CERN-DATA HANDLING DIVISION
DD 75-23 Revision 1
R. Brun
M. Hansroul
J. Kubler
P. Palazzi
H. Wind*)
June 1977

MULTI-DIMENSIONAL FIT PROGRAM (MUDIFI)

*) CERN, NP Division.
1. INTRODUCTION

1.1 Motivation

A common problem encountered in different fields of applied science is to find an expression for one physical quantity in terms of several others which are directly measurable. An example in high-energy physics is the evaluation of the momentum of a charged particle from the observation of its trajectory in a magnetic field. The problem is to relate the momentum of the particle to the observations, which consist of position measurements at intervals along the particle trajectory. The exact functional relationship between these quantities is in general not known, but one possible way of solving the problem is to find an expression which reliably approximates the dependence of the momentum on the observations. This explicit function of the observations can be obtained by a least squares fitting procedure applied to a representative sample of data for which both the dependent quantity (momentum) and the independent observations are known. The function can then be used to compute the quantity of interest for new observations of the independent variables.

This report explains the usage of the program MUDIFI, which performs the least squares fitting using an extension of the original procedure from Ref. 1 which is more adapted to CERN's needs than the version described in Ref. 2. The program makes use of the automatic extension of memory space available on the CDC 7600 computer, thus freeing the user from the burden of adjusting various dimensions of arrays to the number of data points to be fitted. The histogramming package HBOOK (Ref. 3) is also used by MUDIFI for the presentation of results *).

1.2 Definition of the problem

Let D be the quantity of interest which depends smoothly on the observable quantities \( x_1, \ldots, x_k \), which we denote by \( \hat{x} \) in the following. Given a sample of data providing the values \( D(i) \) of the quantity D at

*) The source code of MUDIFI is available in the pool of "Physics Data Handling Algorithms" under the reference S6101.
several points \( \vec{x}(i) \) in the \( \ell \)-dimensional space, the program performs a least squares fit of the \( D(i) \) quantities by an expression of the form

\[
D = \sum_{n=1}^{N} c_n f_n(\vec{x}),
\]

where

\[
\begin{cases}
  c_n & \text{are constant coefficients} \\
  f_n(\vec{x}) & \text{are polynomial functions of the } \ell \text{ variables } \vec{x} \\
  N & \text{is the number of terms in the expression.}
\end{cases}
\]

The coefficients \( c_n \), the number \( N \), and the degrees of polynomials \( f_n \) are determined by the fitting procedure. However, the user has the option of imposing a given expression, i.e. the number \( N \) and the degrees of polynomials \( f_n \) (see data card POL), in which case only the coefficients \( c_n \) are computed by the program. The functions \( f_n \) are optionally chosen by the user to be products of Monomials, Legendre- or Chebyshev-polynomials in each of the independent variables, e.g.:

\[
f_n(\vec{x}) = T_{i(n)}(x_1) \ldots T_{m(n)}(x_\ell),
\]

where \( T_{i(n)}(x_1) \) is a Chebyshev polynomial of order \( i(n) \) in the variable \( x_1 \), etc. The functions \( f_n \) are thus defined by the indices \( i(n), \ldots, m(n) \).

When taking all combinations of polynomial functions we readily end up with an impossibly high number of terms. We therefore have to carefully select the functions that we actually include in the model. The selection procedure followed in the program is explained in the Appendix.
2. **STRUCTURE OF THE PROGRAM**

![Diagram showing the structure of the program](image)

3. **USAGE**

In order to facilitate the use of the program, a set of data cards must be provided by the user for the definition and selection of various options. In addition, the user has to write a main program and one small subprogram whose action is to read in one element from the sample (see 3.1.B). A third subroutine may also be optionally provided by the user in case he wishes to control the setting-up of the table of functions which are to be tried in the fitting procedure.
3.1 User main program and subroutines

3.1.1 Compulsory

A Main Program

The purpose of the main program is just to define the logical files which will be needed and to call the control subroutine of MUDIFI.

