \bar{v}_{.2} - & \bar{v}_{..}
\end{bmatrix}
\]

Hence, we see that \( P_{X \cdot \cdot} \) can be computed as a linear combination of the same means required to compute \( E[v] \). This form is also well-known for any balanced complete structure with design matrix \( X \) and may be easily computed. Henceforth we shall use the abbreviated notation \( [\bar{v}_{i.} + \bar{v}_{.j} - \bar{v}_{..}] \) to denote the \( P_X \) operator.
If $\hat{\beta}_0 = 0$, the formulas for $R_k = X\hat{\alpha}_k$ and $B_k = X\hat{\beta}_k$ are easy to derive:

\[
R_k = X\hat{\alpha}_k = X(\zeta_k + c_k\hat{\beta}_{k-1}) = X\xi_k + c_kX \hat{\beta}_{k-1}
\]

\[
= X\xi_k + c_kX \hat{\beta}_{k-1} = XM_k \xi_k + c_kR_{k-1}
\]

\[
= P_k\xi_k + c_kR_{k-1}
\]

\[
B_{k+1} = X\hat{\beta}_{-k+1} = X(\hat{\beta}_{-k} + a_k\hat{\alpha}_k) = X\hat{\beta}_{-k} + a_kX \hat{\alpha}_k
\]

\[
= B_k + a_kR_k.
\]

It may appear that the computation of $G_k$ requires that we store $D\tilde{y}$. This is not the case. If $\hat{\beta}_0 = 0$, then $B_0 = \phi$ and $G_0 = D\tilde{y}$. It is possible to obtain $G_{k+1}$ from $G_k$ without storing $G_0$ since

\[
G_{k+1} = D\tilde{y} - DB_{k+1} = D\tilde{y} - DB_k - a_kDR_k = G_k - a_kDR_k.
\]

The description of the computing algorithm given below can be programmed directly. It is not meant as a theoretical derivation of the method. Note that the vector $Z_k$ is used for two different
purposes within the algorithm to reduce the storage requirement. Also, the value of \( R_{-1} \) and \( \mathcal{Q}_{-1} \) are arbitrary.

**Algorithm 6.1.2**

Given \( \hat{\beta}_0 = 0 \) and \( G_0 = D\hat{\gamma} \), for \( k = 0, 1, \ldots \)

1. \( \xi_k = \mathbb{E}[G_k] \),
2. \( Z_k = P_X[G_k] \),
3. \( d_k = Z_kZ_k' \),
4. \( c_k = d_k/d_{k-1} \), \( c_0 = 0 \)
5. \( R_k = Z_k + c_kR_{k-1} \),
6. \( \mathcal{Q}_k = \xi_k + c_k\mathcal{Q}_{k-1} \),
7. \( Z_k = DR_k \),
8. \( a_k = d_k/R_kZ_k' \),
9. \( \hat{\beta}_{k+1} = \hat{\beta}_k + a_k\mathcal{Q}_k \),
10. \( G_{k+1} = G_k - a_kZ_k \).

Terminate at \( m \)th step if \( Z_{m+1} = P_X[G_{m+1}] = 0 \). Then, \( m \leq \text{rank}(X) \)

and \( \hat{\beta}_{m+1} \) solves the Equation 6.1.1.

6.2 Sum of Squares for Regression and Residual

For hypothesis testing about \( \hat{\beta} \), we are interested in computing the regression sum of squares for the model. If, as before, we denote
this regression sum of squares for the model by SSM and let $X_o$ and $X$ be the unbalanced and the balanced design matrix respectively, from 2.1.7 we have

$$SSM = \hat{\beta}'X_o'y = \hat{\gamma}'X_o'\hat{\beta} = \hat{\gamma}'DX\hat{\beta}.$$ 

Thus a natural approximation of the regression sum of squares for the model is simply

$$SSM_k = \hat{\gamma}'DX\hat{\beta}_k.$$

(6.2.1)

Because of the iterative nature of the modified generalized conjugate gradient method, we would like to derive an iterative formula for the regression sum of squares for the model.

In the following derivation, we shall rely on the formulas from computing Algorithm 6.1.2 given in the previous section.

Let

$$SSM_{k+1} = \hat{\gamma}'DX\hat{\beta}_{k+1}$$

$$= \hat{\gamma}'DB_{k+1}$$

$$= \hat{\gamma}'D(\tilde{B}_k + a_k\tilde{R}_k)$$

$$= \hat{\gamma}'DB_k + a_k\hat{\gamma}'DR_k$$

$$= SSM_k + a_k(G_k + B_k^{'\prime}R_k'\tilde{R}_k)$$

$$= SSM_k + a_kG_k\tilde{R}_k + a_kB_k^{'\prime}D\tilde{R}_k.$$

(6.2.2)
Direct substitution shows that

\[ G'_{k} R'_{k} = G'_{k} (Z'_{k} + c'_{k} R'_{k-1}) \], where \( Z'_{k} = P'_{k} G'_{k} \)

\[ = G'_{k} Z'_{k} + c'_{k} G'_{k} R'_{k-1} \]

\[ = d'_{k} + c'_{k} G'_{k} R'_{k-1} \], where \( d'_{k} = G'_{k} Z'_{k} \)

Since the matrix \( M \) is orthogonal to the vectors \( \{z'_{i}\} \),

\[ R'_{k-1} G'_{k} = \rho'_{k-1} X'_{k} G'_{k} \]

\[ = \rho'_{k-1} M'_{z'_{k}} \]

\[ = (c'_{k-1} \rho'_{k-2} + z'_{k-1}) M'_{z'_{k}} \], where \( c'_{k} = d'_{k}/d'_{k-1} \)

\[ = c'_{k-1} \rho'_{k-2} M'_{z'_{k}} \]

\[ \vdots \]

\[ = c'_{0} \rho'_{-1} M'_{z'_{k}} \]

\[ = 0 \quad \text{where} \quad c'_{0} = 0 . \quad \text{(6.2.4)} \]

And since \( a^{'k}_{k} R'_{k} = G'_{k} - G'_{k+1} \),

\[ a^{'k}_{k} B^{'k}_{k} = B^{'k}_{k} (G'_{k} - G'_{k+1}) \]

\[ = (B^{'k}_{k-1} + a^{'k-1}_{k-1} R^{'k}_{k-1}')(G'_{k} - G'_{k+1}) \]

\[ \vdots = B^{'k}_{k-1}(G'_{k} - G'_{k+1}) \]

\[ \vdots \]
Thus, the Equation 6.2.2 becomes

\[ SSM_{k+1} = SSM_k + a_k d_k. \]  \hspace{1cm} (6.2.5)

Consequently, we have the following computing algorithm for obtaining the regression sum of squares for the model:

**Algorithm 6.2.1**  
Given \( G_0 = D \) and \( SSM_0 = 0 \) for \( k = 0, 1, \ldots \)
and the value \( R_{-1} \) is arbitrary.

1. \( Z_k = P_x[G_k] \),
2. \( d_k = Z_k Z_k \),
3. \( c_k = d_k / d_{k-1} \), \( c_0 = 0 \),
4. \( R_k = Z_k + c_k R_{k-1} \),
5. \( Z_k = DR_k \),
6. \( a_k = d_k / R_k Z_k \),
7. \( SSM_{k+1} = SSM_k + a_k d_k \),
8. \( G_{k+1} = G_k - a_k Z_k \).

Terminate at \( m \)th step if \( Z_{m+1} = P_x[G_{m+1}] = 0 \). Then, \( m \leq \text{rank}(X) \) and
SSM is the regression sum of squares for the model.

If we add statements 7 and 8 to Algorithm 6.1.2, we can obtain the parameter estimates and sum of squares for the model from the same iterations.

The residual sum of squares can be derived easily from the regression sum of squares for the model. The residual sum of squares is

$$RSS = \hat{y} \hat{y} - SSM.$$ 

Next, we show that the successive values $SSM_k$ form a monotonically increasing sequence.

**Lemma 6.2.1** $\hat{y} DX_k > 0$, for the modified generalized conjugate gradient Algorithm 6.1.1, $k = 0, 1, \ldots$

**Proof:** (By induction). At the first iteration, $p_o = \zeta_0$ and $M \zeta_0 = X D \zeta_0$. Thus

$$\hat{y} DX_0 = \zeta_0 M \zeta_0$$

$$= \zeta_0 M \zeta_0$$

$$= \zeta_0 M \zeta_0$$

$$> 0.$$ 

At iteration $k$,
\[ \tilde{y}' DX_0 k = \tilde{y}' DX(c_k + c_{k-1}) \]
\[ = \tilde{y}' DXc_k + c_k \tilde{y}' DX_0 k-1 \]
\[ = \tilde{c}_0 \tilde{y}' - c_k \tilde{y}' DX_0 k-1 \]
\[ = c_k \tilde{y}' DX_0 k-1 \]
\[ > 0. \]

\( \tilde{c}_0 \tilde{y}' \) will vanish because of conjugate gradient property.

**Lemma 6.2.2** The sequence \( \{\tilde{y}' DX_0 k\} \) is monotonically increasing.

**Proof:** By above lemma,
\[ \tilde{y}' DX_0 k+1 - \tilde{y}' DX_0 k = \tilde{y}' DX(\hat{a}_k + a_k c_k) - \tilde{y}' DX_0 k \]
\[ = a_k \tilde{y}' DX_0 k \]
\[ > 0. \]

Therefore, the approximations to the regression sum of squares for the model, \( \{SSM_k\} \), is monotonically increasing. Since \( \{SSM_k\} \) is monotonically increasing, we may terminate Algorithm 6.2.1 at mth step if \( a \_m = 0 \) instead of checking vector \( Z_m = 0 \).

### 6.3 Rank of the Design Matrix when Missing Cells are Present

When missing cells are present, the rank of the design matrix, \( X_o \), cannot be obtained using Rule 1 in Chapter 3. To find the rank
of \( X_0 \), Hemmerle uses an iterative algorithm called the line search algorithm. We shall use the modified generalized conjugate gradient method to obtain the rank of the design matrix.

Let

\[
\Delta = \text{diagonal}(\delta_1, \delta_2, \ldots, \delta_n),
\]

where

\[
\delta_i = \begin{cases} 
1 & \text{if } d_i \neq 0 \\
0 & \text{if } d_i = 0 
\end{cases},
\]

and \( d_i \) is the ith diagonal element of the matrix \( D \). Then it follows that

\[
\text{rank}(X_0) = \text{rank}(X_0'X_0) = \text{rank}(X_0 DX) = \text{rank}(\sqrt{D}X)
\]

\[
= \text{rank}(\Delta X) = \text{rank}(X' \Delta X) = \text{rank}(\Delta X(X' \Delta X)^{-1}X \Delta)
\]

\[
= \text{trace}(\Delta X(X' \Delta X)^{-1}X \Delta)
\]

\[
= \sum_{i=1}^{n} e_i' \Delta X(X' \Delta X)^{-1}X \Delta e_i
\]

(6.3.1)

where \( e_i \) is a unit \( n \)-vector with one in the ith position. Now the ith component of 6.3.1.

\[
e_i' \Delta X(X' \Delta X)^{-1}X \Delta e_i
\]

(6.3.2)

is obtained as the regression sum of squares, \( S_i \), by applying the modified generalized conjugate gradient algorithm for the solution of the normal equations

\[
X' \Delta e_i X' \Delta e_i = X' \Delta e_i.
\]

(6.3.3)
Consequently, the rank of \( X_o \) is

\[
\text{rank}(X_o) = \text{trace}(\Delta X(X \Delta X)^-X \Delta)
\]

\[
= \sum_{i=1}^{n} e_i' \Delta X(X \Delta X)^-X \Delta e_i
\]

\[
= \sum_{i=1}^{n} S_i .
\]  

(6.3.4)

Notice that if the \( i \)th cell is missing then

\[
e_i' \Delta = 0 .
\]  

(6.3.5)

So, the corresponding components of the summation 6.3.4 are not computed for empty cells. In order to avoid the use of matrix storage, the components \( S_i \) must be obtained one at a time. The modified generalized conjugate gradient algorithm to compute the rank makes very efficient use of computer storage. However, for a large \( n \) with few empty cells, the method will be expensive time-wise since \( S_i \) have to be computed for each filled cell.

We now give an approach which uses storage of matrices of order \( m \), where \( m \) is the number of missing cells. Hemmerle proved this result for full rank models. We shall prove that the same result holds for the nonfull rank case. This method is very efficient for large \( n \) and small \( m \).

Let us write

\[
E = (I - M'X \Delta X) ,
\]

then
\[ M'X X E = M'X X - M'X XM'X \Delta X \]
\[ = M'X X - M'X \Delta X \]
\[ = M'X (I - \Delta)X \]
\[ = M'X \Delta X , \]

where \( \Delta = I - \Lambda \) and \( M' \) is a generalized inverse of \( XX' \) as explained in Chapter 3, and

\[ EMXX = MXX - MXAXMXX \]
\[ = MXX - MXAX \]
\[ = MXX. \]

Therefore, we have

\[ M'X XE = E M'X X = M'X \Delta X \] \( (6.3.6) \)

Further

\[ (I - E)M'X X = M'X \Delta XM'X X = M'X \Delta X = I - E \] \( (6.3.7) \)

and

\[ M'X X(I - E) = M'X XM'X \Delta X = M'X \Delta X = I - E \]

Next,

\[ M'X XE_o = M'X XM'X \Delta X \]
\[ = M'X \Delta X \]
\[ = E_o \]

and
where \( E_o = M^* X \overrightarrow{Ax} \). Thus, we also have

\[
M^* X E_o = E_o M^* X X = E_o .
\] (6.3.8)

Similarly, we have

\[
M^* X (I - E_o) = (I - E_o) M^* X X = I - E_o .
\] (6.3.9)

From Theorem 3.2.2 and Lemma 3.2.4, we obtain that

\[
\lim_{k \to \infty} (I - E) S_k (I - E) = I - E ,
\] (6.3.10)

and that

\[
\lim_{k \to \infty} (I - E) S_k
\]

is an idempotent matrix, (6.3.11)

where

\[
S_k = I + E + E^2 + \ldots + E^k .
\]

Thus, by 6.3.7 we have that

\[
\lim_{k \to \infty} M^* X X (I - E) S_k M^* X X (I - E) = M^* X X (I - E) \]

(6.3.12)

and

\[
\lim_{k \to \infty} M^* X X (I - E) S_k
\]

is also idempotent . (6.3.13)

Equation 6.3.10 becomes
\[
\lim_{k \to \infty} M^* X^* (I - E)_k = \lim_{k \to \infty} M^* X^* (I - E)^{k+1}
\]
\[
= M^* X^* - \lim_{k \to \infty} M^* X^* E^k^{k+1}.
\]
\[
= M^* X^* - \lim_{k \to \infty} E^k_0 E^k
\] by 6.3.6
\[
= M^* X^* - \lim_{k \to \infty} E^k_0 M^* X^* E^k
\] by 6.3.8
\[
= M^* X^* - \lim_{k \to \infty} E^2_0 E^{k-1}
\]
\[
= \vdots
\]
\[
= M^* X^* - \lim_{k \to \infty} E^k_0.
\]

Therefore,

\[
\text{rank}(X_o) = \text{rank}(X \Delta X) = \text{rank}(I - E)
\]
\[
= \text{rank}(\lim_{k \to \infty} (I - E)_k)
\]
\[
= \text{rank}(\lim_{k \to \infty} M^* X^* (I - E)_k)
\]
\[
= \text{trace}(M^* X^* - \lim_{k \to \infty} E^k_0^{k+1})
\]
\[
= \text{trace}(M^* X^*) - \lim_{k \to \infty} \text{trace}(E^k_0^{k+1})
\]
\[
= \text{rank}(X X) - \lim_{k \to \infty} \text{trace}(E^k_0^{k+1}).
\]

Since \( E_o = M^* X^* \Delta X \),

\[
E^k_0 = M^* X^* \Delta (\Delta X M^* X^*)^{k-1} \Delta X,
\]
and
\[
\text{trace}(E^k) = \text{trace}\{(\bar{\Delta}X^* \bar{\Delta})^k \bar{\Delta}X^* \bar{\Delta}\}
\]
\[
= \text{trace}\{(\bar{\Delta}X^* \bar{\Delta})^k\}.
\]
Let
\[
\bar{V} = \Delta X^* \bar{\Delta} = \bar{P}_X \bar{\Delta}
\]
then
\[
\bar{V} \bar{V} = \bar{\Delta}X^* \bar{\Delta} = \bar{\Delta}P_X \bar{\Delta}.
\]
Therefore, we have
\[
\text{trace}(E^k) = \text{trace}\{(\bar{V} \bar{V})^k\}
\]
Furthermore, from 6.3.14 and 6.3.15, we have that
\[
\bar{V} \bar{V} = \bar{\Delta}V.
\]
The only nonzero rows and columns of the n x n matrix \( \bar{V} \bar{V} \) given by 6.3.16 are those corresponding to the missing cells. Suppose that \( m \) cells are empty and we form the m x m symmetric matrix \( \bar{W} \) by eliminating all of the zero rows and columns \( \bar{V} \bar{V} \). Then
\[
\text{trace}(E^k) = \text{trace}(\bar{W}^k)
\]
and we may obtain the rank of \( X_0 \) by powering the m x m matrix \( \bar{W} \).
In so doing, we would compute the sequence \( \bar{W}, \bar{W}^2, \bar{W}^4, \ldots \) and obtain quadratic convergence. The sequence
\[
\text{trace}(\bar{W}), \text{trace}(\bar{W}^2), \text{trace}(\bar{W}^4), \ldots
\]
is again monotonic. It decreases monotonically to \( p - r \), where \( p \) and \( r \) are rank(\( X \)) and rank(\( X_0 \)) respectively. Notice that convergence and monotonicity of the sequence
trace(W), trace(W^2), trace(W^3), trace(W^4), ...

are direct consequences of relationship 6.3.4, relationship 6.3.17
and the monotonic convergence established for the modified generalized
conjugate gradient algorithm.

An example of this procedure is given below. Matrix storage is
required only for the m x m matrix W since the V_i vectors may be
formed one at a time, the appropriate elements being selected each
time for W. For a modest number of empty cells, both storage and
computational time are minimal. It is also satisfying in that the
approximations converge monotonically to an integer.

Example 6.3.1 We conclude with a simple example of the second
method applied to find the rank of the design matrix for a three-way
classification with interaction and missing cells among the data.
The overparameterized model is given by
\[ y_{ijkl} = \mu + a_i + b_j + a_b_{ij} + c_k + e_{ijkl} \]
and there are 2 levels for each of the three factors. There are total
of n = 8 cells involved and we assume that cells (1,1,2), (2,1,1), and
(2,1,2) are empty. With this pattern of missing cells, we have
which has rank 4.

The rank computations are illustrated in Table 6.3.1. The three missing cells correspond to rows 2, 5, and 6 of the $V$ matrix given in the table. This matrix, $V$, was formed by applying the operator $P_x$ to the unit vectors $e_2$, $e_5$, and $e_6$ respectively. The matrix $W$ consists of the 2nd, 5th, and 6th rows of the matrix $V$. We then powered $W$ by computing $W^2$, $W^4$, $W^8$, $W^{16}$, and $W^{32}$ — a total of 5 iterations. The results for $W^{32}$ are shown in Table 6.3.1 rounded to 6 decimal places. Notice the monotonic and quadratic convergence of the sequence of traces which are also given in Table 6.3.1.
Table 6.3.1. Rank computation

<table>
<thead>
<tr>
<th>V</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>.375</td>
<td>.125</td>
</tr>
<tr>
<td>.625</td>
<td>-.125</td>
</tr>
<tr>
<td>.125</td>
<td>.125</td>
</tr>
<tr>
<td>.125</td>
<td>-.125</td>
</tr>
<tr>
<td>-125</td>
<td>.625</td>
</tr>
<tr>
<td>.125</td>
<td>.375</td>
</tr>
<tr>
<td>-125</td>
<td>.125</td>
</tr>
<tr>
<td>.125</td>
<td>-.125</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>k</th>
<th>trace(W^k) = trace(E_o^k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1.87500</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1.51563</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1.23462</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>1.05456</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>1.00298</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1.00001</td>
</tr>
</tbody>
</table>
6.4 Weighted Least Squares Estimation

Consider the model
\[ y = X_0 \beta + e; \]
(6.4.1)

\( e \) is \( N \times 1 \) random vector such that \( E(e) = 0 \) and \( \text{Var}(e) = \sigma^2 H \), where \( H \) is a given \( N \times N \) diagonal matrix. We should use the Aitken equations instead of using the normal equations to obtain the best linear unbiased estimate (BLUE) of the parameters. The Aitken equations for 6.4.1 are
\[ X_0 H^{-1} X_0 \hat{\beta} = X_0 H^{-1} y. \]
(6.4.2)

Using \( T_{N \times n} \), a matrix which was introduced in Section 2.1, we can express 6.4.2 in the form
\[ X_0 H^{-1} X_0 \hat{\beta} = X_0 H^{-1} y \]
\[ \iff X^T H^{-1} X \hat{\beta} = X^T H^{-1} y \]
\[ \iff X^T H^{-1} TX \hat{\beta} = X^T y^*, \quad \hat{y}^* = H^{-1} y \]
\[ \iff X^T W X \hat{\beta} = X^T y^*_S \quad \text{(6.4.3)} \]

where \( y^*_S \) is cell sums vector of \( y^* \) and \( W = T H^{-1} T \). Since \( H^{-1} \) is a diagonal matrix, we can easily obtain \( y^* \) and \( W \) which also becomes a diagonal matrix. For example, if we have
Therefore, we have the linear system of equations

\[
X W X^*_s = X y_s \tag{6.4.4}
\]

The modified generalized conjugate gradient algorithm discussed in Chapter 5 may also be applied for the solution of 6.4.4. If we apply Algorithm 5.3.3 to solve the Equation 6.4.4 for \( \hat{\beta} \), we obtain the following:
Algorithm 6.4.1 An initial guess \( \hat{\beta}_0 \) is given. For \( k = 1, 2, \ldots \)

1. Solve \( M_{\xi_k} = X^* y_s - X W X_{\beta_k} \) for \( \xi_k \),

2. \( c_k = \frac{\xi_k^T M \xi_k}{\xi_{k-1}^T M \xi_{k-1}} \), \( c_0 = 0 \)

3. \( \hat{\varphi}_k = \xi_k + c_k \varphi_{k-1} \)

4. \( a_k = \frac{\xi_k^T M \xi_k}{\hat{\varphi}_k (X W X) \hat{\varphi}_k} \),

5. \( \hat{\beta}_{k+1} = \hat{\beta}_k + a_k \varphi_k \).

If we follow the notation of Section 6.1, we have

\( M_{\xi_k} = X^* y_s - X W X_{\beta_k} \), for \( \xi_k \).

But, then \( \xi_k = E[G_k] \), where

\( E = M^{-1} X = M^* X \)

\( G_k = y_s^* - W B_k \),

\( B_k = X_{\beta_k} \).

Here, \( E \) is the solution operator for a balanced analysis of variance with design matrix \( X \) as explained in Section 6.2. Since

\( \xi_k = M^* X G_k \),

\( \xi_k^T M \xi_k = G_k X(M^*) M^* X G_k \).
128

\[ = G^\prime_k X^\prime M^* X G_k, \text{ since } M^* = M^{-1} \]

\[ = G^\prime_k P_k C \]

\[ = (P_k X^\prime_k) (P_k X C) \]

Recall that \( P_X \) is the projection matrix onto the column space of \( X \) for the corresponding balanced analysis of variance.

If \( \hat{\theta}_o = 0 \), the formulas for \( R_k = X_o\hat{\theta}_k \) and \( B_{k+1} = X_o\hat{\theta}_{k+1} \) are easy to drive:

\[ R_k = X_o\hat{\theta}_k = X(\hat{\xi}_k + c_{k-1} \hat{\theta}_{k-1}) \]

\[ = X\hat{\xi}_k + c_k X_o\hat{\theta}_{k-1} \]

\[ = X\hat{\xi}_k + c_k X_o \hat{\theta}_{k-1} \]

\[ = P_k X C + c_k R_{k-1} \cdot \]

\[ B_{k+1} = X_o\hat{\theta}_{k+1} = X(\hat{\xi}_{k+1} + a_k \hat{\theta}_k) \]

\[ = X\hat{\xi}_{k+1} + a_k X_o \hat{\theta}_k \]

\[ = B_k + a_k R_k \cdot \]

It may appear that the computation of \( G_k \) requires that we store \( D\hat{\xi} \). This is not the case. If \( \hat{\theta}_o = 0 \), then \( B_o = 0 \) and \( G_o = Y_s^* \).

It is possible to compute \( G_{k+1} \) from \( G_k \) without storing \( G_o \):
\[ G_{k+1} = y_s^* - WB_{k+1} \]

\[ = y_s^* - WB_k - a_k WR_k \]

\[ = G_k - a_k WR_k . \]

Note that the vector \( Z_k \) is used for two different purposes within the algorithm to reduce the storage requirement. Also, the value of \( R_1 \) and \( \beta_1 \) are arbitrary. The complete algorithm may now be stated as follows:

**Algorithm 6.4.2** Given \( \hat{\beta}_0 = 0 \), \( G_0 = y_s^* \), and \( SSM_0 = 0 \). For \( k = 1, 2, \ldots \)

1. \( \xi_k = E[G_k] \),
2. \( Z_k = P_X[G_k] \),
3. \( d_k = Z_k^* Z_k \),
4. \( c_k = d_k / d_{k-1} \), \( c_0 = \xi \)
5. \( R_k = Z_k^* + c_k R_{k-1} \),
6. \( \beta_k = \xi_k + c_k \beta_{k-1} \),
7. \( Z_k = WR_k \),
8. \( a_k = d_k / R_k Z_k \),
9. \( SSM_{k+1} = SSM_k + a_k d_k \).
10. $\hat{\beta}_{k+1} = \hat{\beta}_k + a_k \hat{e}_k$

11. $G_{k+1} = G_k - a_k Z_k$

Terminate at mth step if $Z_{m+1} = P_X[G_{m+1}] = 0$. Then, $m \leq \text{rank}(X)$, $\hat{\beta}_{m+1}$ solves the Equation 6.4.3 and $SSM_m$ is the weighted regression sum of squares for the model.
The estimation of parameters is frequently of more direct interest to the experimenter than tests of hypotheses concerning these parameters. For a given model, the estimates of the model parameters would be computed using Algorithm 6.1.2. However, the experimenter may be interested in testing certain hypotheses in order to validate his assumed model, such as testing for no interaction between factors. Usually in performing an analysis of variance, the experimenter is less interested in estimating parameters than in determining which factors contribute significantly to the observed phenomenon. Attempts at analyzing designs with unequal subclass numbers are generally based on extensions of the methods for balanced data. In this chapter we shall investigate methods for computing sums of squares for testing linear hypotheses about parameters of the original overparameterized model with unbalanced data, using $M^*$ which is a generalized inverse of $XX'$ discussed in Chapter 3. In the first section we will introduce two iterative methods using $M^*$. The first of these generalizes the method of Hemmerle (1974) and the second is an extension of a method based on conjugate gradients, given by Golub and Nash (1982). Since both of these new methods use the overparameterized model, and the methods in the literature cited use the reparameterized model, iterative formulas different from those of Hemmerle and Golub and Nash must be proved. The necessary proofs are obtained by using results given in Chapter 3.
In the second section we will investigate reparameterizing the nonfull rank model using $M^*$. If the analysis of variance model does not include any missing cells, $M^{*\prime}X$ is the set of the same estimable functions as type III estimable functions of Goodnight (1976). In the last part of this section we shall discuss a method for computing these sums of squares.

In section three we shall present the conjugate gradient algorithm for obtaining the sum of squares for testing a linear hypothesis in the original overparameterized model (including the possibility of missing cells).

In the last section we shall give the basic properties of the conjugate gradient method for the analysis of covariance.

7.1 Iterative Methods

As discussed earlier, Hemmerle (1974) proposed the convergent iterative algorithm for solving the analysis of variance problems with unbalanced data including the possibility of having missing cells. Also, Golub and Nash (1982) developed another iterative procedure, based on the modified conjugate gradient algorithm, to reduce the number of iterations, over that required by Hemmerle's algorithm. Hemmerle's algorithm utilized balanced analysis of variance operators iteratively in solving the relevant normal
equations and conducting tests of hypotheses. His basic idea begins with the reparameterized model with the usual summation restrictions. The reparameterized model procedure has many advantages in hypotheses testing, but it is not suitable for computing the sum of squares for testing a linear hypothesis specified in terms of the original parameters. Our objective is to propose an iterative method based on unrestricted models. Suppose we want to solve the system of linear equations

\[ X_o'X_o\hat{\beta} = X_o'y \quad (7.1.1) \]

for \( \hat{\beta} \), where \( X_o \) is the \( N \times p \) unbalanced design matrix with 0 and 1 elements and \( y \) is \( N \)-vector of observations. As we discussed in Chapter 2, Equations 7.1.1 are equivalent to the set of equations

\[ X'D\hat{\beta} = X'D\bar{y} \quad (7.1.2) \]

where \( X \) is the \( n \times p \) balanced design matrix with one observation in each cell, \( D \) is an \( n \times n \) diagonal matrix of cell frequencies with elements \( d_1 = n_{11}, \ldots, d_n = n_{1j} \), and \( \bar{y} \) is the \( n \)-vector of the cell means. Let

\[ E_o = [I - (1/c)M^*XDX] \quad (7.1.3) \]

where \( M^* \) is positive definite and is a generalized inverse matrix of \( X'X \). The construction of one such matrix and its basic properties were discussed in Chapter 3. The constant \( c \) must be chosen such that \( c \geq \max\{d_i\} \). Then by Theorem 3.2.2
converges to a generalized inverse matrix of \((I - E_0)\) as \(k \to \infty\).

Also, by Theorem 3.2.3

\[
\lim_{k \to \infty} S_k (1/c) M^* = (7.1.5)
\]

is a generalized inverse of \(X' DX\). Therefore, \(\hat{\beta}_k = S_k (1/c) M^* X' \tilde{Y}\)
approaches a solution, \(\hat{\beta}\), to the normal equations 7.1.1.

An iterative procedure for computing \(\hat{\beta}_k\) \((k = 1, 2, \ldots)\) can be
now obtained as follows:

\[
\hat{\beta}_{k+1} = S_{k+1} (1/c) M^* X' \tilde{Y}
= (I + E_0 + \ldots + E_0^k) (1/c) M^* X' \tilde{Y}
= (1/c) M^* X' \tilde{Y} + E_0 (I + E_0 + \ldots + E_0^k) (1/c) M^* X' \tilde{Y}
= (1/c) M^* X' \tilde{Y} + E_0 \hat{\beta}_{k-1}
= (1/c) M^* X' \tilde{Y} + (I - (1/c) M^* X' DX) \hat{\beta}_k
= \hat{\beta}_k + (1/c) M^* (X' \tilde{Y} - X' DX \hat{\beta}_k)
= \hat{\beta}_k + (1/c) M^* (X' \tilde{Y} - DX \hat{\beta}_k) (7.1.6)
\]

where \(\hat{\beta}_0 = (1/c) M^* X' \tilde{Y}\). Now, recall that \(M^*\) is the balanced
analysis of variance operator which when applied to any \(n\)-vector
\(y\) will produce balanced analysis of variance estimates for \(y\), as
described in Chapter 6. If we let
\[ E = M'X', \]

then

\[ \hat{\beta}_{k+1} = \hat{\beta}_k + (1/c)E[D\hat{y} - DX\hat{\beta}_k], \]

where

\[ \hat{\beta}_0 = (1/c)E[D\hat{y}]. \] 

That is, we could compute the improved value \( \hat{\beta}_{k+1} \) by performing a balanced analysis on the weighted residual \( D(\hat{y} - X\hat{\beta}_k) \), multiplying this result, a p-vector, by \( 1/c \) and adding it to \( \hat{\beta}_k \).

This approach would seem to require the explicit calculation of \( X\hat{\beta}_k \). Although \( X \) is sparse for classification effects, we shall obtain an improved procedure as follows: If we multiply 7.1.7 on the left by \( X \), we have

\[ X\hat{\beta}_k = X\hat{\beta}_{k-1} + (1/c)XM'X' (D\hat{y} - DX\hat{\beta}_{k-1}). \]

\[ X\hat{\beta}_0 = (1/c)XM'X' D\hat{y} \] 

(7.1.8)

The expression for \( \hat{\beta}_{-k} \) may be written as

\[ \hat{\beta}_{-k} = (1/c)D\hat{y} - (I - XM'X')(1/c)D\hat{y} \]

\[ = (1/c)D\hat{y} - (I - P_X)(1/c)D\hat{y} \]

where \( P_X \) is the orthogonal projection matrix onto column space \( X \).

We may thus write the expression for \( \hat{\beta}_{-k} \) as
\[ \hat{X}^*_k = \hat{X}^*_k - 1 + (1/c)D(\hat{y} - \hat{X}^*_k - 1) - (I - P_X) \]

\[ (1/c)D(\hat{y} - \hat{X}^*_k - 1) \quad (7.1.9) \]

If we let

\[ R = I - P_X \]

and

\[ \nu^*_k = \hat{X}^*_k \]

then the iterative procedure can be expressed by the following formulas

\[ \hat{\beta}^{k+1} = \hat{\beta}^0 + \hat{\beta}^k - E[(1/c)D\nu^*_k]; \hat{\beta}^0 = E[(1/c)D\hat{y}] \]

\[ \nu^*_k = \nu^0 + [I - (1/c)D]\nu^*_{k-1} + R[(1/c)D\nu^*_{k-1}] \]

\[ \nu^0 = (1/c)D\hat{y} - R[(1/c)D\hat{y}] \quad (7.1.10) \]

where the operator \( R \) is the error residual operator for a balanced analysis of variance. For example, consider the model

\[ y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} \quad i,j = 1, 2 \]

and

\[ \nu = \begin{bmatrix} \nu_{11} \\ \nu_{12} \\ \nu_{21} \\ \nu_{22} \end{bmatrix} \]

Then we have

\[ X = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \]

and

\[ R[\nu] = \begin{bmatrix} \nu_{11} - \overline{\nu}_{1.} - \overline{\nu}_{.1} - \overline{\nu} \\ \nu_{12} - \overline{\nu}_{1.} - \overline{\nu}_{.2} - \overline{\nu} \\ \nu_{21} - \overline{\nu}_{.1} - \overline{\nu}_{1.} - \overline{\nu} \\ \nu_{22} - \overline{\nu}_{.2} - \overline{\nu}_{.2} - \overline{\nu} \end{bmatrix} \]
As $\hat{\beta}_k$ converges to $\hat{\beta}$, $\overline{\gamma} D\bar{y}_k$ becomes the sum of squares for the model.

This iterative algorithm converges more slowly than the conjugate gradient algorithm. Hence, we shall attempt to combine the conjugate gradient algorithm with the above approach. Suppose we are going to solve the linear system of equations

$$X' DX_\hat{\beta} = X' D\bar{y}$$

(7.1.11)

for $\hat{\beta}$ using the conjugate gradient method. Among several modified conjugate gradient algorithms, Algorithm 5.6.2 is the most suitable. That is as follows: An initial guess $\hat{\beta}_0$ is given. For $k = 0, 1, ...$

1. Solve $M \xi_k = X' D\bar{y} - X' D\xi_{k-1}$ for $\xi_k$

2. $c_k = \frac{\xi_k^T M \xi_k}{\xi_{k-1}^T M \xi_{k-1}}$, $c_0 = 0$

3. $\xi_k = \xi_{k-1} + c_k \xi_{k-1}$

4. $a_k = \frac{\xi_k^T M \xi_k}{\xi_{k-1}^T M \xi_{k-1}}$

5. $\hat{\beta}_{k+1} = \hat{\beta}_k + a_k \xi_k$

where $M$ is some positive definite approximation to $X' DX$ such that Equation 7.1.11 is "easy" to solve.
An analysis of the formulas in the algorithm of 7.1.12 shows that
\[ \hat{\beta}_{n+1} = \hat{\beta}_0 + P_n(K)\zeta_0, \quad (7.1.13) \]
where \( P_n(K) \) is a polynomial of degree \( n \) in the matrix \( K = M^{-1}XDX \). It can be shown (Golub and Nash (1982)) that, among all algorithms of the form 7.1.12, the conjugate gradient algorithm is optimal in that it minimizes a particular norm of the error at each iteration. The error at each iteration is defined by \( \xi_n = \hat{\beta}_n - \hat{\beta} \), where \( \hat{\beta} \) is the true solution of the system of Equations 7.1.14. The norm that the algorithm minimizes is given by the formula \( ||\xi||^2_{X^tDX} = \xi^tXDX\xi \).

This is a natural norm to use for comparing algorithms since it reflects the scalings implicit in the original problem. Since 7.1.11 can be premultiplied by any invertible matrix \( A \) without changing the solution, the 2-norm of the error is as much a function of the scaling of the variables as it is the actual error vector.

The class of algorithms defined by 7.1.13 is quite general, and almost all iterative algorithms for solving systems of linear equations are of this type. In particular, the algorithm described in 7.1.10 is a member of this class of algorithms. Thus, the optimality property for the method is of some significance.

Another significant fact is that the polynomial \( P_n(K) \) described in 7.1.13 could be chosen so that \( P_n(K) = K^{-1} \). If this choice is made, then the error at that iteration will be zero. Since the
conjugate gradient method is an optimal method, the number of iterations required for convergence is bounded by the degree of the minimal polynomial of $K$.

Returning to the formula 7.1.12 of the algorithm, if the matrix $M$ is chosen following Rule 5 in Section 3.1, then $M^{-1}$ is a generalized inverse of $XX$. If we restate the computational algorithm to solve Equation 7.1.11, and find the sum of squares for the model, then

$$SSM = \hat{\beta}'XX\hat{\beta}'$$

The algorithm is as follows:

Algorithm 7.1.1

Given $\hat{\beta}_o = 0$, $G_o = D\bar{y}$, and $SSM_o = 0$.

For $k = 0, 1, \ldots$

1. \(Z_k = E[G_k]\)

2. \(Z_k' = P[G_k]\)

3. \(d_k = Z_kZ_k'\)

4. \(c_k = d_k/d_{k-1}, \quad c_0 = 0\)

5. \(z_k = z_{k-1} + c_kc_{k-1}\)

6. \(R_k = Z_k + c_kR_{k-1}\)

7. \(Z_k = DR_k\)
8. \( a_k = d_k / R_k Z_k \)

9. \( \hat{\beta}_{k+1} = \hat{\beta}_k + a_k \hat{\sigma}_k \)

10. \( SSM_{k+1} = SSM_k + a_k d_k \)

11. \( G_{k+1} = G_k - a_k Z_k \)

Terminate at mth step if \( a_{m+1} d_{m+1} = 0 \)

This algorithm is the combination of Algorithms 6.1.2 and 6.2.1.