Example:

PROGRAM MAIN (INPUT, OUTPUT, TAPE4, TAPE5, TAPE6)

C TAPE4 = logical file reserved by MUDIFI for the results.
C
C TAPE5 = logical file from which the sample of data is to be read in order to perform the fit.
C
C TAPE6 = logical file from which a different sample of data is to be read in order to test the expression obtained by the fit on this new set of data.
C
C REMARK: this second test sample of data can also be read from logical file TAPE5 if the user has stored it on this same file. The control of this operation is done via data cards. If this is the case, TAPE6 does not have to appear in the program card.
C
C Call Control program of the fitting procedure.
C
CALL MUDIFI
C
STOP
C
END

B Subroutine GETONE

The task of this subroutine is to provide one element of the training sample to the fitting program. An element consists of one value D of the dependent variable and the vector \( \mathbf{x} \) of its corresponding independent variables. This subroutine is called as many times as needed until the total number of elements requested via data cards has been reached or until an "End of File" has been encountered on the logical file from which the sample
is read. The call to this subroutine by the fitting program has the form

CALL GETONE (D, X, NVAR, EOFDAT, NTAPE)

where the meaning of the parameters is explained in the example below.

Example:

SUBROUTINE GETONE (D, X, NVAR, EOFDAT, NTAPE)

DIMENSION X(NVAR)

LOGICAL EOFDAT

C INPUT PARAMETERS: (set via data cards)
C
C NVAR = number of independent variables
C
C NTAPE = logical file number from which to read the events
C
C D = function value for the event read
C
C (X(I), I=1, NVAR) = Values of the NVAR independent variables for the
C event read
C
C EOFDAT = logical variable to be set .TRUE. when an END OF FILE has
C been sensed on the logical file from which the sample is
C read.
C
C The following array BUFFER is the storage space reserved by the user
C for one logical record read from logical tape number NTAPE
C
C DIMENSION BUFFER (50)
C
C LOGICAL XEOF
C
C This logical function is a CERN Library
C function allowing the test for an END OF FILE.
C
C COMMON/LISVAR/LIST(10)
C This common may be optionally used for convenience in filling the
C vector X from the input storage BUFFER.
C
C The array LIST can be set by data cards by the user and contains the
C following information:
C
C (LIST(I), I = 1, NVAR) = pointer in array BUFFER where the variable
C X(I) is stored
C
C i.e. if LIST(1) = 10 then
X(1) = BUFFER(10)

XREAD is a CERN Library routine equivalent to a FORTRAN READ statement.

CALL XREAD (NTAPE, 1, BUFFER, 1, 50)

IF (XEOF(NTAPE)) G0TO 998

The function value is in BUFFER (5)

D = BUFFER (5)

D0 10 I = 1, NVAR

NV = LIST(I)

X(I) = BUFFER(NV)

10 CONTINUE

999 RETURN

998 EOFDAT = .TRUE.

G0TO 999

END

3.1.2 Optional subroutine

The user subroutine SELECT serves the purpose of selecting certain classes of functions among all possible functions of the independent variables.

A function is defined by the powers of the monomial or the order of the Legendre or Chebyshev polynomials in each of the independent variables.

If the dependence of the quantity D to be fitted is known to obey certain symmetries in some or all of the variables, it is possible to implement in the model those particular conditions by rejecting those functions which do not satisfy these symmetry rules.

The subroutine SELECT is called at the beginning of the program (once for each possible function) when it sets up the complete table of candidate functions to be considered in the fitting process. The call to this subroutine has the form
CALL SELECT (IGOOD)

where the parameter IGOOD has to be set equal to 1/0 if the function is to be retained/rejected.

Example:

```
SUBROUTINE SELECT (IGOOD)

COMMON/POWERS/NVAR, MP(10), MAXPOW(10), RI(10), IV(10), QM

C
C This common is needed for the transfer of the function definition
C as follows:
C
C NVAR = number of independent variables
C
C (IV(N), N=1, NVAR) contains the order +1 of the monomials, or the
C Legendre-, or Chebyshev- polynomials of the Nth variable.
C
C In this example, we suppose that the functions are products of mono-
C mials in each variable and that we want to reject the odd powers of
C the third variable.
C
IGOOD = 1
C Reject odd powers of third variable
C
C Remember that the IV's are the powers + 1.
C
IPOW = IV(3) - 1
IF (MOD(IPOW,2).NE.0) IGOOD=0
RETURN
END
```

Remark

The default action of subroutine SELECT is to consider all functions, i.e. IGOOD=1 in all cases. The program itself will, of course, make a rather severe selection from all candidate functions.