7.2 Reparameterizing the Nonfull Rank Model Using M

Consider the design model \( y = X_0 \beta + e \), where \( e \sim N(0, \sigma^2 I) \) and \( X_0 \) is \( N \times p \) of rank \( r (< p) \). A procedure, which changes (or transforms) the \( \hat{\beta} \) vector and the \( X_0 \) matrix to a new vector \( \hat{\theta} \) and a new matrix \( U_0 \) of size \( N \times q \) such that \( X_0 \hat{\beta} = U_0 \hat{\theta} \), is called a transformation. If \( U_0 \) is \( N \times r \) of rank \( r \), then the transformed model \( y = U_0 \hat{\theta} + e \) satisfies all conditions of the general linear model of full rank. Such a transformation may be accomplished by premultiplying the parameter \( \hat{\beta} \) by a \( r \times p \) matrix \( K \) of rank \( r \), where each element of \( K \hat{\beta} \) is a linearly independent estimable function. This transformation \( \hat{\theta} = K \hat{\beta} \) of the parameter \( \hat{\beta} \) to the parameter \( \hat{\theta} \) is called a "reparameterization" of \( \hat{\beta} \) to a basis set of estimable functions. These ideas are formally stated in the following:
Definition 7.2.1 (Graybill (1976)). Consider the design model
\[ y = X_0 \beta + e, \]
where \( e \sim N(0, \sigma^2 I) \) and \( X_0 \) is \( nxp \) of rank \( r(\leq p) \).
Let \( K \) be any \( pxq \) known matrix and let \( \theta = K \beta \). Denote the \( i \)th column of \( K \) by \( k_1 \) so that \( K = [k_1, ..., k_q] \). Then:

1. \( K \beta \) is defined to be a "transformation" of the vector \( \beta \) to the vector \( \theta \).

2. \( K \beta \) is defined to be an "estimable transformation" of the vector \( \beta \) to the vector \( \theta \) if and only if each \( k_i \beta \) is estimable for \( i = 1, 2, ..., q \).

3. \( K \beta \) is defined to be a "reparameterization" of the vector \( \beta \) to the vector \( \theta \) if and only if each \( k_i \beta \) is estimable for \( i = 1, 2, ..., q \), where \( K \) has rank \( r \) and \( r = q \); that is \( K \beta \) is a basis set of estimable function of \( \beta \).

Before we introduce a reparameterization using the matrix \( M^* \), we state some matrix results that will be needed. These are modified versions of theorems from Graybill (1976).

Theorem 7.2.1 In the design model \( y = X_0 \beta + e \), where \( e \sim N(0, \sigma^2 I) \) and \( X_0 \) is \( nxp \) of rank \( r(\leq p) \), let \( \theta = K \beta \) be a reparameterization of the vector \( \beta \) to the vector \( \theta \).

1. There exists a \( nxn \) matrix \( A \) of rank \( r \) such that \( K = A X_0 \).

2. \( K (K)^{-} = I \) for any generalized inverse of \( K \).
(3) The equation $X_0(K')^{-} = X_0$ holds for any generalized inverse of $K'$; also $X_0(K')^{-}$ has rank $r$.

(4) If $\theta = K_1$ is another reparameterization of $\theta$, then there exists a $r \times r$ nonsingular matrix $B$ such that $K_1 = BK'$.

(5) $X_0(K')^{-}$ is unique for any generalized inverse of $K'$.

Proof: The proof of (1) follows from the fact that since $K'$ has rows consisting of the vectors $k_1', k_2', \ldots, k_r'$ where each $k_i\theta$ is an estimable function, there must exist an $N \times 1$ vector $a_i$ such that $k_i' = a_i K'$. Putting these $r$ equations into a matrix equation and we get $K' = A X_0$, where $A = [a_1, a_2, \ldots, a_r]$. Since by hypothesis $K'$ has rank $r$, it follows that $\text{rank}(A) \geq r$; but since $A$ is $n \times r$, we know that $\text{rank}(A) \leq r$; so $\text{rank}(A) = r$.

To prove (2), we know that

$K'(K')^{-} = K'$

which implies

$K'(K')^{-} K'(K')^{-} = K'(K')^{-}$. \hspace{1cm} (7.2.1)

Since matrix $K'(K')^{-}$ is $r \times r$ of rank $r$, $K'(K')^{-}$ is invertible. If we premultiply or postmultiply Equation 7.2.1 by $[K'(K')^{-}]^{-1}$, we have $K'(K')^{-} = I$.

To prove (3), we know that $k' = A'X_0$ for some matrix $A$ of rank $r$, so write
which is equivalent to

\[(A X_o')(A X_o'a)(A X_o') = A X_o'.\]

This can be written as

\[(A X_o'aL'X_o')(A X_o'a) = A X_o'aL'X_o'\]

where \(X_o'aL'X_o'\) is a full rank factorization of \(X_o\). So \(A X_o'aL\) is a r\times r matrix of rank \(r\), since \(A X_o'aL'X_o' = K\), and \(K\) has rank \(r\), since \(K\) is a reparameterization of \(\beta\).

Multiply both sides of the equation \(A X_\alpha'X_\alpha(\alpha X_\alpha')^\alpha X_\alpha = A X_\alpha'\) by \(X_\alpha(\alpha X_\alpha')^{-1}\), and the result is \(X_\alpha(\alpha X_\alpha')^\alpha X_\alpha = X_\alpha\), which is \(X_\alpha(K')^{-1}X_\alpha = X_\alpha\) and \(r = \text{rank}(X_\alpha) > \text{rank }[X_\alpha(K')^{-1}] > \text{rank }[X_\alpha(K')^{-1}K'] = \text{rank }(X_\alpha) = r\).

To prove (4) we use the fact that there exist r\times N matrices \(A\) and \(A_1\) each of rank \(r\) such that \(K_1 = A_1 X_\alpha\) and \(K' = A X_\alpha\). But, we can write \(K_1 = A_1 X_\alpha'aX_\alpha\) and \(K' = A X_\alpha'aLX_\alpha\). Multiply the equations on the left by \((A_1 X_\alpha'a)\) and \((A X_\alpha'aL)\) respectively. We get

\[(A_1 X_\alpha'a)_1 K_1 = X_\alpha = (A X_\alpha'a)_1 K'\]

so

\[K_1 = (A_1 X_\alpha'a)(A X_\alpha'aL)^{-1}K' = BK\]

where \(B\) is r\times r of rank \(r\), since both \(A_1 X_\alpha'a\) and \((A X_\alpha'aL)^{-1}\) are r\times r matrices of rank \(r\). These proofs are given in Graybill (1976).

To prove (5), we will use the properties of (2) and (3). Let \((K')_1\) and \((K')_2\) be two generalized inverse matrices of \(K\). This implies that \(K(K')_1 = K\) and \(K(K')_2 = K\).
Now \( X_o (K')_1 = X_o (K')_2 K (K')_1 \)

\[
= X_o (K')_2 I \\
= X_o (K')_2
\]

which proves that \( X_o (K')^- \) is unique for any generalized inverse of \( K' \).

As a consequence of this theorem, the design model \( y = X_\beta + e \),

where \( X_o \) is \( n x p \) of rank \( r \), can be written as \( y = X_o (K')^- K' \beta + e \),

which we write as \( y = U_o \tilde{\beta} + e \), where \( \tilde{\beta} = K' \beta \) is a reparameterization

of \( \beta \). Hence \( U_o \), which is \( X_o (K')^- \), is unique for this choice of \( K \)

and is of size \( n x r \) and rank \( r \). The UMVU estimator of \( \tilde{\beta} \), using the model

\( y = U_o \tilde{\beta} + e \), is \( \tilde{\beta} = (U_o U_o')^{-1} U_o y \), which is the solution to the equations

\( U_o U_o \tilde{\beta} = U_o y \). Replace \( U_o \) with \( X(K')^- \) to get \( [X_o (K')^-] [X_o (K')^-] \tilde{\beta} = [X_o (K')^-] y \) and premultiply by \( K \) to obtain \( X_o X_o (K')^- \tilde{\beta} \), since

\( \{K[(K')^-] X_o \}' = X_o (K')^- K = X_o \) by (3) of Theorem 7.2.1.

The general solution of \( X_o X_o (K')^- \tilde{\beta} = X_o y \) for \( (K')^- \) is

\[
(K')^- \tilde{\beta} = (X_o X_o)^{-} X_o y + [I - (X_o X_o)^{-} (X_o X_o)] z
\]

for any \( p x l \) vector \( z \). Now multiply the result on the left by \( K' \)

and by (2) of Theorem 7.2.1 we get

\[
\tilde{\beta} = K' (X_o X_o)^{-} X_o y
\]

If we use model \( y = X_\tilde{\beta} + e \), the estimator of the basis set

of estimable functions \( K \tilde{\beta} \) can be obtained by multiplying the
normal equations $X'X\hat{\beta} = X'y$ on the left by $K'(X'X)^{-1}$. Since

$$K'(X'X)^{-1}X'X = K',$$

by the estimability condition,

$$K'(X'X)^{-1}(X'X)\hat{\hat{\beta}} = K'\hat{\hat{\beta}} = K'(X'X)^{-1}X'y.$$

Thus $\hat{\hat{\beta}}$, using the reparameterized model $y = U_0\hat{\beta} + e$, is identical to $K'\hat{\hat{\beta}}$ when the model $y = X_0\beta + e$ is used, that is, $K'\hat{\hat{\beta}} = \hat{\hat{\beta}}$. Clearly $\hat{\hat{\sigma}}^2$ is also the same for each model. We have proved the following theorem.

**Theorem 7.2.2** Let $y = X_0\beta + e$ be the design model with $e \sim N(0, \sigma^2 I)$ and $X_0$ is $Nxp$ of rank $r(\leq p)$, and let $y = U_0\hat{\beta} + e$ be any reparameterization of the design model. The UMVUE estimator of any estimable function $\hat{\beta}$ can be obtained from the normal equations of either model. The UMVUE estimator of $\sigma^2$ is obtainable from either model.

Although an infinite number of sets of estimable functions exists for a given set of data, one such set of estimable functions will be introduced here. This set is defined by the rows of $M'X'$ where $X'$ is the $nxp$ balanced design matrix with 0 and 1 elements, and $M'X\beta$ forms a set of estimable functions. If the data have no missing cell observations, it is estimable since

$$M'X' = M'X'D^{-1}DX = M'X'D^{-1}TX = M'X'D^{-1}TX_0$$

where $T$ is the matrix defined in Chapter 2 and $D$ is the diagonal matrix with cell frequencies as its diagonal elements. When we examine the matrix
it is found that it is the set of the same estimable functions as
the type III estimable functions of Goodnight (1976). Suppose we have
the model

\[ y_{ijk} = \mu + a_i + b_j + (ab)_{ij} + e_{ijk} \]

\[ i = 1, 2; \quad j = 1, 2; \quad k = 1, \ldots, n_{ij}. \]

Then

\[
M^* \times X = \begin{bmatrix}
\mu & a_1 & a_2 & b_1 & b_2 & ab_{11} & ab_{12} & ab_{21} & ab_{22} \\
1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{4} & 1 & \frac{1}{4} \\
0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 1 & \frac{1}{4} & \frac{1}{4} & -1 \\
0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & -1 & \frac{1}{4} & \frac{1}{4} & 1 \\
0 & 0 & 0 & 1 & -\frac{1}{2} & 1 & \frac{1}{4} & -1 & \frac{1}{4} \\
0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 1 & \frac{1}{4} & -1 & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{4} & -1 & \frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4}
\end{bmatrix}
\]

since \( M^* \times X \) is not full row rank, we select rows of \( M^* \times X \) in the
following way to obtain a full row rank matrix: Each of the rows in
\( M^* \times X \) is associated with a given model term. For example, in the
above example of \( M^* \times X \), the first row corresponds to \( \mu \), the second
and the third rows correspond to $a_1$ and $a_2$ respectively, the fourth and the fifth to $b_1$ and $b_2$ and rest of the rows to $ab_{11}$, $ab_{12}$, $ab_{21}$, $ab_{22}$, respectively. Using Rule 1 in Chapter 3, the degrees of freedom, $d$, are calculated for each model term. Then the first $d$ rows associated with each model term are retained. By following this procedure we can obtain a basis set of estimable functions. This operation can be accomplished by premultiplying $M^*X'X$ by some matrix $B$ whose dimension is $r \times p$ where $r$ is rank ($X$). For the above example

$$B = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

and the full rank set of estimable functions, $K'$, is

$$K' = BM^*X'X = \begin{bmatrix}
1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}
0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4}
\end{bmatrix}$$

Next, we reparameterize the model using $K'$, which is $BM^*X'X$. The design model $y = X\beta + e$ can be written as

$$y = X_o'(K')^{-1}K'\beta + e$$

$$= U_o\beta + e$$
where $U_o = X_o(K)'$ and $\theta = K\beta$. By Theorem 7.2.1 the matrix $U_o$ will be unique for this choice of $K$. The matrix $U_o$ is the same as the reparameterized design matrix, whose elements are $-1, 0, 1$ which would be obtained using the usual summation restrictions.

The following example will illustrate the form of the matrix $U_o$, the design matrix of the reparameterized model using usual summation restrictions.

**Example 7.2.1** Consider the model

$$y_{ijk} = \gamma + a_i + b_j + (ab)_{ij} + \epsilon_{ijk}$$

$i = 1, 2; \quad j = 1, 2, 3; \quad k = 1, \ldots, n_{ij}$ and

$n_{11} = 2, n_{12} = 1, n_{13} = 1, n_{21} = 1, n_{22} = 2, n_{23} = 1.$

Then

$$K' = BM X X = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\
\frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \\
0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} & -\frac{1}{6} \\
0 & 0 & -\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \\
\end{bmatrix}, \quad (7.2.2)
Thus $U_o$ is of the same form as the design matrix obtained by the usual summation restrictions. Using the reparameterized model we can test hypotheses of the form $K_1 \beta = 0$ where $K = [K_1, K_2]$.

Suppose that we partition $K$ as $K = [K_1, K_2]$ where $K_1 = B \hat{M} X' X$.

To test hypothesis $H_0 : K_1 \beta = 0$ against $H_a : K_1 \beta \neq 0$ under the model $y = X_0 \beta + e$ where $e \sim N(0, \sigma^2 I)$ and $X_0$ is the design matrix with rank $r$, we proceed as follows. The design model, $y = X_0 \beta + e$, can be written as

$$y = X_0 (K)' K' \beta + e$$

$$= X_0 [G_1, G_2] \begin{bmatrix} K_1' \\ K_2' \end{bmatrix} \beta + e$$

$$= X_0 G_1 K_1 \beta + X_0 G_2 K_2 \beta + e$$

$$= U_{oi} \beta_i + U_{o2} \beta_2 + e$$

where $U_{oi} = X_0 G_i$ and $\beta_1 = K_1 \beta$ for $i = 1, 2$. Therefore, the reduced model under hypotheses $K_1 \beta = 0$ is $y = U_{o2} \beta_2 + e$. A generalized likelihood ratio test statistic for $H_0 : K_1 \beta = 0$ vs. $H_a : K_1 \beta \neq 0$ can be given in the following two forms: The first form is
where \( \mathbf{\hat{\beta}} \) is any solution of the normal equations \( \mathbf{X}_o\mathbf{X}_o\mathbf{\hat{\beta}} = \mathbf{X}_o\mathbf{y} \),
\( \hat{\sigma}^2 = (\mathbf{y}'\mathbf{y} - \mathbf{\hat{\beta}}\mathbf{X}_o\mathbf{y})/(\mathbf{N}-\mathbf{r}) \), and \( \mathbf{r}_1 = \text{rank}(\mathbf{K}_1) \). The second is

\[
W = \frac{(\hat{\sigma}^2 - \hat{\sigma}^2_{\Omega})}{\hat{\sigma}^2_{\Omega}} (\mathbf{N}-\mathbf{r}) \quad (7.2.4)
\]

where \( \hat{\sigma}^2_{\Omega} = \frac{1}{\mathbf{N}} (\mathbf{y}'\mathbf{y} - \mathbf{\hat{\beta}}\mathbf{X}_o\mathbf{y}) \) is the maximum likelihood estimator of \( \sigma^2 \) in the full model \( \mathbf{y} = \mathbf{U}_o\mathbf{\theta} + \mathbf{\epsilon} \) and \( \hat{\sigma}^2_{\omega} = \frac{1}{\mathbf{N}} (\mathbf{y}'\mathbf{y} - \mathbf{\hat{\beta}}\mathbf{U}_o\mathbf{y}) \) is the maximum likelihood estimator of \( \sigma^2 \) in the reduced model \( \mathbf{y} = \mathbf{U}_o\mathbf{\theta}_2 + \mathbf{\epsilon} \) which is the full model reduced by the hypothesis \( \mathbf{H}_o \).

Henceforth, we shall consider only this second form of the statistic. The computational method for use of the first form, using iterative procedures will be discussed in the next section.

We can obtain the normal equations for the full model \( \mathbf{X}_o\mathbf{X}_o\mathbf{\hat{\beta}} = \mathbf{X}_o\mathbf{y} \).

Then

\[
\hat{\sigma}^2_{\Omega} = \frac{1}{\mathbf{N}} (\mathbf{y}'\mathbf{y} - \mathbf{\hat{\beta}}\mathbf{X}_o\mathbf{y})
= \frac{1}{\mathbf{N}} (\mathbf{y}'\mathbf{y} - \mathbf{\hat{\beta}}' \mathbf{p}_x \mathbf{y}) \quad (7.2.5)
\]

where

\[
\mathbf{p}_x = \mathbf{X}_o (\mathbf{X}_o\mathbf{X}_o)^{-1} \mathbf{X}_o'.
\]

The reduced normal equations are \( \mathbf{U}_o\mathbf{U}_o\mathbf{\hat{\theta}}_o = \mathbf{U}_o\mathbf{y} \) and
From this we get for the generalized likelihood ratio:

$$w = \frac{\hat{\gamma}' P_x \hat{\gamma} - \hat{\gamma}' P_{U02} \hat{\gamma}}{\hat{\gamma}' \hat{\gamma} - \hat{\gamma}' P_{X0} \hat{\gamma}} \left( \frac{N-r}{r} \right)$$

(7.2.7)

Thus, if the sum of squares of the reduced model, \(\hat{\gamma}' P_{U02} \hat{\gamma}\), is known, the generalized likelihood ratio statistic can be computed.

We have already mentioned in Chapter 6 and Chapter 7 that Algorithm 7.1.1 depends on the balanced design matrix \(X\) and the frequencies matrix \(D\). Next, we shall prove that Algorithm 7.1.1 for obtaining sum of squares for the model depends only on the projection matrix, \(P_X\), and \(D\).

Lemma 7.2.1 When Algorithm 7.1.1 is used for obtaining the sum of squares, the sum of squares for the model depends on the projection matrix of balanced design matrix, \(P_X\), and \(D\).

Proof: If we use Equation 7.1.3 for obtaining the model sum of squares with \(\hat{\beta}_0 = 0\), then

$$SSM_{k+1} = \hat{\gamma}' DX Q_k (M^* X DX)^* M^* X D\hat{\gamma}$$

where \(Q_k (M^* X DX)\) is a polynomial of degree \(k\) in the matrix
Since \( P_X = X^* X' \),

\[
SSM_{k+1} = \sum \left[ a \odot I + a_1 M X' DX + a_2 (M X DX)^2 + \ldots \right] + a_k (M X DX)^k M X' D\bar{y}
\]

\[
= \sum D[a_0 P_X + a_1 P_X DP_X + a_2 (P_X D)^2 P_X + \ldots] + a_k (P_X D)^k P_X D\bar{y}
\]

\[
= \sum D[R_{k+1} (P_X D)]\bar{y}
\]

where \( R_{k+1} \) is a polynomial of degree \( k+1 \) in the matrix \( P_X D \).

Thus, \( SSM_{k+1} \) depends on \( X \) only through \( P_X \) and \( D \). Therefore, if only the projection matrix corresponding to the balanced design matrix instead of the actual design matrix is known, we still obtain the sum of squares of the desired model.