3.2 Data cards

The program uses the subroutine FREAD to read the data cards in free format.
The first four characters are used as a **KEYWORD** by the program in order to identify the group of variables set in the remaining part of the card.

The remaining pieces of information are values to be assigned to certain control variables used by the program.

The data cards are divided into a compulsory set and an optional one, for which default actions are taken if they are absent.

The end card must be read last. All other cards may be placed in any order.

**A Compulsory data cards**

All following cards to be written with the format (A4, FREE FORMAT)

<table>
<thead>
<tr>
<th>A4 KEYWORD</th>
<th>COMMENT</th>
<th>VARIABLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIT</td>
<td>TITLE</td>
<td>74 characters of comments : have to be enclosed by the dollar sign $ (see ex.)</td>
</tr>
<tr>
<td>DAT</td>
<td>1 (MDATA)</td>
<td>Number of data points to be read from the sample (≤ 4000). (Training sample).</td>
</tr>
<tr>
<td></td>
<td>2 (NVAR)</td>
<td>Number of independent variables (≤ 10).</td>
</tr>
<tr>
<td></td>
<td>3 (LSTUDY)</td>
<td>Maximum number of function considerations (≤ 3000)</td>
</tr>
<tr>
<td></td>
<td>4 (LFINAL)</td>
<td>Maximum number of terms allowed in final expression (≤ 60)</td>
</tr>
<tr>
<td>CON</td>
<td>CONTROL</td>
<td>Relative error tolerated.</td>
</tr>
<tr>
<td></td>
<td>1 (ERROR)</td>
<td>Minimum value of angle between a new candidate function and the subspace spanned by the previously accepted functions (in degrees between 0 and 90) (see Appendix).</td>
</tr>
<tr>
<td></td>
<td>2 (WOVERD)</td>
<td>Control parameter of function choice QM. By default is set to 1. (See Appendix).</td>
</tr>
<tr>
<td></td>
<td>3 (QM)</td>
<td></td>
</tr>
<tr>
<td>POW</td>
<td>POWERS</td>
<td>1</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAP</th>
<th>1</th>
<th>Logical file number from which to read the sample of data with which to perform the fit (training sample).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Logical file number from which to read the (test) sample of data on which to test the final expression obtained with the training sample. This logical file number can be the same as the first one, in which case the program assumes that the &quot;test&quot; sample immediately follows the &quot;training sample&quot;. A maximum number of 500 points are read from this tape.</td>
</tr>
</tbody>
</table>

| END | Terminates the data cards reading. (This card has to be the last one.) |

**Remark**

The logical file numbers set by the data card TAP must of course be defined in the PROGRAM card of the main program provided by the user (see 3.1.1 A). The same applies for the optional card OUT.
Optional data cards

Written with the same format as for the compulsory data cards.

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>COMMENTS</th>
<th>VARIABLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIS</td>
<td>LIST</td>
<td>Pointer to variable 1 in input buffer</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>ANG</td>
<td>ANGLE</td>
<td>Maximum value of the angle between the initial data vector to be fitted and the new candidate function to be included in the fit (in degrees). By default it is set to 0°, which automatically selects an alternative way of testing the construction of this function. (See Appendix)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>(ZERO)</td>
</tr>
<tr>
<td>HIS</td>
<td>HISTOGRAM</td>
<td>If set to 1, suppress all histograms.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>If set to 1, selects the NVAR 2-dimensional scatter plots of variable (i) versus residual, in addition to the 1-dimensional histograms of residuals for the training and test samples which are set by default.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>If set to 1, selects the 2-dimensional scatter plot of the function values versus the computed residuals</td>
</tr>
<tr>
<td>OUT</td>
<td>Logical file number of the result file. By default, it is the printer file OUTPUT. (See Remark on previous page.)</td>
<td></td>
</tr>
<tr>
<td>POL</td>
<td>This card allows the user to impose a given polynomial expression. For this, he has to give the sequence of powers of the variables in each polynomial. For example, the expression of three variables ( x_1^2 + x_2 x_3 + x_1 x_2 x_3 + x_3^2 ) can be imposed by the following data card: POL 2,0,0,0,1,2,1,1,1,0,0,3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If a continuation card is necessary, it must follow the first card and the keyword POL must be omitted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Important remark : in the case of an imposed expression the variable LFINAL (see DAT data card) must be set to the number of terms in the polynomial expression; in the example above LFINAL must be set to 4.</td>
<td></td>
</tr>
<tr>
<td>COR</td>
<td>CENTRE</td>
<td>Allows the printing of the correlation matrix of (D, X1, ..., XNVAR). By default, it is not printed.</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CEN</td>
<td>CENTRE</td>
<td>Allows the user to transform original variables to &quot;central&quot; variables, i.e. ( X_{\text{NEW}} = X_{\text{OLD}} - X_{\text{AVERAGE}} )</td>
</tr>
<tr>
<td>NOR</td>
<td>NO RESIDUAL</td>
<td>The value per default is TRUE. Allows the user to suppress the test over the residual sample. (Otherwise the test is made.)</td>
</tr>
<tr>
<td>LEG</td>
<td>LEGENDRE</td>
<td>Legendre polynomials used for the fit.</td>
</tr>
<tr>
<td>CHE</td>
<td>CHEBYSHEV</td>
<td>Chebyshev polynomials used for the fit. By default single monomials are used.</td>
</tr>
</tbody>
</table>

**Remark:** The last five cards do not need anything in columns 5 to 80.
Example of data cards

Example 1:

TIT  $ NO HISTOGRAMS ASKED$
DAT  2000.  5.  1000.  30.
CON  0.01  10.  1.5
POW  7.  7.  5.  6.  6.
TAP  5.  6.
HIS  1=1
COR
END

Example 2:

TIT  $ FORCED POLYNOMIAL X1 + X2**2 + X1 * X2$
DAT  1000.  2.  100.  3.
CON  0.1  7.5  1.2
POL  1,0  0,2  1,1
TAP  5.  5.
OUT  2.
LIS  7.  12.  23.  4.
ANG  60.
CEN
NOR
HIS  2=1  3=1
LEG
END

Remark: the data card POL can also be written as: POL 1=1 4=2 5=1 6=1

3.3 Deck set-up (CERN CDC 7600 system)

JOB,T10.
ACCOUNT,NAME,DIV,123456.
FTN,B=USER,L,R.
FIND,LIB1,MUDIFIBINARY, ID=DE100.
FIND,LIB,7600LIBRARY,ID=PROGLIB.
FIND,LIB2,HBOOKS7600LIB,ID=PROGLIB.
LIBRARY,LIB1,LIB2,LIB.
FIND,TAPE5,DATASAMPLE,ID=USERID.
FIND,TAPE6,TESTSAMPLE,ID=USERID.
LOAD, USER.
EXECUTE.
DISPOSE, TAPE4, PU, ST=CSCP. (To punch out results)
End-of-Record

PROGRAM MAIN(INPUT, OUTPUT, TAPE4, TAPE5, TAPE6)
...
END

SUBROUTINE SELECT(IGOOD)
...
END

SUBROUTINE GETONE(D, X, NVAR, EOFDAT, NTAPE)
...
END

End-of-Record
Data cards
End-of-file.

4. REMARKS

4.1 General remarks

This program is not a universal fitting procedure. Not all functions of several variables can be fitted reasonably well. We repeat here a remark from Ref. 1 (page 87) to make, prior to the fit, an analytical estimate of D. The multi-dimensional fit in terms of \( \hat{x} \) can then be made only for the difference with the correct value. For example, if the magnetic field of a spectrometer is fairly homogeneous the trajectory will be almost a helix. We can then estimate the momentum by assuming that the trajectory is a helix and only make the multi-dimensional fit to the difference with the correct value. Since expressing a helix in a power series is rather inefficient, we can thus substantially economize on the length or precision of the final expression.