When Lemma 7.2.1 is applied to obtain the reduced model sum of squares, where the reduced model is \( \bar{Y} = U \odot \bar{e} + \bar{e} \), the algorithm needs only the projection matrix of the balanced design matrix of the reparameterized model with one observation in each cell. The balanced design matrix corresponding to the reparameterized model is the \( U = X(K^{'})^{-} \). This matrix \( U \) will also be the same as the matrix obtaining by transforming the \( X \) matrix using the usual summation restrictions. Consider the model given in Example 7.2.1. Then,
Suppose we partition $U = [U_1, U_2, U_3, U_4]$ as shown above. Notice that the $U_i$'s are mutually orthogonal. Then $P_{U_i} = U_i (U_i U_i)^{-1} U_i^T$, for $i = 1, 2, 3, 4$ are unique, symmetric, idempotent matrices. By Cochran's theorem, the sum of any subset of these are also symmetric and idempotent. If these projection operators are applied on the vector $\bar{y}$ where $\bar{y} = [y_{11}, y_{12}, y_{13}, y_{21}, y_{22}, y_{23}]$, we will obtain the following:

$$
P_{U_1} \bar{y} = [\bar{y}_{..}]
$$

$$
P_{U_2} \bar{y} = [\bar{y}_{i.} - \bar{y}_{..}]
$$

$$
P_{U_3} \bar{y} = [\bar{y}_{.j} - \bar{y}_{..}]
$$

$$
P_{U_4} \bar{y} = [y_{ij} - \bar{y}_{i.} - \bar{y}_{.j} + \bar{y}_{..}]
$$

where the dot and bar notation implies averaging over the respective subscripts. $P_{U_{o2}} \bar{y}$ where $U_{o2}$ any subset of the columns of $U_0$ can be obtained by pooling the respective terms in the above set. These forms can be obtained by pooling the elements in $M' X \bar{y}$, which is also used for E and P operators discussed in Section 6.1. Now, we will look at $M' X \bar{y}$ and discuss how we can define the E and P operators which are necessary for computing regression sums of squares for the full and restricted models.
When we use Algorithm 7.1.1 to obtain $\hat{\beta}$ and the model sum of squares, all of the operations are simply vector additions or multiplications with the exception of the $E$ and $P$ operators. These operators are respectively $E = M^*X'$ and $P = XM^*X'$. Both of these operators can be implemented by $M^*_{X_f}X_f$, where $X_f$ is the nxp full row rank balanced design matrix and $M^*_f$ is a generalized inverse of $X_f'X_f$ obtained using Rule 4 in Section 3.1. The full row rank means that the model includes all possible main effects and interactions. $M^*_{X_f}X_f$ is illustrated in Table 7.2.1 for three factors with vector $y$. Thus, if we specify any model involving a subset of the main effects and/or interactions, then the elements of $E(y)$ are either identical to corresponding elements of $M^*_{X_f}X_fy$ or can be obtained by combining appropriate elements of $M^*_{X_f}X_fy$. For example, consider an experiment involving three factors $A$, $B$, and $C$. Then, $E(y)$ corresponding to the model

$$y_{ijk} = \mu + a_i + b_j + ab_{ij} + c_k + e_{ijk} \quad (7.2.8)$$

for $i = 1, \ldots, I; \quad j = 1, \ldots, J; \quad k = 1, \ldots, K,$

can be obtained by using Table 7.2.1 which displays $M^*_{X_f}X_fy$ for three factors. Thus,

$$E(y) = \begin{bmatrix}
v & \cdots & \cdots \\
v_{.j} & \cdots & \cdots \\
v_{ij} & \cdots & \cdots \\
v_{ik} & \cdots & \cdots \\
v_{ikj} & \cdots & \cdots \\
-\bar{y} & \cdots & -\bar{y}
\end{bmatrix}$$
Table 7.2.1 Factorial decomposition of the vector $y$

<table>
<thead>
<tr>
<th>Label</th>
<th>$X^*_{FZ}$</th>
<th>$M^*_{FZ}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>$y_{..}$</td>
<td>$\bar{y}_{..}$</td>
</tr>
<tr>
<td>A</td>
<td>$y_i..$</td>
<td>$\bar{y}<em>{i..} - \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>B</td>
<td>$y_{.j}$</td>
<td>$\bar{y}<em>{.j} - \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>AB</td>
<td>$y_{ij}$</td>
<td>$\bar{y}<em>{ij} - \bar{y}</em>{i..} - \bar{y}<em>{.j} + \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>C</td>
<td>$y_{..k}$</td>
<td>$\bar{y}<em>{..k} - \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>AC</td>
<td>$y_{i.k}$</td>
<td>$\bar{y}<em>{i.k} - \bar{y}</em>{i..} - \bar{y}<em>{..k} + \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>BC</td>
<td>$y_{.jk}$</td>
<td>$\bar{y}<em>{.jk} - \bar{y}</em>{.j} - \bar{y}<em>{..k} + \bar{y}</em>{..}$</td>
</tr>
<tr>
<td>ABC</td>
<td>$y_{ijk}$</td>
<td>$\bar{y}<em>{ijk} - \bar{y}</em>{ij} - \bar{y}<em>{i.k} - \bar{y}</em>{.jk} + \bar{y}<em>{i..} + \bar{y}</em>{.j} + \bar{y}<em>{..k} - \bar{y}</em>{..}$</td>
</tr>
</tbody>
</table>

where dot with $y$'s denote sums and $\bar{y}$'s denote means.

where we see that the elements of $E(y)$ are a proper subset of $M^*_{FZ}$. On the other hand if the model is given by

$$y_{ijk} = \mu + a_i + ab_{ij} + c_k + e_{ijk}$$

(7.2.8)

$$i = 1, \ldots, I; \quad j = 1, \ldots, J; \quad k = 1, \ldots, K$$

which involves a nested effect, in order to obtain the elements of $E(y)$ which correspond to the interaction term (ab), the residual terms of A and AB in Table 7.2.1 must be pooled. Thus, $E(y)$ for this model is
Next, we will consider the $P$ operator. We have to extract the factor residuals in the right-hand column of Table 7.2.1 for the model that has been specified and add them together to obtain the representation corresponding to $P(y)$. A simple example is the projection matrix of the full row rank model with three factors which is of course the identity matrix. The $P$ operator of this full row rank model is obtained by pooling all factor residuals in Table 7.2.1 which turns out to be $[y_{ijk}]$. The $P$ operator for the model Equation 7.2.8 is obtained by pooling together the factor residuals $M, A, B, AB, \text{and } C$, which gives the representation for $P(y)$ to be $[\bar{y}_{ij} + \bar{y}_{..k} - \bar{y} ...]$. For Model 7.2.8, the same terms should be pooled together which gives the identical $P$ operator as that of Model 7.2.8.

For obtaining the type III sums of squares, we first need the $P$ operator of the reduced model, $\bar{y} = U_0 \bar{\theta}_2 + \bar{e}$. The procedure for obtaining the $P$ operator of the reduced model is as follows:

First, obtain the $P$ operator of the full model using $M_f X_f^T$. Next, subtract the factor residual corresponding to the specified
hypothesis. To clarify this procedure, we will illustrate the
procedure using Model 7.2.8 where $I = 2$, $J = 3$ and $K = 2$. Then

$$\begin{bmatrix}
1 & 1/2 & 1/2 & 1/3 & 1/3 & 1/6 & 1/6 & 1/6 & 1/6 & 1/2 & 1/2 \\
0 & 1/2 & -1/2 & 0 & 0 & 1/6 & 1/6 & -1/6 & -1/6 & 0 & 0 \\
0 & 0 & 0 & 2/3 & -1/3 & -1/3 & 6/3 & 6/3 & 6/3 & 6/3 & 0 & 0 \\
0 & 0 & 0 & -1/3 & 2/3 & -1/6 & -1/6 & -1/6 & -1/6 & -1/6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/3 & 1/6 & 1/3 & 1/3 & 1/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1/6 & 1/6 & 6/6 & 6/3 & 6/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1/2 & -1/2 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$K = EM' X K = \text{'}$

The first row of $K'$ is corresponding to $u$, the second row to $A$, third
and fourth rows to $B$, fifth and sixth to $AB$, and last to $C$, respectively.

If we want to obtain the hypothesis sum of squares for testing

- $H_0: a_1 - \bar{a} + (ab)_1 - (ab)_{..} = 0$ which corresponds to
- $H_0: k_1 \beta = 0$ where $k_1$ is the row labelled $A$, we should find the appro­
priate $P$ operator for the full model, that is $P[y] = [\bar{y}_{ij} + \bar{y}_{..k} - \bar{y}_{..}]$

and obtain model sum of squares using Algorithm 7.1.1. Next,
subtract $[\bar{y}_{ij} - \bar{y}_{..}]$ which is factor residual of $A$ in Table 7.2.1
from $[\bar{y}_{ij} + \bar{y}_{..k} - \bar{y}_{..}]$, which gives $[\bar{y}_{ij} + \bar{y}_{..k} - \bar{y}_{ij..}]$. With
this $P$ operator we can obtain the reduced model sum of squares
using Algorithm 7.1.1. Finally, subtract the reduced model sum of
squares from the full model sum of squares, which gives the type III
sum of squares for $A$. 


We have used algebraic notations for pooling factor residuals. The actual computations for pooling should be accomplished by mapping one array of factor residuals at a time into the $y$ vector.

In the next section we will introduce the direct computation of hypothesis sum of squares for estimable functions $K\hat{\beta}$ using an iterative method.

### 7.3 Direct Computation of Hypothesis Sum of Squares

Assume $K$ is a matrix of full row rank $r$ with $r \leq \text{rank} (X_o)$, and $K' = T'X_o$ for some matrix $T$ (i.e., all elements of $K'\hat{\beta}$ are estimable). Then, under the normality assumptions, the hypothesis

$$H: K\hat{\beta} = 0 \quad (7.3.1)$$

can be tested by

$$F(H) = \frac{Q}{r\hat{\sigma}^2}, \quad (7.3.2)$$

where

$$Q = \hat{\beta}'K'[X_o'X_o]^{-1}K\hat{\beta},$$

$$\hat{\sigma}^2 = (y - X_o\hat{\beta})'(y - X_o\hat{\beta})/[N - \text{rank} (X_o)],$$

and $\hat{\beta}$ is a solution to the normal equations, $X_o'X_o\hat{\beta} = X_o'y$.

Calculating the numerator sum of squares for the testing is not easy. First, we need to find a generalized inverse of $X_o'X_o$ and we also need to calculate the inverse of $K(X_o'X_o)^{-1}K$. Since $K(X_o'X_o)^{-1}K$ is symmetric and $rrx$ matrix (usually $r$ is smaller than
N and p), the second inverse is not difficult. If we use the iterative method, we can save much storage and time calculating $K'(X_oX_o)^-K$ and the inverse of $K(X_oX_o)^-K$. Before introducing the algorithm, we will change $Q$ to a simpler form.

Assume $K_2$ is estimable, $K \in C(X_o)$. That is

$$K = T X_o X_o$$

which implies $(X DX)T = K$. (7.3.3)

Thus

$$Q = \gamma' X'_o (X'_o X'_o) K [K' (X'_o X'_o)^-K]^{-1} K' (X'_o X'_o)^-X'_o y$$

$$= \gamma' X'_o (X'_o X'_o)^-X'_o T [T X'_o X'_o (X'_o X'_o)^-X'_o X'_o T]^{-1} T X'_o (X'_o X'_o)^-X'_o y$$

$$= \gamma' X'_o T [T X'_o X'_o T]^{-1} T X'_o y$$

$$= \gamma' DXT [K' T]^{-1} T X D\widetilde{y}.$$ (7.3.4)

If we knew $T$ in the Equation 7.3.4, we could compute $Q$ easily.

To find the matrix $T$ using Equation 7.3.3, let the ith column vectors of $T$ and $K$ be $t_i$ and $k_i$, respectively. Then

$$X' DX t_i = k_i, \quad i = 1, \ldots, r. \quad (7.3.5)$$

These equations look like normal equations, but right-hand side of equation 7.3.3 does not involve $X$ and $D$ matrices. Thus, we cannot
apply Algorithm 6.1.2 directly for computing $t_i$. We shall develop a different computing algorithm using the modified generalized conjugate gradient method. To make notations simpler, we rewrite the Equations 7.3.5 as

$$X DX_t = k. \quad (7.3.6)$$

We start by discussing the modified generalized conjugate gradient algorithm for solving Equation 7.3.6.

**Algorithm 7.3.1** An initial value for $t_0$ is given. For $n = 0, 1, 2, ...$

1. Solve $M \zeta_n = k - X DX_t$, for $\zeta_n$
2. $c_n = \frac{\zeta_n M \zeta_n}{\zeta_{n-1} M \zeta_{n-1}}$, $c_0 = 0$
3. $\rho_n = \zeta_n + c_n \rho_{n-1}$
4. $a_n = \frac{\zeta_n M \zeta_n}{\rho_n X DX \rho_n}$
5. $t_{n+1} = t_n + a_n \rho_n$

The matrix $M$ is the same matrix that we discussed in Section 7.1. If we examine the above algorithm carefully, first we have to solve
\[ M_{n} = k - X^t D_{n}, \quad \text{for } t_{n} \]

that is

\[ t_{n} = M^{-1}_{n} k - M^{-1}_{n} X^t D_{n} \]

\[ = M_{n}[G_{n}] \]

where \( M_{n}[G_{n}] = M^{-1}_{n} G_{n}, G_{n} = k - X^t D_{n}, \) and \( B_{n} = X^t_{-n} \) and \( M_{n}^{\#} \) is an operator. To illustrate this operator consider the model

\[ \gamma_{ij} = \alpha_{i} + \beta_{j} + \epsilon_{ij}, \quad i = 1, 2; \quad \text{and } j = 1, 2; \quad \text{and} \]

\[ \gamma = [v_1, v_2, v_3, v_4, v_5]. \]

Then

\[ M_{n}^{\#}[v] = M_{n}^{\#} \begin{bmatrix} v_{1/4} \\ v_{2/2} - v_{1/4} \\ v_{3/2} - v_{1/4} \\ v_{4/2} - v_{1/4} \\ v_{5/2} - v_{1/4} \end{bmatrix} \]

This \( M_{n}^{\#} \) is an operator similar to the \( E \) operator. Since

\[ t_{n} = M_{n}^{-1} G_{n}, \quad t_{n} M_{n}^{-1}_{n} = G_{n} (M_{n}^{-1})^t \quad M_{n}^{-1} G_{n} = G_{n} M_{n}^{-1}_{n}. \]

If \( t_{0} = 0 \), the formulas \( R_{n} = X_{n} \) and \( B_{n+1} = X_{n+1} \) are easily to derive:

\[ R_{n} = X_{n} = X(t_{n} + c_{n-1}) \]

\[ = X_{n} + c_{n-1} \]

\[ = X_{n} + c_{n-1} \]

(7.3.7)
and

\[ B_{n+1} = X_{n+1} = X(t_n + a_{n+n}) \]
\[ = B_n + a_{n+n}. \]

It may appear that computation of \( G_n \) requires that we store \( k \). This is not the case. If \( t_0 = 0 \), then \( B_0 = 0 \) and \( G_k = k \). It is possible to compute \( G_{n+1} \) from \( G_n \) without storing \( G_0 \):

\[ G_{n+1} = k - X^\prime D n+1 \]
\[ = k - X^\prime D(B_n + a_{n+n}) \]
\[ = k - X^\prime D B_n - a_{n+n} X^\prime D R_n \]
\[ = G_n - a_{n+n} X^\prime D R_n. \] (7.3.8)

There are matrix multiplications involving \( X \) and \( X^\prime \) in Equation 7.3.7 and Equation 7.3.8, respectively. Without storing the matrix \( X \), we can calculate \( X^\prime v \) and \( X^\prime w \) with the model equations. If we have the model \( y_{ijk} = \mu + \alpha_i + \beta_j + \epsilon_{ijk} \), \( i = 1, 2 \) and \( j = 1, 2, 3 \) and

\[ v = \begin{bmatrix} v_{11} \\ v_{12} \\ v_{13} \\ v_{21} \\ v_{22} \\ v_{23} \end{bmatrix} \quad \text{and} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} \]
Then

\[ X_v = \begin{bmatrix} v_{1.} \\ v_{2.} \\ v_{1} \\ v_{2} \\ v_{3} \end{bmatrix} \]

where

\[ v_{1.} = v_{11} + v_{12} \]
\[ v_{j} = v_{1j} + v_{2j} \quad j = 1, 2, 3 \]  \hspace{1cm} (7.3.9)

and

\[ X_w = \begin{bmatrix} w_{1} + w_{2} + w_{4} \\ w_{1} + w_{2} + w_{5} \\ w_{1} + w_{3} + w_{4} \\ w_{1} + w_{3} + w_{5} \\ w_{1} + w_{3} + w_{6} \end{bmatrix} \]

Therefore, if we know the model we can compute \( X_v \) and \( X_w \) without storing \( X \).

The description of the computing algorithm given here can be directly programmed for solving for the \( i \)th column vector of matrix \( T \).

Note that the vector \( Z_n \) is used for two different purposes within the algorithm to reduce the storage requirement. Also, the value \( R_{-1} \) and \( D_{-1} \) are arbitrary.
Algorithm 7.3.2 Given $t_0 = 0$ and $G_0 = k_i$. For $n = 0,1,2,\ldots$

1. $Z_n = M_n^g[G_n]$

2. $d_n = G_n Z_n$

3. $c_n = d_n / d_{n-1}$, \hspace{1cm} $c_0 = 0$

4. $R_n = XZ_n + c_n R_{n-1}$

5. $Z_n = Z_n + c_n Z_{n-1}$

6. $Z_n = DR_n$

7. $a_n = d_n / R_n Z_n$

8. $t_{n+1} = t_n + a_n Z_n$

9. $G_{n+1} = G_n - a_n X DR_n$.

Terminate at $m$th step if $Z_m = M_m^g[G_{m+1}] = 0$. Then $t_{m+1}$ solves the Equation 7.3.5.

After obtaining $t_i$, where $t_i$ is the $i$th column vector of $T$, we need two more calculation steps for saving the storage. One is obtaining $c$ such that

$$c_j = \begin{cases} 0 & j = 1,2,\ldots,i-1 \\ k_j t_i & j = i,i+1,\ldots,r \end{cases}$$
where \( c_j \) is the elements of \( c \) and \( k_j \) is \( j \)th row vector of \( K \).

And move the row vector \( c \) to the \( i \)th row of \( K \) whose row size always bigger than or equal to the size of \( c \).

The second calculation is finding

\[ \hat{y}'DX_t^i. \]

The calculation of \( \hat{y}'DX_t^i \) seems to require matrix multiplication

But if we change the order of calculating to

\[ \hat{y}'DX_t^i = (X'D\hat{y})t^i \]

and use procedure outlined in 7.3.9, then this can be avoided.

In these calculations, notice that it is not required to store \( t_i \) for \( i = 1,2, \ldots, r \). At the end of these computations, we have \( \hat{y}'DX_t \) in a \( r \)-vector and the lower triangular part of \( K'T \) in \( K' \). The final step is to find \( (K'T)^{-1} \). Since \( K'T \) is \( r \times r \) symmetric matrix where \( r \) is less than or equal to \( p \), we may use the Cholesky method to find the inverse. In this case, the Cholesky method is easier to program than the conjugate gradient method.

The necessary and sufficient conditions of the testability for hypotheses, \( H_0: K\beta = 0 \), are (i) \( K'(X'X_k)^{-1}X_kX_o = K' \) and (ii) \( K' \) is of full row rank. For checking the condition (i), we would first solve the following equations

\[ (X'DX)B = X'D \]  \hspace{1cm} (7.3.11)

where the elements of the diagonal matrix \( D \) is the number of observations
per each cell. Since Equation 7.3.11 is consistent, there exists a solution for B. This equation is solved by applying the modified conjugate gradient method discussed in Section 7.1, to columns of B, i.e., solving \((X DX)b_i = X d_i\) for \(i = 1, \ldots, n\), where \(B = [b_1, \ldots, b_n]\) and \(D = [d_1, \ldots, d_n]\). Then we have
\[
B = (X DX)^{-1} D = (X'X)^{-1} X D.
\]

Premultiplying and postmultiplying 7.3.12 by \(K\) and \(X\), respectively, we have
\[
K BX = K (X'X)^{-1} X DX = K (X'X)^{-1} X X' .
\]
Thus, by checking whether \(K BX = K\), we can determine whether the first condition is satisfied.

The second condition, assuming that condition (i) is satisfied, requires that
\[
\text{rank } [K (X'X)^{-1} K] = \text{rank } [A X'X A]
= \text{rank } [A X']
= \text{rank } [K], \text{ where } K = A X'X .
\]
Thus, if the inverse of \(K (X'X)^{-1} K\) exists condition (ii) will be satisfied.

7.4 Analysis of Covariance

Hemmerle (1976b) described how his algorithm for solving the non-orthogonal analysis of variance problem could be applied to the analysis of covariance problem. It is equally simple to apply the conjugate gradient algorithm to the analysis of covariance problem.

If we write the nonorthogonal fixed-effects analysis of covariance model in matrix form as
\[
y = X \hat{\beta} + Zy + e, \quad (7.4.1)
\]

where \( Z = [z_1 \ldots z_q] \) is the Nxq matrix of covariate values and \( \gamma \) is a q-vector of unknown parameters associated with the covariates, then the normal equations for 7.4.1 to yield the so-called error normal equations

\[
\begin{align*}
X'X \hat{\beta} + X'Zy &= X'y \\
Z'X \hat{\beta} + ZZy &= Zy \quad (7.4.2)
\end{align*}
\]

These equations are equivalent to

\[
Z'[I - X_o(X'_oX_o)^{-1}X'_o]Zy = Z'[I - X_o(X'_oX_o)X'_o]y
\]

\[
\hat{\beta} = (X'_oX_o)^{-1}X'_o(y - Zy) \quad (7.4.3)
\]

If we let

\[
\tilde{\beta} = (X'_oX_o)^{-1}X'_oy
\]

and

\[
\tilde{\gamma} = (\tilde{\gamma}_1 \ldots \tilde{\gamma}_q),
\]

where

\[
\tilde{\gamma}_i = (X'_oX_o)^{-1}X'_o z_i,
\]

then

\[
\hat{\beta} = \tilde{\beta} - \tilde{\gamma}.
\]

The residual sum of squares, RSS, for the covariance Model 7.4.1 is given by

\[
\text{RSS} = y'y - y'X \hat{\beta} - y'Zy. \quad (7.4.4)
\]
To solve the first equations of 7.4.3, we define the nxq matrix of cell means of the q covariates by $\tilde{Z} = [\tilde{z}_1 | ... | \tilde{z}_q]$. This leads us to the equivalent problem

$$(z'z - \tilde{Z}D[b_1 | ... | b_q])\hat{Y} = z'y - \tilde{Z}D_{q+1}$$

(7.4.5)

where

$$b_i = X\tilde{Y}_i = X(XDX)'X\tilde{Z}_{i+1}, \quad b_{q+1} = X(XDX)'X\tilde{D}_q.$$ 

It is necessary then to compute the vector $\tilde{Y}_i$ and $b_i$ for $i = 1, 2, ..., q+1$. The computation of $\tilde{Y}_i$ is an analysis of variance problem that we can solve using the modified generalized conjugate gradient algorithm. The vector $b_i$ corresponds to the vector $B$ described in Section 6.1, and thus, is a direct by-product of the computations.

Let $T_k$ be the kth approximation to the second term in the coefficient matrix in 7.4.5. If we use Algorithm 7.1.1, then

$$T_k = z'DX[p_k(M^{-1}X)M^{-1}X'DZ]$$

where $P_k(.)$ is a polynomial of degree k. In the form described above, the algorithm should compute final approximations to $\hat{Y}_i$ and $b_i$ before $\hat{Y}$ and RSS can be computed. To obtain intermediate values, we define the operator $C_k$ where

$$C_k = I_N - X_oP_k(M^{-1}X)M^{-1}X_o.$$ 

We then denote the kth approximation $\hat{Y}_k$ to $\hat{Y}$ by a solution of
The approximations to RSS are computed from the formula
\[ (\text{RSS})_k = \hat{y}_k^\prime C_k \hat{y}_k - \hat{y}_k^\prime (Z C_k Z)^\prime \hat{y}_k. \]

It is possible to show that the sequence \((\text{RSS})_k\) is monotonically decreasing.

**Lemma 7.4.1** \(C_k\) is nonnegative definite.

**Proof:** Consider \(y \in \{P_k (M^{-1} X DX)\} M^{-1} X_o y\) for some vector of \(y\). If we use Equation 7.1.13 with \(\hat{\beta}_o = 0\), we have
\[
y' X_o (P_k (M^{-1} X DX)) M^{-1} X_o y' = y' DX P_k (M^{-1} X DX) M^{-1} X_D \hat{y}_k^\prime = y' DX \hat{y}_k^\prime \]
Thus, \(y' X_o (P_k (M^{-1} X DX)) M^{-1} X_o y\) is a monotonically increasing sequence by Lemma 6.2.2. This implies that \(\{y' C_k y\}\) is a monotonically decreasing sequence that is tending to the nonnegative residual sum of squares.

Since this does not depend on the choice of \(y\), \(C_k\) is nonnegative definite.

**Lemma 7.4.2** \(C_k = C_{k-1} - C_k\) is nonnegative definite.

**Proof:** From the proof of Lemma 7.4.1, it follows that
\[ y' C_k y \leq y' C_{k-1} y. \]
hence,
\[ \hat{y}^\top C_k \hat{y} = \hat{y}^\top (C_{k-1} - C_k) \hat{y} \geq 0. \]

Since this does not depend on the particular choice of \( \hat{y} \), \( C_k \) must be nonnegative definite.

**Lemma 7.4.3** \((RSS)_k \leq (RSS)_{k-1}\).

**Proof:** Since \( C_k \) is symmetric nonnegative definite, we can write \( C_k = H_k^\top H_k \). Then 7.4.6 can be rewritten as
\[ (H_k \hat{y})^\top (H_k \hat{y}) = (H_k \hat{y})^\top (H_k \hat{y}), \]
a set of normal equations in \( \hat{y}_k \). Thus, \( r(\hat{y}_k) = ||H_k \hat{y} - H_k \hat{y}^\dag ||_2^2 \) will be minimum for \( \hat{y}^\dag = \hat{y}_k \). Direct substitution shows that \( r(\hat{y}_k) = (RSS)_k \). Therefore, \( (RSS)_k \leq (RSS)_{k-1} \), or equivalently
\[
(RSS)_k = \hat{y}^\top C_k \hat{y} - 2 \hat{y}_k^\top (Z C_k \hat{y}) + \hat{y}_k^\top (Z C_k Z) \hat{y}_k
\]
\[
= (RSS)_{k-1} - (\hat{y} - Z \hat{y}_k) \top G_k (\hat{y} - Z \hat{y}_k)
\]
\[
\leq (RSS)_{k-1},
\]
since \( C_k \) is nonnegative definite by Lemma 7.4.2.

Because the sequence of estimates residual sum of squares, \( (RSS)_k \), is monotonically decreasing, the corresponding \( F \) statistic is also monotonically decreasing, and we may terminate the iteration whenever this value falls below the threshold \( F \) value.
8. COMPUTER PROGRAMMING IMPLEMENTATION

Analysis of variance data is frequently unbalanced and is likely to have missing cells. In some cases, the unbalanced data arises from a balanced design in which some of the observations are missing as a result of the experiment. Computationally, there are a number of missing data algorithms, notably Healy and Westmacott's iterative algorithm (Healy and Westmacott (1956)) and Rubin's noniterative algorithm (Rubin (1972)), which essentially fill in the missing values for use with balanced algorithms. Hemmerle (1974, 1976a,b) expands considerably on the work of Healy and Westmacott (1956), describing in detail how iterative improvement can be used to solve large non-orthogonal analysis of variance and analysis of covariance problems without having to specify the design matrix. His matrix free algorithm (Hemmerle (1982)) reduces much of the needed storage to generate an analysis of variance, but it requires more iterative steps than the algorithm based on the conjugate gradient algorithm. This chapter discusses the computer programming implementation for solving analysis of variance problems including missing cells case, based on the conjugate gradient method and the cell means model.

The general features and objectives of the FORTRAN program given in the Appendix, will be discussed in Section 8.1. Section 8.2 will discuss the computational algorithms for analysis of variance which can be partitioned into four basic component parts: the conjugate gradient algorithm (Algorithm 7.1.1), iterative rank algorithm
discussed in Section 6.3, a balanced factorial decomposition algorithm discussed in the last part of Section 7.2, and an E-P list construction algorithm. Section 8.3 will describe subroutines and functions, and logical diagram in the program. Some example output from the FORTRAN program are included in Section 8.4. Even though the FORTRAN program in the Appendix was implemented on an IBM PC, the discussion of computer implementation given in this chapter is very general.

8.1 Features and Objectives

In what follows, we shall describe a global algorithm for the analysis of variance with the following features.

(1) Balanced data, unbalanced data, and unbalanced data with missing cells are all processed by the algorithm. This is accomplished without losing the operational efficiencies obtained from the balanced data case and without applying approximate statistical methods to the unbalanced data.

(2) The algorithm is very general with respect to the kind of problem it can handle. Specifically, it bases its calculations upon an algebraically specified analysis of variance model of the type discussed in Searle (1971). This includes models with crossed factors, nested factors, and interactions between factors. This general model, along with the facility to handle missing cells, also includes such design as incomplete blocks, lattices, and Latin squares.
(3) Very large problems may be processed using a relatively small amount of computer storage. With two minor exceptions, no matrices are stored and no explicit matrix operations are performed. In particular neither $X_o$ or $X_o'X_o$ are stored or computed. An exact generalized inverse solution to the normal equations is obtained without ever computing a generalized inverse. The rank of the design matrix $X_o$ is obtained from the pattern of missing cells without explicit operations on $X_o$.

(4) The program has been written in FORTRAN 77 for microcomputers. A microcomputer executing the program needs at least 128K bytes with a numeric data co-processor or more than 128K bytes without a numeric data co-processor.

8.2 Elementary Algorithms

8.2.1 The conjugate gradient algorithm

The computational conjugate algorithm based on Algorithm 7.1.1 applies $E$ and $P$ operators discussed in Section 6.1 for balanced data, iteratively to obtain exact results for unbalanced data. The algorithm computes an analysis of variance for unbalanced data by successive computation of balanced analyses. Since the algorithm uses balanced $E$ and $P$ operators to obtain its results, there is no need to create indicator variables or the form of the design matrix $X_o$ or $X_o'X_o$. The algorithm operates upon a vector of cell sums of means and frequencies rather than upon a matrix (potentially large) so that array storage requirements are minimal. Since even when missing
cells are present the normal equations are consistent, convergence of
the algorithm is guaranteed. A useful monotonic property mentioned
in Section 6.2 serves to limit the numbers of iterations when testing
hypotheses. A balanced analysis of variance is a special case re­
quiring only one iteration. More detailed computational procedure
will be discussed in the next section.

8.2.2 Iterative rank algorithm

With unbalanced data, including missing cells, the rank of the
design matrix must be computed to determine degrees of freedom for
F statistics. The rank algorithm is necessary to determine rank
noniteratively from the pattern of missing cells for a given model when
possible. When this cannot be accomplished, the conjugate gradient
method or the iterative method discussed in Section 6.3 is used to
compute rank in one of two ways, depending upon the number of missing
cells. For large number of missing cells (> 8), the conjugate gradient
method is used to compute rank. Since the conjugate gradient algorithm
has to be applied for each filled cell, this becomes inefficient when
number of missing cells is smaller. For large number of observations
with few missing cells (≤ 8), the iterative method is used to compute
rank. This iterative method requires some matrix storage but monotonically
increasing quadratic convergence can be obtained.

8.2.3 Balanced factorial decomposition algorithm

The E and P operators discussed in Chapters 6 and 7 are identical
to the balanced analysis of variance operators, $M^*X'$ and $XM^*X'$
respectively, which can be obtained as linear combinations of the partial means from the observations. A very efficient algorithm for forming a complete set of factorial residuals for balanced data has been developed by Hartley (1956) and modified by Hemmerle (1964). This procedure was utilized in implementing the iterative algorithm, because of its efficiency and convenience. From the complete set of factorial residuals formed by this algorithm and considered as a vector, the linear combination of partial means specified by the E and P operators for wide range of linear models may be extracted.

We use the algorithm described in Hemmerle (1967) which does not require explicit use of the mapping functions that uses multiplications to locate the position of the appropriate means in the array of means. In order to construct the E and P operators needed for the conjugate gradient algorithm, elements of arrays in the complete factorial decomposition should be pooled as discussed in Section 7.2. The complete factorial decomposition will be stored in a single linear, one dimensional array, and it is mapped into arrays corresponding to the E and P operators, using the algebraically specified model. We use essentially the same algorithm described in Schlater and Hemmerle (1966) to carry out both the mapping and pooling.

8.2.4 E-P list construction algorithm

One of the important numerical calculations in the global algorithm is driven by what we have called the E-P list. This E-P list construction algorithm deciphers the model specification and
translates it into a numeric coding for subsequent use. The E-P list is constructed by scanning or parsing an algebraic model statement specified by the users. If there are \( n \) factors in the full analysis of variance model, then the E-P list will have \( 2^n \) terms (total mean, main effects, and interactions) in a full factorial model. Numeric entries are made in the list and are uniquely described by the model. The following procedure is a modification of Hemmerle (1964): For this procedure, increasing powers of 2 are assigned as numeric values to factor symbols and their associated subscripts, with unity for the first factor and its associated subscript.

The sum of numeric values of the factor symbols plus one is computed for each term in the model. For crossed factors or interactions between crossed factors, this sum is entered in the corresponding location of the E-P list. For nested factors and interactions involving nested factors, multiple entries are made in the E-P list. The sum is entered into each location of the E-P list must be pooled. The algorithm determines these locations from the numerical values of the factor symbols in the model term.

8.3 Logical Components

A logic diagram of the program in the Appendix is included in Figure 8.3.1. The following discussion will further describe the modules or components parts of the program. The program for the global algorithm consists of a main program and ten subroutines. The principal function of each component is described below.
Figure 8.3.3 Logic diagram of the program
Print ANOVA table

Compute Type I and Type III sums of squares

Print Type I and Type III sums of squares

$q$ estimate desired? Yes

Compute parameter estimate for $\hat{q}$

Print $\hat{q}$

Another model? Yes

Figure 8.3.3 (continued)
MAIN processes the factor and level statement; computes cell sums, cell frequencies and $\bar{y}$ as it reads the data.

IGET is used by MAIN program, subroutines SCAN and ANOVA to sequentially retrieved characters except blank, plus, comma, and equal sign from the input buffer.

SCAN processes the model statement to construct the E-P list.

RANK computes the degrees of freedom of applicable to data with no missing cells; restructures the cell frequencies when appropriate; checks for balance and alternative noniterative computations; computes rank noniteratively if possible or iteratively otherwise.

ANOVA is the principal numeric computational component for analysis of variance tables. It computes model sum of squares, error sum of squares, type I sums of squares, type III sums of squares, F-statistics, probability values and a solution to the normal equations.

CGM performs the conjugate gradient algorithm based on the algorithms of Chapter 6. The basic steps of the program is as follows:

1. $A \leftarrow Y$
2. $V \leftarrow Y$
3. $A \leftarrow P[A]$
4. $d \leftarrow A^T A$
5. $c \leftarrow d/dl$
6. $R \leftarrow A + c*R$
7. $Z \leftarrow D*R$
8. $t \leftarrow R^T Z$
9. $dl \leftarrow d$
10. $d \leftarrow d/t$
11. $Z \leftarrow V - d*Z$
\[(12) \quad t \gets d * dl\]

\[(13) \quad \text{test } t < 10^{-8} \text{ then return, else go to (19)}\]

\[(14)\* \quad A \gets V\]

\[(15)\* \quad A \gets E[A]\]

\[(16)\* \quad \text{test } A < 10^{-8} \text{ then return}\]

\[(17)\* \quad L \gets A + c * L\]

\[(18)\* \quad B \gets B + d * L\]

\[(19) \quad V \gets Z\]

\[(20) \quad A \gets Z\]

\[(21) \quad \text{Go to (3)}\]

where;

- \(Y\) is the vector of cell sums, \(D\) is the diagonal matrix of cell frequencies stored as a vector;

- \(R, V,\) and \(Z\) are work vectors of size \(n\), with \(n\) being the number of cells, and \(A, L,\) and \(B\) are also work vectors with the size of the total number of parameters;

- \(c, d, dl,\) and \(t\) are scalars. Initial value of \(c\) is zero and of \(dl\) is arbitrary;

- \(P[A]\) is the factor residual operator applied to vector \(A\). Step (3) above consists of a factorial decomposition of vector \(A\), followed by pooling the appropriate arrays of the decomposition back into \(A\);

- \(E[A]\) is the estimates operator applied to \(A\). For performing the \(E\) operator, step (15) calls EOPER subroutine;

step (14)* - (18)* need for estimating parameter \(\hat{\theta}\). For calculating the sum of squares, these steps will not operate.
**EPOPER** performs E operator for subroutine CGM by pooling the appropriate arrays of the decomposition.

**DECOMP** obtains a factorial decomposition of a given vector; determines the classification frequencies needed in RANK to restructure data; computes classification means and sums for MAIN.

**POOL** either moves the secondary array into the primary, duplicating entries where needed, or it pools the secondary array into the primary array by addition.

**LABEL** calculates the array of coefficients for the array map needed in pooling and produces output labels for classification means and for the solution to the normal equations.

**PF** computes \( F \) probability using method suggested by Davis and Khalil (1972) to evaluate \( F \)-distribution.

Of these logical components, all of SCAN may be deleted to reduce program storage, provided the user constructs and inputs the required E-P list.

### 8.4 Examples

This section contains computer outputs for four examples. The first example data is taken from Searle (1971, page 287). This example has two factors, 18 observations, and 4 missing cells (1,2), (2,3), (2,4), and (3,1). If the model command is specified

\[ Y = A B A \times B \text{ for the model } y_{ijk} = \mu + a_i + b_j + a_i b_j + e_{ijk} \text{ where} \]

...
i = 1, 2, 3 and j = 1, 2, 3, 4, the computer outputs are shown in Example 8.4.1. The degrees of freedom and sums of squares of A and B in type III are zero. That means hypotheses

\[ H_A: a_1 - \overline{a} + (\overline{ab})_1 - (\overline{ab}) = 0 \]

\[ a_2 - \overline{a} + (\overline{ab})_2 - (\overline{ab}) = 0 \]

and

\[ b_1 - \overline{b} + (\overline{ab})_1 - (\overline{ab}) = 0 \]

\[ b_2 - \overline{b} + (\overline{ab})_2 - (\overline{ab}) = 0 \]

\[ b_3 - \overline{b} + (\overline{ab})_3 - (\overline{ab}) = 0 \]

are not testable with these data. The second example is the same example used by Hemmerle (1982). With the same data and the same model, \[ Y = \mu + A + B + AB + C + e \], Hemmerle's program gave a different result for type III sum of squares for AB. The third example is an actual data set from a consulting problem analyzed at the Statistical Laboratory in the Department of Statistics of Iowa State University. This data set consisted of 233 observations and 4 factors. For the full model (we use command \[ Y = \ast \] in the program), it took about two minutes and twenty seconds for the computation to be performed in the IBM PC. The final example, a Latin square, is taken from Snedecor and Cochran (1982, page 273).