The use of the automatic extension of memory in SCM as well as in LCM is particular to the CDC 7600 computer. In case one wants to run the program on another computer, a small number of modifications are necessary. These are available from the authors.
4.2 Additional comments for the output interpretation

a) Even if a good fit has been obtained for the training sample, it is not certain that the results of this fit applied to a different sample will give correct results. The reasons for this are mainly:

- the original training sample was not sufficiently representative of the full variety of possible events, and/or

- the original variables were not sufficiently "independent", or non-correlated. In this latter case, a better set of variables can be chosen, by performing a principal components analysis of the sample and applying the fitting procedure using the principal components obtained as the independent variables (Ref. 1.)

b) Three criteria are available in order to estimate the goodness of the fit:

- the multiple correlation coefficient: its value must be close to 1 for a good fit;

- the variance of the coefficients of the final expansion: this gives an estimate of the precision with which the coefficients of the final expression are known;

- confidence interval for each coefficient [Cmin, Cmax]: the "true" value of the coefficient C in the expansion lies in this interval with an estimated probability of 90%. If the value "0" is contained in this interval, the contribution to the fit of the polynomial corresponding to this coefficient is not meaningful.

c) In some cases it is useful to examine the correlation matrix of the original variables optionally printed by the card COR. It has the form of a symmetric matrix:

\[
\begin{pmatrix}
X_1 & X_2 & \ldots & X_L \\
D & a_{11} & a_{12} & & a_{1L} \\
X_1 & a_{21} & & & \\
X_2 & a_{31} & a_{32} & & \\
& \ddots & \ddots & \ddots & \ddots \\
X_L & a_{L1} & a_{L2} & \ldots & a_{LL}
\end{pmatrix}
\]
d) The logical file number "4" is used by the program to write a full function subroutine giving the FORTRAN code expressing the found algebraic expression of the final result of the fit.

e) For a fit to be meaningful, the number of data points must be sufficiently large, for instance not less than NCOEFF*50, where NCOEFF is the number of terms in the final expression. If the model does not fit the training sample, we have an "underfit", i.e. not enough coefficients. If, however, it fits the training sample but not a fresh sample, we may have an "overfit", i.e. too many coefficients.

f) The program normalizes the original variables between -1 and +1 after having determined their domain of variation for the sample studied. It is important to include in the training sample points whose values of the variables lie near the boundaries of their respective domain of variation expected in the real application, to avoid extrapolation. If the user has requested via data card to center the original variables about their sample average, the final expansion obtained after the fit is expressed in terms of these new centered variables.
REFERENCES


APPENDIX A.1

THE GRAM-SCHMIDT TRANSFORMATION

We consider an $M \times N$ matrix $\hat{F}$, an element of which is given by

$$\hat{f}_{ij} = F_j(\xi_{1i}, \xi_{2i}, \ldots, \xi_{Li})$$

where $i$ labels the $M$ observations and $j$ labels $N$ functions of $L$ variables ($N \leq M$).

We now want to determine coefficients $\hat{c}$ such that $(\hat{D} - \hat{F} \hat{c})^2$ is a minimum, where $D_i$ are the known function values, say the momentum, at the point $(\xi_{1i}, \xi_{2i}, \ldots, \xi_{Li})$.

We make use of the Gram-Schmidt orthogonalizing process. By this is meant that we define a matrix $\hat{W}$ of which the columns $\hat{w}_j$ are given by:

$$\hat{w}_i = \hat{f}_1 \quad (= F_1(\xi_{1i}, \xi_{2i}, \ldots, \xi_{Li})) \quad \text{with} \ i = 1, 2, \ldots, M$$

$$\hat{w}_j = \hat{f}_j - \sum_{k=1}^{j-1} (\hat{f}_j \cdot \hat{w}_k) \hat{w}_k / \hat{w}_k^2.$$  \hspace{2cm} (A1)

Then $\hat{w}_i \cdot \hat{w}_j = 0$ if $i \neq j$ (see further).

We now take as a new model

$$\hat{\hat{W}} \hat{a}.$$  \hspace{2cm} - 17 -

We thus want to minimize

$$S \equiv (\hat{D} - \hat{\hat{W}} \hat{a})^2.$$  \hspace{2cm} (A2)

Differentiation with respect to $a_j$ gives (since $\hat{w}_j \cdot \hat{w}_i = 0$ if $i \neq j$)

$$\hat{D} \cdot \hat{w}_j - a_j \hat{w}_j^2 = 0$$

or

$$a_j = (\hat{D} \cdot \hat{w}_j) / \hat{w}_j^2.$$  \hspace{2cm} (A2)

To derive now $\hat{c}$, we first note that formula (A1) can be written as

$$\hat{F} \hat{W} = \hat{D} \hat{a}.$$
where

\[
\begin{align*}
    b_{ij} &= \left(\hat{f}_j \cdot \hat{w}_i\right)/\hat{w}_i^2 & \text{if } i < j \\
    b_{ij} &= 1 & \text{if } i = j \\
    b_{ij} &= 0 & \text{if } i > j.
\end{align*}
\]

Consequently, \( \hat{\mathbf{B}} \) is an upper triangular matrix, which can readily be inverted.

We now evaluate

\[
\hat{\mathbf{F}} \hat{\mathbf{B}}^{-1} \hat{\mathbf{F}}^T = \hat{\mathbf{W}}.
\]

The model \( \hat{\mathbf{W}} \) can therefore be written as \( (\hat{\mathbf{F}} \hat{\mathbf{B}}^{-1}) \hat{\mathbf{a}} = \hat{\mathbf{F}}(\hat{\mathbf{B}}^{-1}) \hat{\mathbf{a}} \).

The original model \( \hat{\mathbf{F}} \hat{\mathbf{c}} \) is therefore identical with this if

\[
\hat{\mathbf{c}} = (\hat{\mathbf{B}}^{-1}) \hat{\mathbf{a}} = \left[\hat{\mathbf{a}}^T (\hat{\mathbf{B}}^{-1}) \hat{\mathbf{a}} \right]^T.
\]

The reason why we use \( (\hat{\mathbf{B}}^{-1})^T \) rather than \( \hat{\mathbf{B}}^{-1} \) is to save storage, since \( (\hat{\mathbf{B}}^{-1})^T \) can be stored in the same matrix as \( \hat{\mathbf{B}} \).

The more coefficients we evaluate, the more the sum of squares of residuals \( S \) will be reduced. We can evaluate \( S \) before inverting \( \hat{\mathbf{B}} \).

Let \( S_j \) be the sum of squares of residuals when taking \( j \) functions into account. Then

\[
S_j = \left(\hat{\mathbf{D}} - \sum_{i=1}^{j} a_{i} \hat{w}_{i} \right)^2 = \hat{\mathbf{D}}^2 - 2 \hat{\mathbf{D}} \sum_{i=1}^{j} a_{i} \hat{w}_{i} + \sum_{i=1}^{j} a_{i}^2 \hat{w}_{i}^2.
\]

We derived [see formula (A2)]:

\[
(\hat{\mathbf{D}} \cdot \hat{\mathbf{w}}) = a_{i} \hat{w}_{i}^2,
\]

so that

\[
S_j = \hat{\mathbf{D}}^2 - 2 \sum_{i=1}^{j} a_{i}^2 \hat{w}_{i}^2 + \sum_{i=1}^{j} a_{i}^2 \hat{w}_{i} = \hat{\mathbf{D}}^2 - \sum_{i=1}^{j} a_{i}^2 \hat{w}_{i}^2.
\]

So for each new function \( f_j \) included in the model we get a reduction of the sums of squares of residuals of \( a_{j}^2 \hat{w}_{j} \) [where \( \hat{w}_{j} \) is given by formula (A1) and \( a_{j} \) by (A2)]. This way we can decide if we want to include this function in the final model before the matrix inversion.
FUNCTIONS SELECTION

Supposing that (N-1) steps of the procedure have been performed, the problem now is to consider the N\textsuperscript{th} function.