The table of cell sums, frequencies, and means and table of parameter estimates in examples, should be read the tables by the following way: For example, if we have two factors and two levels in each factor, 1 is represented by (1,1), 2 by (2,1), 3 by (1,2), and 4 by (2,2) respectively.
Example 8.4.1

INPUT DATA IS AS FOLLOWS:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>13.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>9.</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>12.</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7.</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>11.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>6.</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>12.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>12.</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>14.</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>9.</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>7.</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>14.</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>16.</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>10.</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>14.</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>11.</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>13.</td>
</tr>
</tbody>
</table>

*** Cell sums, frequencies, and means ***

<table>
<thead>
<tr>
<th>CELL</th>
<th>SUM</th>
<th>FREQ</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.30000000E+02</td>
<td>3</td>
<td>.10000000E+02</td>
</tr>
<tr>
<td>2</td>
<td>.18000000E+02</td>
<td>2</td>
<td>.90000000E+01</td>
</tr>
<tr>
<td>3</td>
<td>.26000000E+02</td>
<td>2</td>
<td>.13000000E+02</td>
</tr>
<tr>
<td>4</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.16000000E+02</td>
<td>2</td>
<td>.80000000E+01</td>
</tr>
<tr>
<td>6</td>
<td>.12000000E+02</td>
<td>1</td>
<td>.12000000E+02</td>
</tr>
<tr>
<td>7</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>.30000000E+02</td>
<td>2</td>
<td>.15000000E+02</td>
</tr>
<tr>
<td>9</td>
<td>.18000000E+02</td>
<td>2</td>
<td>.90000000E+01</td>
</tr>
<tr>
<td>10</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>.48000000E+02</td>
<td>4</td>
<td>.12000000E+02</td>
</tr>
<tr>
<td>12</td>
<td>.48000000E+02</td>
<td>5</td>
<td>.96000000E+01</td>
</tr>
<tr>
<td>1</td>
<td>.42000000E+02</td>
<td>4</td>
<td>.10500000E+02</td>
</tr>
<tr>
<td>2</td>
<td>.42000000E+02</td>
<td>3</td>
<td>.14000000E+02</td>
</tr>
<tr>
<td>3</td>
<td>.66000000E+02</td>
<td>6</td>
<td>.11000000E+02</td>
</tr>
<tr>
<td>4</td>
<td>.60000000E+02</td>
<td>6</td>
<td>.10000000E+02</td>
</tr>
<tr>
<td>2</td>
<td>.44000000E+02</td>
<td>4</td>
<td>.11000000E+02</td>
</tr>
<tr>
<td>3</td>
<td>.94000000E+02</td>
<td>8</td>
<td>.11750000E+02</td>
</tr>
</tbody>
</table>
MODEL: \( Y = A \times B \)

Factor A has 3 levels
Factor B has 4 levels

### ANOVA Table

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>MEAN SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
<td>7</td>
<td>82.00000000</td>
<td>11.7142857</td>
<td>2.092</td>
<td>.13995</td>
</tr>
<tr>
<td>ERROR</td>
<td>10</td>
<td>56.00000000</td>
<td>5.6000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORR. TOTAL</td>
<td>17</td>
<td>138.00000000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Type I Sum of Squares

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>1</td>
<td>2178.00000000</td>
<td>388.929</td>
<td>.00000</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>10.50000000</td>
<td>.937</td>
<td>.42348</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>36.7857143</td>
<td>2.190</td>
<td>.15232</td>
</tr>
<tr>
<td>AB TOTAL</td>
<td>2</td>
<td>34.7142857</td>
<td>3.099</td>
<td>.08965</td>
</tr>
</tbody>
</table>

### Type III Sum of Squares

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>.00000000</td>
<td>.000</td>
<td>.00000</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>.00000000</td>
<td>.000</td>
<td>.00000</td>
</tr>
<tr>
<td>AB TOTAL</td>
<td>2</td>
<td>34.7142857</td>
<td>3.099</td>
<td>.08965</td>
</tr>
</tbody>
</table>
### PARAMETER ESTIMATES

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>ESTIMATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>7.3333333</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.4166667</td>
</tr>
<tr>
<td>2</td>
<td>-1.8333333</td>
</tr>
<tr>
<td>3</td>
<td>1.4166667</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1.0000000</td>
</tr>
<tr>
<td>2</td>
<td>-0.3333333</td>
</tr>
<tr>
<td>3</td>
<td>1.6666667</td>
</tr>
<tr>
<td>4</td>
<td>-0.3333333</td>
</tr>
<tr>
<td>AB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.2500000</td>
</tr>
<tr>
<td>2</td>
<td>4.5000000</td>
</tr>
<tr>
<td>3</td>
<td>-7.7500000</td>
</tr>
<tr>
<td>4</td>
<td>-7.4166667</td>
</tr>
<tr>
<td>5</td>
<td>7.8333333</td>
</tr>
<tr>
<td>6</td>
<td>-0.4166667</td>
</tr>
<tr>
<td>7</td>
<td>2.5833333</td>
</tr>
<tr>
<td>8</td>
<td>-7.1666667</td>
</tr>
<tr>
<td>9</td>
<td>4.5833333</td>
</tr>
<tr>
<td>10</td>
<td>1.5833333</td>
</tr>
<tr>
<td>11</td>
<td>-5.1666667</td>
</tr>
<tr>
<td>12</td>
<td>3.5833333</td>
</tr>
</tbody>
</table>
Example 8.4.2

INPUT DATA IS AS FOLLOWS:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3.1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2.4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2.7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2.6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5.2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4.5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>5.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3.7</td>
</tr>
</tbody>
</table>

**Cell sums, frequencies, and means**

<table>
<thead>
<tr>
<th>CELL</th>
<th>SUM</th>
<th>FREQ</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.0000000E+01</td>
<td>2.0000000E+01</td>
<td>.25000000E+01</td>
</tr>
<tr>
<td>2</td>
<td>5.1000000E+01</td>
<td>2.0000000E+01</td>
<td>.25500000E+01</td>
</tr>
<tr>
<td>3</td>
<td>4.5000000E+01</td>
<td>1.0000000E+01</td>
<td>.45000000E+01</td>
</tr>
<tr>
<td>4</td>
<td>3.1000000E+01</td>
<td>1.0000000E+01</td>
<td>.31000000E+01</td>
</tr>
<tr>
<td>5</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.0000000E+01</td>
<td>1.0000000E+01</td>
<td>.50000000E+01</td>
</tr>
<tr>
<td>7</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(Missing cell)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3.6000000E+01</td>
<td>1.0000000E+01</td>
<td>.36000000E+01</td>
</tr>
<tr>
<td>10</td>
<td>4.5000000E+01</td>
<td>1.0000000E+01</td>
<td>.45000000E+01</td>
</tr>
<tr>
<td>11</td>
<td>7.8000000E+01</td>
<td>2.0000000E+01</td>
<td>.39000000E+01</td>
</tr>
<tr>
<td>12</td>
<td>3.7000000E+01</td>
<td>1.0000000E+01</td>
<td>.37000000E+01</td>
</tr>
<tr>
<td>.BC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.4600000E+02</td>
<td>5.0000000E+01</td>
<td>.29200000E+01</td>
</tr>
<tr>
<td>2</td>
<td>8.1000000E+01</td>
<td>2.0000000E+01</td>
<td>.40500000E+01</td>
</tr>
<tr>
<td>3</td>
<td>3.6000000E+01</td>
<td>1.0000000E+01</td>
<td>.36000000E+01</td>
</tr>
<tr>
<td>4</td>
<td>1.6000000E+02</td>
<td>4.0000000E+01</td>
<td>.40000000E+01</td>
</tr>
<tr>
<td>.A.C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8.1000000E+01</td>
<td>3.0000000E+01</td>
<td>.27000000E+01</td>
</tr>
<tr>
<td>2</td>
<td>5.1000000E+01</td>
<td>2.0000000E+01</td>
<td>.25500000E+01</td>
</tr>
<tr>
<td>3</td>
<td>9.5000000E+01</td>
<td>2.0000000E+01</td>
<td>.47500000E+01</td>
</tr>
<tr>
<td>4</td>
<td>4.5000000E+01</td>
<td>1.0000000E+01</td>
<td>.43000000E+01</td>
</tr>
<tr>
<td>5</td>
<td>7.8000000E+01</td>
<td>2.0000000E+01</td>
<td>.39000000E+01</td>
</tr>
<tr>
<td>6</td>
<td>7.3000000E+01</td>
<td>2.0000000E+01</td>
<td>.36500000E+01</td>
</tr>
<tr>
<td>.C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.2700000E+02</td>
<td>7.0000000E+01</td>
<td>.32428571E+01</td>
</tr>
<tr>
<td>2</td>
<td>1.9600000E+02</td>
<td>5.0000000E+01</td>
<td>.39200000E+01</td>
</tr>
</tbody>
</table>

AB.
MODEL: $Y = A \times B \times C$

Factor A has 3 levels
Factor B has 2 levels
Factor C has 2 levels

*** A N O V A TABLE ***

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>MEAN SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
<td>6</td>
<td>6.4541667</td>
<td>1.0756944</td>
<td>.939</td>
<td>.53859</td>
</tr>
<tr>
<td>ERROR</td>
<td>5</td>
<td>5.7283333</td>
<td>1.1456667</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORR. TOTAL</td>
<td>11</td>
<td>12.1825000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*** TYPE I SUM OF SQUARES ***

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>1</td>
<td>149.1075000</td>
<td>130.149</td>
<td>.00009</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
<td>2.7450000</td>
<td>1.198</td>
<td>.37578</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2.9008333</td>
<td>2.532</td>
<td>.17243</td>
</tr>
<tr>
<td>AB</td>
<td>2</td>
<td>.7016667</td>
<td>.306</td>
<td>.74911</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>ESTIMATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERCEPT</td>
<td>3.5027778</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-0.4194444</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.2777778</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6972222</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>-0.5583333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5583333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>-0.1583333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.2500000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.40833333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.15833333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.25000000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.4083333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.13333333</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.1333333</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>2.8512917</td>
<td>1.244</td>
<td>.36425</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2.4938889</td>
<td>2.177</td>
<td>.20012</td>
</tr>
<tr>
<td>AB</td>
<td>2</td>
<td>.8005208</td>
<td>.349</td>
<td>.72107</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>.1066667</td>
<td>.093</td>
<td>.77256</td>
</tr>
</tbody>
</table>
Example 8.4.3

INPUT DATA IS AS FollowS:

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>D</th>
<th>A</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>23.85</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>27.42</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>26.73</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>36.43</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>22.17</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>34.54</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>19.08</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>26.27</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>26.43</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>21.70</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>23.85</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>28.00</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>28.57</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>26.57</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>25.11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>26.70</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>42.16</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>26.74</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>24.65</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>39.36</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>28.29</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>22.19</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>26.67</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>19.25</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>26.82</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>26.25</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>25.06</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>22.07</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>19.75</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>19.47</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>36.19</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>25.12</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>20.90</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>26.48</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>28.20</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>23.94</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>22.39</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>23.71</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>25.82</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>32.58</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>22.32</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>21.62</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>24.77</td>
</tr>
<tr>
<td>3 1 2 3 &amp; 67.27</td>
<td>4 1 1 4 &amp; 55.91</td>
<td>4 1 2 5 &amp; 66.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 1 3 &amp; 54.51</td>
<td>4 2 2 4 &amp; 59.34</td>
<td>4 1 1 5 &amp; 69.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 1 3 &amp; 71.43</td>
<td>4 1 2 4 &amp; 68.19</td>
<td>4 2 2 5 &amp; 64.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 2 3 &amp; 40.56</td>
<td>4 1 1 4 &amp; 76.83</td>
<td>4 1 2 5 &amp; 63.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 2 3 &amp; 50.91</td>
<td>4 2 2 4 &amp; 56.86</td>
<td>4 2 1 5 &amp; 64.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 1 3 &amp; 65.00</td>
<td>4 1 2 4 &amp; 62.04</td>
<td>4 1 1 5 &amp; 50.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 1 3 &amp; 51.16</td>
<td>4 2 1 4 &amp; 66.19</td>
<td>4 2 2 5 &amp; 57.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 2 2 3 &amp; 47.90</td>
<td>4 1 1 4 &amp; 52.91</td>
<td>4 1 1 5 &amp; 49.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 1 2 3 &amp; 43.19</td>
<td>4 2 2 4 &amp; 51.87</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MODEL: Y = "

Factor R has 4 levels
Factor S has 2 levels
Factor A has 2 levels
Factor D has 5 levels

---

**ANOVA TABLE**

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>MEAN SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
<td>79</td>
<td>41910.7941918</td>
<td>530.5163822</td>
<td>9.870</td>
<td>.00000</td>
</tr>
<tr>
<td>ERROR</td>
<td>153</td>
<td>8223.4313000</td>
<td>53.7479170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORR. TOTAL</td>
<td>232</td>
<td>50134.2254918</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**TYPE I SUM OF SQUARES**

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>1</td>
<td>492641.6757082</td>
<td>9165.782</td>
<td>.00010</td>
</tr>
<tr>
<td>R</td>
<td>3</td>
<td>29317.8386061</td>
<td>181.823</td>
<td>.00000</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>98.3585230</td>
<td>1.830</td>
<td>.17812</td>
</tr>
<tr>
<td>RS</td>
<td>3</td>
<td>43.7647576</td>
<td>.271</td>
<td>.84594</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1617.3281716</td>
<td>30.091</td>
<td>.00000</td>
</tr>
<tr>
<td>RA</td>
<td>3</td>
<td>28.7765924</td>
<td>.178</td>
<td>.91087</td>
</tr>
<tr>
<td>SA</td>
<td>1</td>
<td>377.9989369</td>
<td>7.033</td>
<td>.00885</td>
</tr>
<tr>
<td>RSA</td>
<td>3</td>
<td>40.6828248</td>
<td>.252</td>
<td>.85960</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>7492.2290641</td>
<td>34.849</td>
<td>.00000</td>
</tr>
<tr>
<td>RD</td>
<td>12</td>
<td>1657.9786755</td>
<td>2.571</td>
<td>.00394</td>
</tr>
<tr>
<td>SD</td>
<td>4</td>
<td>34.4373314</td>
<td>.160</td>
<td>.95810</td>
</tr>
</tbody>
</table>
192

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>Sum of Squares</th>
<th>F-Value</th>
<th>P-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>3</td>
<td>29720.3780840</td>
<td>184.320</td>
<td>.00010</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>54.8149796</td>
<td>1.020</td>
<td>.31415</td>
</tr>
<tr>
<td>RS</td>
<td>3</td>
<td>83.7970033</td>
<td>.520</td>
<td>.66935</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1637.7879058</td>
<td>30.472</td>
<td>.00000</td>
</tr>
<tr>
<td>RA</td>
<td>3</td>
<td>27.2516645</td>
<td>.169</td>
<td>.91717</td>
</tr>
<tr>
<td>SA</td>
<td>1</td>
<td>367.2291018</td>
<td>6.832</td>
<td>.00985</td>
</tr>
<tr>
<td>RSA</td>
<td>3</td>
<td>34.6736775</td>
<td>.215</td>
<td>.88586</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>7486.5692995</td>
<td>34.823</td>
<td>.00010</td>
</tr>
<tr>
<td>RD</td>
<td>12</td>
<td>1658.0549765</td>
<td>2.571</td>
<td>.00394</td>
</tr>
<tr>
<td>SD</td>
<td>4</td>
<td>31.0261854</td>
<td>.144</td>
<td>.96527</td>
</tr>
<tr>
<td>RSD</td>
<td>12</td>
<td>347.2816596</td>
<td>.538</td>
<td>.88686</td>
</tr>
<tr>
<td>AD</td>
<td>4</td>
<td>95.7769451</td>
<td>.445</td>
<td>.77555</td>
</tr>
<tr>
<td>RAD</td>
<td>12</td>
<td>292.4424175</td>
<td>.453</td>
<td>.93835</td>
</tr>
<tr>
<td>SAD</td>
<td>4</td>
<td>166.8830367</td>
<td>.776</td>
<td>.54225</td>
</tr>
<tr>
<td>RSAD</td>
<td>12</td>
<td>315.8819475</td>
<td>.490</td>
<td>.91841</td>
</tr>
</tbody>
</table>

*** Type III Sum of Squares ***
Example 8.4.4

INPUT DATA IS FOLLOWS:

R C T Y
1 1 1 608.
1 2 2 885.
1 3 3 940.
2 1 2 715.
2 2 3 1087.
2 3 1 766.
3 1 3 844.
3 2 1 711.
3 3 2 832.

MODEL: Y = R C T

Factor R has 3 levels
Factor C has 3 levels
Factor T has 3 levels

*** A N O V A TABLE ***

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>MEAN SQUARES</th>
<th>F-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODEL</td>
<td>6</td>
<td>156550.000000</td>
<td>26091.666667</td>
<td>10.775</td>
</tr>
<tr>
<td>ERROR</td>
<td>2</td>
<td>4842.888889</td>
<td>2421.444444</td>
<td></td>
</tr>
<tr>
<td>CORR. TOTAL</td>
<td>8</td>
<td>161392.888889</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P-VALUE .08735

*** TYPE I SUM OF SQUARES ***

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>1</td>
<td>6064727.111111</td>
<td>2504.591</td>
<td>.00040</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
<td>5900.222222</td>
<td>1.218</td>
<td>.45079</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>47213.555556</td>
<td>9.749</td>
<td>.09303</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>103436.222222</td>
<td>21.358</td>
<td>.04473</td>
</tr>
<tr>
<td>TOTAL</td>
<td>6</td>
<td>6221277.111111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TYPE III SUM OF SQUARES

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>DF</th>
<th>SUM OF SQUARES</th>
<th>F-VALUE</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>2</td>
<td>5900.2222222</td>
<td>1.218</td>
<td>.45079</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>47213.5555556</td>
<td>9.749</td>
<td>.09303</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>103436.2222222</td>
<td>21.358</td>
<td>.04473</td>
</tr>
</tbody>
</table>

### PARAMETER ESTIMATES

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>ESTIMATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERCEPT</td>
<td>820.8888889</td>
</tr>
<tr>
<td>R</td>
<td>-9.8888889</td>
</tr>
<tr>
<td></td>
<td>35.1111111</td>
</tr>
<tr>
<td></td>
<td>-25.2222222</td>
</tr>
<tr>
<td></td>
<td>-98.5555556</td>
</tr>
<tr>
<td></td>
<td>73.4444444</td>
</tr>
<tr>
<td></td>
<td>25.1111111</td>
</tr>
<tr>
<td>T</td>
<td>-125.8888889</td>
</tr>
<tr>
<td></td>
<td>-10.2222222</td>
</tr>
<tr>
<td></td>
<td>136.1111111</td>
</tr>
</tbody>
</table>
9. BIBLIOGRAPHY


I wish to express my sincere thanks to Professor Mervyn M. Marasinghe and Professor William J. Kennedy for suggesting this area of research, for their encouragement, and for their valuable advice.

I wish to thank Mr. Dan Schnell and Ms. Kathleen Shelley for English editing and Ms. Darlene Wicks for her patience and skill in typing of this manuscript.

Finally, I wish to express my deepest gratitude to my parents, my wife, Inkwang, and especially my mother-in-law, Ms. Hawchan Lee, for their understanding, encouragement, and moral support throughout the years of my studying at Iowa State University.
11. APPENDIX
The Conjugate Gradient Method FORTRAN Program.

Main program reads the factor, levels statement, computes cell sums, frequencies, and y'y as it reads the data.

W is a linear array which is used for all of numeric computation.
LSTFI is an array used in formation and subsequent manipulation of the arrays in the factorial decomposition of size $2^{n}$ where $n$ is number of factors.
LER is the E-P list which is also of size $2^n$.
LIV is an array of the numerical values assigned to the factor symbol and their associated subscripts.
LLIM contains the number of levels for each factor and is of size $n$.
LT is a temporary work array of size $n$.
LP is also a temporary array of size 10.
LD is an alpha numeric array of size 10 containing the digits 0 - 9.
LE is an alphanumeric array for factor symbols of size $n$.
IA is an alphanumeric input buffer.

```
DIMENSION W(3400),LSTFI(64),LER(64),LIV(6),LLIM(6)
DIMENSION LT(6),LP(10),LD(10)
CHARACTER*32 FNAME,FNAMEO
DOUBLE PRECISION W,YPY,TEMP
CHARACTER*2 lA
COMMON /TPAR/YPY,LE(6)
COMMON /APAR/NCELLS,LD1,LD2,LV,LB,LA
COMMON /RPAR/IA(82),II
COMMON /BPAR/IIP,IDF,NOBS,ISA
DATA IIF/'F'/,IY/'Y'/,ILP('''IP'/)'/,IBLANK'/ '</DATA LD/'0','1','2','3','4','5','6','7','8','9'/
```

This program performs analysis of variance for balanced and unbalanced data from experimental designs using the conjugate gradient method. This program is designed for micro computers and programed for interactive mode.
The program accepts up to 6 factors and up to 300 model degrees of freedom. Also the program does not care about the number of observations.

L=80
MAXN=6
MAXW=3400
WRITE(*,1)
1 FORMAT(15X,'Welcome to CGM Analysis of Variance Program ***',
       '/15X,' Output file name(for printer, type in PRN):'.
       READ(*,142)FNAME0
OPEN(6,FILE=FNAME0,STATUS='NEW')
WRITE(*,6)
6 FORMAT(15X,'Please specify the factor symbol & number',
       '/15X,' EX: A(3) B(2)C(2)--'.
       READ(*,10)(IA(I),I=1,L)
10 FORMAT(80A1)

II=1
ICD=II
N=0
IF(II .GT. L) GO TO 500
IC=IGET(L)
50 N=N+1
IF(N .GT. MAXN) GO TO 500
LE(N)=IC
LEVEL=0
60 IF(II .GT. L) GO TO 500
IC=IGET(L)
IF(IC .EQ. ILP) GO TO 60
DO 70 J=1,10
   IF(LD(J) .EQ. IC)LEVEL=LEVEL*10+J-1
70 CONTINUE
IF(IC .NE. IRP) GO TO 60
LLIM(N)=LEVEL
IF(II .GT. L) GO TO 100
IC=IGET(L)
IF(IC .EQ. IBLANK .AND. II .GT. L) GO TO 100
GO TO 50
Determine number of cells (=NCELLS) and
M = 2^N. Create LIST, LIV, and LSTFI.

EX:
NPRIME: (3+1)(2+1)(2+1),
NCELLS: 3^2 = 12,
LSTFI: IJK, JK, IK, K, IJ, J, I, 1
12, 4, 6, 2, 6, 2, 3, 1
LIV: 1, 2, 4.

100 NCELLS = 1
NPRIME = 1
DO 110 I = 1, N
NPRIME = NPRIME * (LLIM(I) + 1)
110 NCELLS = NCELLS * LLIM(I)
120 LISTFI(1) = NCELLS
M = 1
K1 = 2
DO 130 I = 1, N
DO 120 J = 1, M
LISTFI(K1) = LISTFI(J) / LLIM(I)
120 K1 = K1 + 1
130 M = 2 * M

* Initialize the pointer to vectors in W array and check the problem size.
LD1 = NCELLS
LD2 = LD1 + NCELLS
LV = LD2 + NCELLS
LB = LV + NCELLS
LA = LB + NCELLS
NW = LA + NPRIME
INCW = NW - MAXW
IF (INCW .GT. 0) GO TO 490

DO 140 I = 1, NCELLS
ID1 = LD1 + I
W(ID1) = .0D0
140 W(ID1) = .0D0

* Read input data from FILE and check
WRITE(*,141)
141 FORMAT(15X,'***Input your data file name---' ,).
READ(*,142)FNAME
142 FORMAT(A)
OPEN(7,FILE=FNAME)
CALL LABEL(M,0,LLIM,N,LIV,LP)
ICD=ID
YPY=.000
NOBS=0

150 READ(7,*,END=190)(LT(I),1=1,N),TEMP
1=1
DO 160 J=1,N
160 I=I+(LT(J)-1)-LP(J)
IF(I .GT. NCELLS) GO TO 500
IF(I .LT. 1) GO TO 500
ID1=LD1+I
W(I)=W(I)+TEMP
W(ID1)=W(ID1)+1.0D0
NOBS=NOBS+1
YPY=YPY+TEMP
GO TO 150
190 WRITE(*,191)
191 FORMAT(' Read all data!!! ***',/,' Do you want to see',
' cell and classification mean? (Y or N)----',..)
READ(*,10)(IA(I),I=1,L)
II=1
ICD=IY
IC=IGET(L)
IF(IC .NE. IY) GO TO 300
WRITE(*,200)
WRITE(6,200)
200 FORMAT(' *** Cell sums, frequencies, and means ***',///,
' CELL',7X,'SUM',9X,'FREQ',9X,'MEAN')
DO 210 I=1,NCELLS
ID1=LD1+I
IF(W(ID1) .GT. .000) THEN
  TEMP=W(I)/W(ID1)
  WRITE(*,230)I,W(I),W(ID1),TEMP
  WRITE(6,230)I,W(I),W(ID1),TEMP
ELSE
WRITE(*,220)I
WRITE(6,220)I
ENDIF

210 CONTINUE
220 FORMAT(1X,I6,12X,'(Missing cell)')
230 FORMAT(1X,I6,1X,E16.8,F6.0,1X,E16.8)

C
C * Compute classification means. *
C

WRITE(*,250)
250 FORMAT(' Classification sums, frequencies, and means. ***')
DO 260 I=1,NCELLS
   IAA=LA+I
   W(IAA)=W(I)
   CALL DECOMP(1,LA,W,M,LSTFI,N,LE,LIV,LLIM,LP)
   DO 270 I=1,NCELLS
      ID1=LD1+I
      IAA=LA+I
      W(IAA)=W(ID1)
      CALL DECOMP(2,LA,W,M,LSTFI,N,LE,LIV,LLIM,LP)
C
C * Read the model and get SSR SSE estimate *
C * beta and so on. *
C

300 WRITE(*,350)
350 FORMAT(' Specify your model-----')
   READ(*,10)(IA(I),I=1,L)
   II=1
   IC=IGET(L)
   ICD=IC
   IF(IC .NE. IY) GO TO 500
   IERR=0
   CALL SCAN(M,LER,N,LE,LIV,LLIM,LP,IERR)
   IF(IERR .EQ. 0) THEN
      WRITE(*,355)
      WRITE(6,354)(IA(I),I=1,L)
      DO 351 I=1,N
         IA(I)=LE(I)
      ENDIF
      WRITE(6,352)IA(I),LLIM(I)
      CALL ANOVA(W,M,LSTFI,LER,N,LIV,LLIM,LP)
   ELSE
      WRITE(*,355)
      GO TO 300
   ENDIF
   WRITE(*,355)
   WRITE(6,355)
   FORMAT(*,355)

352 FORMAT(/,10X,'Factor',5X,A1,5X,'has',I4,2X,'levels')
354 FORMAT(IH1,//,1X,'MODEL: ',72A1,/,8X,60A1)
355 FORMAT(' You specify incorrect model. Try again!!!')

C
C * Finish. Try another model, if you want. *
C

WRITE(*,360)
360 FORMAT(' Do you want another model? (Y or N)',.)
   READ(*,10)(IA(I),I=1,L)
   II=1
   IC=IGET(L)
   IF(IC .NE. IY) GO TO 550
   GO TO 300

490 WRITE(*,495)
495 FORMAT(' Program size is too big.')
   GO TO 550

500 IA(1)=ICD
   WRITE(*,510)IA(1)
510 FORMAT(' Error in ',A1,' specification.')
   GO TO 5

550 WRITE(6,*), ''
   STOP 'Program well done.'
END
FUNCTION IGET(L)
COMMON /RPAR/IA(82),II
DATA IBLANK/' '/,IPLUS/'+/',ICOMMA/',IEQUAL/'='
10 IGET=IA(II)
  II=II+1
  IF(II .GT. L) RETURN
  IF(IGET .EQ. IBLANK .OR. IGET .EQ. IPLUS) GO TO 10
  IF(IGET .EQ. ICOMMA .OR. IGET .EQ. IEQUAL) GO TO 10
  RETURN
END
Subroutine SCAN processes the model statement to construct the E-P list.

SUBROUTINE SCAN(M, LER, N, LE, LIV, LLIM, LP, LERR)
  DIMENSION LER(1), LE(1), LIV(1), LLIM(1), LP(1)
  COMMON /RPAR/ IA(82), II
  COMMON /BPAR/ IIP, IDF, NOBS, ISA
  DATA ILP/('/', IRP/')', ISTAR/'*'/', ISLASH/'/'/'', IBLANK/' '/

  * Read the model equations and construct the E-P list. *

  I = 80
  IF(II .GT. L) GO TO 250
  IC = IGET(L)
  IF(IC .EQ. ISTAR) GO TO 140
  M1 = M - 1
  DO 10 I = 1, M1
    LER(I) = 0
    LER(M) = 1
  10 CONTINUE

  * Scan term to construct E-P list. *

  40 DO 50 I = 1, N
    LP(I) = 0
    NE = 0
    NVS = 0
    DO 60 I = 1, N
      IF(IC .NE. LE(I)) GO TO 60
      LP(I) = LIV(I)
      NE = NE + 1
      NVS = NVS + LIV(I)
    60 CONTINUE
    IF(NE .EQ. 0) GO TO 250
    IF(II .GT. L) GO TO 250
    IC = IGET(L)
    IF(IC .EQ. ISTAR) GO TO 80
    IF(II .GT. L) GO TO 250
    IC = IGET(L)
    GO TO 60
  80 K = M - NVS
  LER(K) = NVS + 1
  IF(NE .EQ. 1) GO TO 130
  NE = M - NVS + 1

  * Check nested or not. If the model is nested *
  * find the nested factors. *

  DO 120 I = NE, M1
NUM=M-I
DO 100 J=1,N
  K=N-J+1
  NUM=NUM-LP(K)
  IF(NUM .GT. 0) GO TO 100
  IF(NUM .EQ. 0) GO TO 110
  NUM=NUM+LP(K)
100 CONTINUE
  GO TO 120
110 IF(LER(I) .EQ. 0) LER(I)=NVS+1
120 CONTINUE
130 IF(II .GT. L) GO TO 160
   IF(IC .EQ. IBLANK .AND. II .GT. L) GO TO 160
   GO TO 40
140 DO 150 I=1,M
150 LER(I)=M-I+1
160 RETURN
250 IERR=1
RETURN
END
Subroutine RANK computes the degrees of freedom of applicable data with no missing cells; restructures the cell frequencies when appropriate; checks for balance and alternative computations; computes rank noniteratively if possible or iteratively otherwise.

SUBROUTINE RANK(W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
DIMENSION W(1),LSTFI(1),LER(1),LIV(1),LLIM(1),LT(1),LP(1),B(1)
DIMENSION Q(8,8),QT(8)
DOUBLE PRECISION W,B,YPY,S,TRACE,TEMP,QT,Q
COMMON /APAR/NCELLS,LD1,LD2,LV,LA
COMMON /BPAR/IIP,IRANK,NOBS,ISA
COMMON /TPAR/YPY,LE(6)
IRANK=0
IXST=0
M1=M-1
NSUBS=N
DO 30 I=1,N
   LP(I)=0
   INC1=LIV(I)
   INC2=LIV(N)/INC1
   LOC=1
   DO 20 J=1,INC2
      DO 10 K=1,INC1
         IF(LER(LOC) .GT. 0) LP(I)=LP(I)+1
      10   LOC=LOC+1
   20   LOC=LOC+INC1
   IF(LP(I) .EQ. 0) NSUBS=NSUBS-1
30 CONTINUE

DO 40 I=1,N
   IF(LP(I) .EQ. 0) GO TO 40
   IF(LP(I) .NE. LIV(NSUBS)) GO TO 90
40 CONTINUE
GO TO 80
50 NSUBS=N
DO 70 I=1,N
   LP(I)=LIV(N)
70 CONTINUE
IXST=1
DO 110 I=1,N
   IF(LP(I) .EQ. 0) GO TO 110
   IF(LP(I) .NE. LIV(NSUBS)) GO TO 90
90 CONTINUE
NO1=1
DO 110 I=1,N
IF(LP(I) .EQ. 0)LPOUT=LPOUT*LLIM(I)
IF(LP(I) .NE. 0)NO1=NO1+LIV(I)
110 CONTINUE

C  ********************************************************************
C  * Compute degree of freedom.  *
C  ********************************************************************

IDF=0
DO 130 I=1,M
  IF( LER(I) .LE. 0 ) GO TO 130
  NO2=M-I+1
  CALL LABEL(NO2,0,LLIM,N,LIV,LP)
  K=1
  DO 120 J=1,N
    IF(LP(J) .NE. 0) K=K*(LLIM(J)-1)
 120 CONTINUE
  IDF=IDF+K
130 CONTINUE

C  ********************************************************************
C  * Form cell frequency array.  *
C  ********************************************************************

IF(NSUBS .EQ. N)GO TO 240
DO 140 I=1,NCELLS
  ID1=LD1+I
  IA=LA+I
140  W(IA)=W(ID1)
    CALL DECOMP(1,LA,W,M,LSTFI,N,LE,LIV,LLIM,LP)
  NS=LA
  J=M-NO1
  DO 150 I=1,J
150  NS=NS+LSTFI(I)
    CALL LABEL(NO1,0,LLIM,N,LIV,LP)
    CALL FOOL(0,LD2,NS,W,N,LLIH,LT,LP)
C  ********************************************************************
C  * Check for a square effective X matrix.  *
C  ********************************************************************

160 IF(IXST .EQ. 1) GO TO 220
  J=LD2+1
  NS=0
  DO 170 I=1,NCELLS
    ID2=LD2+I
    IF(W(ID2) .EQ. .0D0) GO TO 260
    IF(W(ID2) .NE. W(J)) NS=1
 170 CONTINUE
  IF(NS .EQ.1) GO TO 210
  IRANK=IDF
180 DO 190 I=1,NCELLS
    ID2=LD2+I
190  W(ID2)=W(ID2)/DBLE(LPOUT)
  200 GO TO 460
All elements of effective D matrix are not zero.

C

IRANK=IDF
GO TO 460

* X is full rank.

DO 230 I=1,NCELLS
   ID2=LD2+I
   IF(W(ID2) .NE. 0.0D0) IRANK=IRANK+1
230 CONTINUE
   IRANK=IRANK/LPOUT
   GO TO 180

DO 250 I=1,NCELLS
   ID1=LD1+I
   ID2=LD2+I
250 W(ID2)=W(ID1)
GO TO 160

NS=0

DO 270 I=1,NCELLS
   ID1=LD1+I
   ID2=LD2+I
   IF(W(ID1) .EQ. .0D0) NS=NS+1
270 W(ID2)=W(I)
   IF(NS .LE. 8) GO TO 330
   W(NO2)=W(ID1)
   IF(W(NO2) .EQ. .0D0) THEN
      W(ID1)=.0D0
   ELSE
      W(ID1)=1.0D0
   ENDIF

CONTINUE
   TRACE=.0D0
   DO 300 J=1,NCELLS
      W(J)=.0D0
      IF(J.EQ.I) W(J)=1.0D0
300 CONTINUE
   CALL CGM(0,S,W,M,LSTFI,LER,N,LIV,LILIM,LT,LP,B)
   TRACE=TRACE+S
300 CONTINUE
GO TO 410

C
* If number of missing cells are less than or equal to 8, find rank of X using iterative method. *
C

330 K=1
DO 360 I=1,NCELLS
   ID1=LD1+I
   IF(W(ID1).NE. 0.0D0) GO TO 360
   DO 340 J=1,NCELLS
      W(J)=.0D0
      IF(J .EQ. I) W(J)=1.0D0
   CONTINUE
   CALL CGM(4,S,W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
   LL=1
   DO 350 J=1,NCELLS
      ID1=LD1+J
      NO2=LA+J
      IF(W(ID1).NE. 0.0D0) GO TO 350
      Q(K,LL)=W(NO2)
      LL=LL+1
350 CONTINUE
   K=K+1
360 CONTINUE

C
* Power Q and compute trace(I - Q**((2*K)) *
C
 TEMP=DBLE(IDF)
 DO 370 I=1,NS
   TEMP=TEMP-Q(I,I)
   LL=0
370 DO 390 J=1,NS
   DO 380 I=J,NS
      QT(I)=.0D0
      DO 380 K=1,NS
         QT(I)=QT(I)+Q(K,J)*Q(K,I)
      CONTINUE
380 Q(K,J)=QT(K)
390 Q(K,J)=QT(K)
   TRACE=DBLE(IDF)
   DO 400 I=1,NS
      TRACE=TRACE-Q(I,I)
   DO 400 J=I,NS
   DO 400 K=1,NS
   Q(I,J)=Q(J,I)
   LL=LL+1
   TEMP=TRACE-TEMP
   IF(TEMP .LE. 1.0D-1) GO TO 430
   IF(LL .GE. 100) GO TO 500
   TEMP=TRACE
GO TO 375
410 DO 420 I=1,NCELLS
   ID1=LD1+I
   ID2=LD2+I
   NO2=LB+I
   W(I)=V(ID2)
420 W(ID1)=W(NO2)
   GO TO 450
430 DO 440 I=1,NCELLS
   ID2=LD2+I
440 W(I)=W(ID2)
450 IRANK=TRACE+.9D0
460 RETURN
500 WRITE(*,505)
505 FORMAT(10X,'******RANK DOES NOT CONVERGE. ******')
RETURN
END
C Subroutine ANOVA is the principal numeric computational components for analysis of variance tables. It computes model sum of squares, residual sum of squares, type I sums of squares, type III sums of squares, F-statistics and probability values and a solution of the normal equations.