The sum of squares of residuals can be written as

\[ S_N = \mathbf{D}^t \cdot \mathbf{D} - \sum_{n=1}^{N} a_n^2 (\mathbf{w}_n^t \cdot \mathbf{w}_n), \]  \hspace{1cm} (A3)

where the relation (A2) has been taken into account. The contribution of the N\textsuperscript{th} function to the reduction of S is then given by

\[ \Delta S_N = a_N^2 (\mathbf{w}_N^t \cdot \mathbf{w}_N). \]  \hspace{1cm} (A4)

Two tests are now applied in order to decide whether this N\textsuperscript{th} function is to be included in the final expression or not.

Test 1

Denoting by \( H_{N-1} \) the subspace spanned by \( \mathbf{w}_1, \ldots, \mathbf{w}_{N-1} \) the function \( \mathbf{w}_N \) is by construction [see (A1)] the projection of the function \( \mathbf{F}_N \) on the direction orthogonal to \( H_{N-1} \). Now, if the length of \( \mathbf{w}_N \) (given by \( \mathbf{w}_N^t \cdot \mathbf{w}_N \)) is very small compared to the length of \( \mathbf{F}_N \), this new function cannot contribute much to the reduction of the sum of squares of residuals. The test consists then in calculating the angle \( \text{THETA} \) between the two vectors \( \mathbf{w}_N \) and \( \mathbf{F}_N \) (see figure) and requiring that it is greater than a threshold value which the user must set via the data card CONTROL, variable WOVERD.
Test 2

Let \( \hat{D} \) be the data vector to be fitted. As illustrated in the figure below, the \( N \)th function \( \hat{W}_N \) will contribute significantly to the reduction of \( S \), if the angle PHI between the vectors \( \hat{W}_N \) and \( \hat{D} \) is smaller than an upper limit which the user can define via the data card ANG:

\[
\hat{W}_N \quad \text{PHI} \quad \hat{D}
\]

\[ H_{N-1} \]

In the absence of an ANG data card, an alternative way of performing this second test is used by default in the program: the \( N \)th function \( \hat{z}_N \) is accepted if \( \Delta S_N > \frac{S_{N-1}}{M-N} \)

where

\( S_{N-1} = \text{sum of squares of residuals after (N - 1) functions have been accepted}; \)

\( M = \text{total number of functions allowed in the final expression of the fit (M = LFINAL set in data card DAT).} \)

From this we see that by restricting \( M \), the number of terms in the final model, the fit is more difficult to make since the above selection criterion is more stringent.

The fitting procedure is pursued step by step, examining one function at a time by taking the previously accepted functions into account in the manner described above. The process is stopped when the sum of squares of residuals have been reduced below an error level acceptable to the user (see data card CONTROL).

It remains to be shown how one can build up the initial table of candidate functions from which the program can select for the final expression.
APPENDIX A.3

CANDIDATE FUNCTIONS TABLE

Let $PM(I)$ be the maximum power allowed for the $I^{th}$ variable. The candidate functions $f(x)$ are products of monomials or Legendre- or Chebyshev-polynomials in each of the variables. They are built up in such a way that the following relation is satisfied:

$$\frac{P(1)}{PM(1)} + \frac{P(2)}{PM(2)} + \ldots + \frac{P(NV)}{PM(NV)} \leq QM,$$

where $P(I)$ is the power of the $I^{th}$ variable in the function and $QM$ is a control parameter set by the user (see data card CON).

A table of all functions satisfying the above relation is constructed, the total number of them becoming larger as the parameter $QM$ increases. However, the maximum number of candidate functions is limited by the program to 1000 because of the DIMENSION statements.
SUMMARY OF THE INPUT/OUTPUT OF MUDIFI

- Training sample $(D, \mathbf{x})_i$
- Test sample $(D, \mathbf{x})_j$
- Data cards

MUDIFI

PRINTOUT
Input data
fit progress
report
Results:
Number of terms
Coefficient values
Functions definitions

HISTOGRAMS
Residuals
on training
and test
samples

PUNCHED CARDS
Fortran code
of function

$$D = \sum_{n=1}^{N} c_n F_n(\mathbf{x})$$