SUBROUTINE ANOVA(W,M,LSTFI,LER,N,LIV,LLIM,LP)
DIMENSION W(1),LSTFI(1),LER(1),LIV(1),LLIM(1),LP(1),B(500)
DIMENSION JR(66),LT(6)
DOUBLE PRECISION W,S,YPY,B,PVAL
CHARACTER*2 IA
COMMON /APAR/ NCELLS,LD1,LD2,LV,LB,LA
COMMON /BPAR/IIP,IDF,NOBS,ISA
COMMON /TPAR/YPY,LE(6)
COMMON /RPAR/IA(82),II
DATA IY/'Y'/',IBLANK/' '/
C
C * Check balanced model or not. *
C
ISA=0
J=LD1+1
DO 5 I=2,NCELLS
   N1=LD1+I
   IF(W(N1).NE.W(J)) GO TO 7
5 CONTINUE
ISA=1
7 M1=M-1
   IIP=0
   DO 10 1=1,M
      J=M-I+1
      IF(LER(J) .EQ. I) IIP=IIP+LSTFI(J)
10 CONTINUE
IF(IIP .GT. 500) GO TO 300
C
C * First, find model sum of square and residual sum of square. *
C
C
N1=0
DO 40 I=1,M
II=M-I+1
   IF(LER(II) .NE. I) GO TO 40
   DO 20 J=1,M
      IF(LER(J) .GT. I) LER(J)=-LER(J)
20 CONTINUE
   CALL RANK(W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
   CALL CGM(0,S,W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
   N1=N1+1

...
B(N1)=S
JR(N1)=IDF
DO 30 J=1,M
30 LER(J)=IABS(LER(J))
40 CONTINUE
NN1=N1-1
DO 50 I=1,NN1
   J=N1-I+1
   JR(J)=JR(J)-JR(J-1)
50 B(J)=B(J)-B(J-1)

C
* B(N1+1)=SSM
* B(N1+2)=SSE
* B(N1+3)=Corrected total
* B(N1+4)=Mean square for model
* B(N1+5)=Mean square for error
* B(N1+6)=F-value
* IDOF1 =Degree of freedom for model
* IDOF2 =Degree of freedom for c. total
* IDOF3 =Degree of freedom for error
C
B(N1+1)=S-B(1)
B(N1+2)=YPY-S
B(N1+3)=YPY-B(1)
IDOF1=IDF-1
IDOF2=NOBS-1
IDOF3=NOBS-IDF
B(N1+4)=B(N1+1)/DBLE(IDOF1)
B(N1+5)=B(N1+2)/DBLE(IDOF3)
B(500)=B(N1+5)
B(N1+6)=B(N1+4)/B(N1+5)
WRITE(55)
WRITE(55)
55 FORMAT(//,25X,'*** A N O V A T A B L E ***',//,1X,'SOURCE',9X,* ' DF',8X,'SUM OF SQUARE',9X,'MEAN SQUARE',8X,'F-VALUE')
WRITE(60,IDOF1,B(N1+1),B(N1+4),B(N1+6),IDOF3,B(N1+2),*
   B(N1+5),IDOF2,B(N1+3)
WRITE(60,IDOF1,B(N1+1),B(N1+4),B(N1+6),IDOF3,B(N1+2),*
   B(N1+5),IDOF2,B(N1+3)
60 FORMAT(1X,'MODEL',10X,I4,1X,F20.7,1X,F19.7,5X,F10.3,/,1X, *
'ERROR',10X,I4,1X,F20.7,1X,F19.7,/,1X,'CORR. TOTAL',4X, *
I4,1X,F20.7)
CALL PF(IDOF1,IDOF3,B(N1+6),PVAL,IDOF2)
WRITE(62)PVAL
WRITE(6,62)
62 FORMAT(69X,'P-VALUE',//,69X,F7.5)
WRITE(6,65)
WRITE(6,65)
65 FORMAT(//,25X,'*** TYPE I SUM OF SQUARES ***',//,1X,'SOURCE', *
8X,'DF',9X,'SUM OF SQUARE',8X,'F-VALUE',3X,'P-VALUE',/)
NN1=0
DO 80 I=1,M
   J=M-I+1
   IF(LER(J) .NE. I) GO TO 80
   NN1=NN1+1
   B(499)=0.0D0
   IF(JR(NN1) .GT. 0) B(499)=(B(NN1)/DBLE(JR(NN1)))/B(500)
   CALL PF(JR(NN1),IDOF3,B(499),PVAL,IDOF2)
   CALL LABEL(I,IBLANK,LE,N,LIV,LP)
   WRITE(*,85)JR(NN1),B(NN1),B(499),PVAL
   WRITE(6,85)JR(NN1),B(NN1),B(499),PVAL
80 CONTINUE
85 FORMAT('+',11X,I6,2X,F20.7,5X,F10.3,3X,F7.5)
WRITE(*,95) S
WRITE(6,95) S
95 FORMAT(IX,'TOTAL',14X,F20.7)

C
C  ********************************************************************
C  * Type III sum of square.                                       *
C  ********************************************************************
C
NN1=1
DO 98 I=2,M
   J=M-I+1
   IF(LER(J) .NE. I) GO TO 98
   NN1=NN1+1
   IF(JR(NN1) .EQ. 0) LER(J)=-99
98 CONTINUE
B(N1+1)=S
JR(N1+1)=IDF
NN1=1
DO 120 I=2,M
   II=M-I+1
   IF(LER(II) .EQ. -99) THEN
      NN1=NN1+1
      B(NN1)=B(N1+1)
      JR(NN1)=JR(N1+1)
      GO TO 120
   ENDIF
   IF(LER(II) .NE. I) GO TO 120
   JR(II)=IABS(JR(II))
120 CONTINUE

C
C  ********************************************************************
C  ********************************************************************
C
100 CONTINUE
CALL RANK(W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
CALL CGM(0,S,W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
NN1=NN1+1
B(NN1)=S
JR(NN1)=IDF
DO 115 J=1,M1
   IF(LER(J) .EQ. -99) GO TO 115
   LER(J)=IABS(LER(J))
115 CONTINUE
DO 125 I=1,M1
   J=M-I+1
   IF(LER(I).EQ.-99) LER(I)=J
125 CONTINUE
DO 130 I=2,N1
   JR(I)=JR(N1+1)-JR(I)
B(I)=B(N1+1)-B(I)
WRITE(*,135)
WRITE(6,135)
135 F0RMAT(///,25X,TYPE III SUM OF SQUARES ***',//,
*8X,'DF',9X,'SUM OF SQUARES8X, 'F-VALUE3XP-VALUE',/>
NN1=1
DO 140 I=2,M
   J=M-I+1
   IF(LER(J).NE.I) GO TO 140
   NN1=NN1+1
   B(499)=0.0D0
   IF(JR(NN1).GT.0) B(499)=(B(NN1)/DBLE(JR(NN1)))/B(500)
   CALL PF(JR(NN1),IDOF3,B(499),PVAL,IDOF2)
   CALL LABEL(I,IBLANK,LE,N,LIV,LP)
140 CONTINUE
145 FORMAT('+',11X,I6,2X,F20.7,5X,F10.3,3X,F7.5)

C * Estimation of parameters. *
C
WRITE(*,150)
150 FORMAT(15X,'Do you want parameter estimate? (Y or N)'
N1=80
READ(*,160)(IA(I),I=1,N1)
160 FORMAT(80A1)
   II=1
   IC=IGET(N1)
   IF(IC .NE. IY) GO TO 245
   DO 170 I=1,II
   B(I)=.0DO
   CALL CGM(1,S,W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)
   N1=LSTFI(J)+N1
   WRITE(*,175)
   WRITE(6,175)
175 FORMAT(///,25X,'PARAMETER ESTIMATES ***',//,
*8X,'PARAMETER',11X,'ESTIMATE')
DO 200 I=1,M
   J=M-I+1
170 CONTINUE
   IF(LER(J).NE.I) GO TO 200
   CALL LABEL(I,IBLANK,LE,N,LIV,LP)
   N1=LSTFI(J)+N1
   DO 190 NN1=1,LSTFI(J)
WRITE(*,205)NN1,W(N1-NN1)
190 WRITE(6,205)NN1,W(N1-NN1)
200 CONTINUE
205 FORMAT(11X,I3,F15.7)
245 CONTINUE
RETURN
300 WRITE(*,305)
305 FORMAT(5X,'Number of parameters is greater than 500')
   RETURN
END
Subroutine CGM performs the conjugate gradient algorithm based on the algorithms of Chapter 6.

**SUBROUTINE CGM(KZ,S,W,M,LSTFI,LER,N,LIV,LLIM,LT,LP,B)**

**DIMENSION W(1),LSTFI(1),LER(1),LIV(1),LLIM(1),LP(1)**

**DIMENSION LT(1),R(300),Z(300),B(1),ZL(500)**

**DOUBLE PRECISION W,S,R,Z,C,DK,DK1,T,B,ZL**

**COMMON /APAR/ NCELLS,LD1,LD2,LV,LB,LA**

**COMMON /BPAR/IIP,IDF,NOBS,ISA**

S=0.0D0

K=1

**Initialization.**

DO 10 I=1,NCELLS

IV=LV+I
IA=LA+I
R(I)=0.0D0
W(IV)=W(I)
10 W(IA)=W(IV)

DO 30 I=1,IIP

30 ZL(I)=0.0D0

45 IA=LA
IB=IA

CALL DECOMP(0,IA,W,M,LSTFI,N,LT,LIV,LLIM,LP)
IFLAG=0
DO 60 I=1,M
IF(LER(I) .GT. 0) GO TO 60
IF(I .EQ. 1) GO TO 50
NO=M-I+1
CALL LABEL(NO,0,LLIM,N,LIV,LP)
CALL POOL(IFLAG,IA,IB,W,N,LLIM,LT,LP)
50 IFLAG=1
60 IB=IB+LSTFI(I)

DK=0.0D0
DO 70 I=1,NCELLS

IV=LV+I
IA=LA+I
IF(IFLAG .EQ. 1) THEN
W(IA)=W(IV)-W(IA)
ELSE
W(IA)=W(IV)
ENDIF

70 DK=DK+W(IA)-W(IA)

IF(KZ .EQ. 4) RETURN
IF(K .EQ. 1) GO TO 80
C = DK/DK1
GO TO 90
80 C = 0.0D0
90 T = 0.0D0
DO 100 I = 1, NCELLS
   IA = LA + I
   ID1 = LD1 + I
   R(I) = W(IA) + C * R(I)
   Z(I) = W(ID1) * R(I)
100 T = T + R(I) * Z(I)
   DK1 = DK
   DK = DK1 / T
DO 110 I = 1, NCELLS
   IV = LV + I
   IA = LA + I
   Z(I) = W(IV) - DK * Z(I)
110 W(IA) = W(IV)
   T = DK1 / DK
C
* For sums of squares for model, residual, *
* type I, and type III, skip calculating the *
* E operator. *
C
IF(KZ .EQ. 0) GO TO 125
CALL EOPER(W, M, LSTFI, LER, N, LIV, LLIM, LP, LT)
DO 113 I = 1, IIP
   IA = LA + I
   IF(W(IA) .GE. 1.0D-8) GO TO 115
113 CONTINUE
GO TO 150
115 DO 120 I = 1, IIP
   IA = LA + I
   ZL(I) = W(IA) + C * ZL(I)
120 B(I) = B(I) + DK * ZL(I)
C
* For next iteration, save the data to the *
* original vectors. *
C
125 DO 130 I = 1, NCELLS
   IV = LV + I
   IA = LA + I
   W(IV) = Z(I)
130 W(IA) = W(IV)
   IF(K .NE. 1) GO TO 140
   IF(ISA .EQ. 1) GO TO 150
140 IF(T .LT. 1.0D-8) GO TO 170
   IF(K .GE. IDF) GO TO 150
   S = S + T
   K = K + 1
GO TO 45
150  S=S+T
    IF(KZ .EQ.0) GO TO 170
    DO 160 I=1,IIP
      IA=LA+I
      K=IIP-I+1
  160    W(IA)=B(K)
    170  RETURN
    END
Subroutine EOPER performs E operator for subroutine CGM by pooling the appropriate arrays of the decomposition.

SUBROUTINE EOPER(W,M,LSTFI,LER,N,LIV,LLIM,LP,LT)
DIMENSION W(1),LSTFI(1),LER(1),LIV(1),LLIM(1),LP(1)
DIMENSION LT(1)
DOUBLE PRECISION W
COMMON /APAR/ NCELLS,LD1,LD2,LV,LB,LA
NST=LA
CALL DECOMPM0,LA,W,H,LSTFI,N,LT,LIV,LLIM,LP
IA=LA
DO 60 I=1,M
IF(LER(I) .LE. 0) GO TO 60
NS=IA
NOP=M-I+1
CALL LABEL(NOP,0,LLIM,N,LIV,LP)
DO 20 K=1,N
IF(LP(K) .EQ. 0) LLIM(K)=-LLIM(K)
CONTINUE
DO 30 J=I,M
IF(LER(J) .EQ. I) GO TO 30
IF(LER(J) .NE. LER(I)) GO TO 30
LER(J)=-LER(I)
NOS=M-J+1
CALL LABEL(NOS,0,LLIM,N,LIV,LP)
CALL POOL(1,IA,NOS,W,N,LLIM,LT,LP)
NS=NS+LSTFI(J)
DO 40 K=1,N
LLIM(K)=IABS(LLIM(K))
MST=LSTFI(I)
DO 50 K=1,MST
NZ=K+IA
NST=NST+1
50 W(NST)=W(NZ)
IA=IA+LSTFI(I)
RETURN
END
SUBROUTINE DECOMP (IND, LOCA, W, M, LSTFI, N, LE, LIV, LLIM, LP)

C Note: the argument LE, LIV, and LP are used for mean.

DOUBLE PRECISION W, TEMP, DNPM, CMEAN
DIMENSION W(1), LSTFI(1), LIV(1), LLIM(1), LP(1), LE(1)
DATA IDOT/.'./
LI=1
MM=1
NN=1
LOCTWO=LOCA+1
10 LOCON=LOCA+1
KK=LL

C * Find number of elements in this mean.
   K1=NN
   NPM=LLIM(K1)
   DNPM=DNPM
20 LOCTWO=LOCTWO+LSTFI(MM)
   MEANST=LSTFI(MM+1)
   K1=M+1-KK
   INC=LSTFI(K1)
   MD=1
   NO=M-MM
   IF (IND .EQ. 2) CALL LABEL(NO, IDOT, LE, N, LIV, LP)
   DO 70 I=1, MEANST, INC
      JTWO=I+INC-1
      DO 60 J=I, JTWO
         L=MD
         LD=MD
         I1=LOCTWO+J-1
         TEMP=0.0D0
         DO 30 K=1, NPM
I2=LOCONEL+1
TEMP=TEMP+W(I2)

30 L=L+INC

C
C \* Deviate (IND=0); sums (IND=1); *
C \* classification means (IND=2). \*
C

IF(IND .EQ. 0) GO TO 40
IF(IND .EQ. 1) GO TO 36
IF(TMP .EQ. 0.0D0) THEN
   WRITE(*,32) J
   WRITE(6,32) J
ELSE
   CMEN=W(I1)/TEMP
   WRITE(*,34) J,W(I1),TEMP,CMEN
   WRITE(6,34) J,W(I1),TEMP,CMEN
ENDIF

32 FORMAT(1X,I6,5X,'(MISSING CLASSIFICATION CELL)')
34 FORMAT(1X,I6,1X,E16.8,F6.0,1X,E16.8)
36 W(I1)=TEMP
GO TO 60
40 W(I1)=TEMP/DNPM

C
C \* Form deviates. \*
C

DO 50 K=1,NPM
   I2=LOCONEL+LD-1
   W(I2)=W(I2)-V(I1)
50 LD=LD+INC
60 MD=MD+1
70 MD=MD+1
   IF(KK .EQ. 1) GO TO 80
   KK=KK-1
   MM=MM+1
   K1=LL-KK
   LOCONEL=LOCONEL+LSTFI(K1)
   GO TO 20
80 IF(NN .EQ. N) RETURN
   LL=LL+LL
   NN=NN+1
   MM=MM+1
   GO TO 10
END
Subroutine POOL either moves the secondary array into the primary array by addition.

SUBROUTINE POOL(IND,NP,NS,W,N,LLIM,LT,LP)
DIMENSION W(1),LLIM(1),LT(1),LP(1)
DOUBLE PRECISION W,TEMP

* NP = location of primary array; *
* NS = location of secondary array; *
* map coefficients obtained from LP; *
* replace (IND=0); and (IND .NE. 0) *

LOC1=NP
I=N
10 DO 20 J=1,I
20 LT(J)=1
30 LOC1=LOC1+1
LOC2=NS+1
DO 40 J=1,N
40 LOC2=LOC2+(LT(J)-1)*LP(J)
TEMP=W(LOC2)
IF(IND .NE. 0) TEMP=TEMP+W(LOC1)
W(LOC1)=TEMP
DO 50 K=1,N
IF(LLIM(K) .LT. 0) GO TO 50
IF(LT(K) .EQ. LLIM(K)) GO TO 50
LT(K)=LT(K)+1
IF(K .EQ. 1) GO TO 30
I=K-1
GO TO 10
50 CONTINUE
RETURN
END
Subroutine LABEL calculates the array of coefficients for the array map needed in pooling and produces output labels for classification means and for the soutions to the normal equations.

```fortran
SUBROUTINE LABEL(NO,ICHAR,LIST,N,LIV,LOA)
  DIMENSION LIST(1),LIV(1),LOA(1),LLL(10),INT(9)
  CHARACTER*2 LLL
  DATA IBLANK/' '/,INT/'I'N','T','E','R','C','E','P','T'/

  NUM=NO-1
  DO 10 I=N,1,10
    LOA(I)=IBLANK
  DO 20 I=1,N
    LOA(I)=ICHAR
  IF(NUM .EQ. 0) GO TO 70
    I1=-1
    J1=-1
  30 I1=I1+1
    I=N-I1
  40 J1=J1+1
    J=N-J1
  NUM=NUM-LIV(J)
    IF(NUM .GE. 0) GO TO 60
    NUM=NUM+LIV(J)
    IF(ICHAR .NE. IBLANK) GO TO 30
    GO TO 40
  60 LOA(I)=LIST(J)
    IF(NUM .NE. 0) GO TO 30
  70 IF(ICHAR .EQ. 0) GO TO 90
    DO 71 K=1,N
      IF(LOA(K) .NE. IBLANK) GO TO 73
    CONTINUE
  71 CONTINUE
    DO 72 K=1,9
      LLL(K)=INT(K)
    CONTINUE
    CONTINUE
  72 LLL(K)=LOA(K)
    WRITE(*,80)(LLL(K),K=1,10)
    WRITE(6,80)(LLL(K),K=1,10)
  80 FORMAT(1X,10A1)
  RETURN
```

Map coefficients: (NO=2\*N-I+1, ICHAR=0, LIST=LLIM)

Labels: Model term (NO=LER(I), ICHAR=, LIST=LE)

Subscripts (NO=2\*N-I+1, ICHAR=., LIST=LS)
DO 110 I=N,1,-1
  IF(LOA(I) .EQ. 0) GO TO 110
  LOA(I)=1
  DO 100 J=I,1,-1
    IF(LOA(J) .EQ. 0) GO TO 100
    LOA(I)=IABS(LOA(I)*LOA(J))
  100 CONTINUE
  CONTINUE
RETURN
END
C Subroutine PF computes F probability using method suggested *  
C by Davis and Khalil(1972) to evaluate F-distributions  *  
C**************************************************************************

SUBROUTINE PF(N1,N2,Z,P0FF,IER)
IMPLICIT REAL*8(A-H,O-Z)
IER=0
IF(Z .GT. 0.0DO)GO TO 5
P0FF=0.0DO
IER=1
RETURN
5 IF(N1 .GT. 0 .AND. N2 .GT. 0) GO TO 10
IER=2
P0FF=0.0DO
RETURN
10 AN1=DBLE(N1)
AN2=DBLE(N2)
A=AN1*Z/(AN1*Z+AN2)
A1=1.0D0-A
IF(A1 .LT. 0.1D-36)A1=.1D-36
D1=AN1*0.5DO
D2=AN2*0.5DO
D3=D1+D2-1.0D0
R=0.0DO
S1=0.0DO
S2=0.0DO
DEL=1.0DO
XM=1.0DO
XK=1.0DO
C=.25D0
PI=3.141592653589793D0
N=M2
15 M=IDINT(D2)
M=2*M
IF(M .NE. N) GO TO 30
N=IDINT(D2)-1
IF(N .EQ. 0) GO TO 25
DO 20 I=1,N
S1=DEL+S1*R
D2=D2-1.0D0
D3=D3-1.0D0
TEM=A1/D2
R=D3*TEM
S2=(R+TEM)*S2
20 CONTINUE
25 S1=DEL+S1*R
DEL=0.0DO
T=-1.0DO
D3=-1.0DO
S2=A*S2
**************************************************************************
C=C+0.5D0
GO TO 45
30  N=IDINT(D2)
   IF(N .EQ. 0) GO TO 40
   DO 35  I=1,N
      S1=DEL+S1*R
      D2=D2-1.0D0
      D3=D3-1.0D0
      TEM=A1/D2
      R=D3*TEM
      S2=(R+TEM)*S2
   35 CONTINUE
40  S1=XK*S1
    S2=XK*S2
    TEM=DSQRT(A1)
    XM=XK*TEM
    T=(XM-TEM)/A1
    D3=-.5D0
    XK=2.0D0/PI
    C=C^2.0D0
45  IF(C .GT. .875D0)GO TO 50
    D2=D1
    D3=D2+D3
    S2=S1
    S1=0.0D0
    A1=A
    IF(A1 .LT. .1D-36)A1=.1D-36
    N=N1
    GO TO 15
50  IF(C .LT. 1.125D0)DEL=4.0D0/PI*DATAN(T)
    POFF=1.0D0-XM*(S2-S1)+DEL
    IF(POFF .GE. 0.0D0 .AND. POFF .LE. 1.0D0)RETURN
    IF(POFF .LT. 0.0D0)POFF=0.0001D0
    IF(POFF .GT. 1.0D0)POFF=.99999D0
    IER=3
    RETURN
END