A NUMERICAL ALGORITHM FOR THE CALCULATION
OF COEFFICIENTS IN MOVING AVERAGES

by

Bengt Lindberg*
## CONTENTS

1. BACKGROUND AND NOTATION FOR THE NUMERICAL ALGORITHM
   1.0 General
   1.1 Optimization problem
   1.2 Numbering of the unknowns
   1.3 Data structure for the matrix Q
   1.4 Constraints
   1.5 Data structure for the constraints
   1.6 Operations involving V

2. NUMERICAL SOLUTION
   2.1 Basic problem
      2.1.1 Algorithm
   2.2 Modified problem, some values of $y$ given
      2.2.1 Algorithm

3. USER INSTRUCTION
   3.1 No coefficients specified
   3.2 The coefficients of the main moving average specified

4. INSTRUCTIONS FOR INSTALLATION AND MODIFICATION

5. PROGRAM ORGANIZATION AND LISTING
   5.1 The main program
   5.2 The subroutine SYSTEM
   5.3 The subroutine CONSTRAINT
   5.4 Input and output subroutines

APPENDIX 1. Construction of Q
APPENDIX 2. NAG library routines

ACKNOWLEDGEMENTS
REFERENCES
1.- BACKGROUND AND NOTATION FOR THE NUMERICAL ALGORITHM

1.0 General

Given a vector of values $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$.
For given integer $a$ construct coefficients $r_j$, $j = 1, 2, \ldots, 2a+1$, and
$\gamma_j^t$, $t = 1, 2, \ldots, a$, $j = 1, 2, \ldots, a+t$ in the averaging formulas

$$\lambda^* = \sum_{j=0}^{a} \gamma_j^t \lambda_j; \quad t = 1, 2, \ldots, a;$$
$$\lambda^* = \sum_{j=1}^{2a+1} r_j \lambda_{t+j-a}; \quad t = a+1, a+2, \ldots, m-a;$$
$$\lambda^* = \sum_{j=1}^{a+t} \gamma_j^t \lambda_{m+1-j}; \quad t = a, a-1, \ldots, 1.$$

The coefficients $r_j$ shall be symmetric with respect to $r_{a+1}$, i.e.,

$$r_{a+1+j} = r_{a+1-j}; \quad j = 1, 2, \ldots, a.$$ In Section 1.2 we will use
the notation $\gamma_j^{a+1} = r_j; \quad j = 1, 2, \ldots, 2a+1$. The coefficients shall be
optimal in a certain sense defined in Section 1.1 and satisfy certain
constraints defined in Section 1.4.

For a general statistical motivation of this numerical problem and
further analysis of the method of moving averages, see Hoem (1978) and
Linnemann (1979, 1980).

This paper deals solely with the practical computer implementation
of a problem suggested to the present author by Jan M. Hoem.
1.1 Optimization problem

Define for given integer $z > 0$

$$L = \sum_{t=1}^{m-z} (\Delta^z \lambda_t - \Delta^z \hat{\lambda}_t)^2,$$

where $\Delta$ is the forward difference

$$\Delta^z \lambda_t = \lambda_{t+1} - \lambda_t$$

and $\Delta^z$ denotes $z$ applications of $\Delta$.

The expression for $L$ is a quadratic function in the coefficients $r_j, \gamma_j$.

If we number the unknown coefficients in the way described in Section 1.2, we can write

$$L = y^T Q y$$

where the matrix $Q$ can be constructed in the sequence of steps described in Appendix 1.
1.2 Numbering of the unknowns

Arrange the unknown coefficients $Y^t_j$ in a table as follows.

<table>
<thead>
<tr>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$Y^1_1$</td>
<td>$Y^2_1$</td>
<td>$Y^3_1$</td>
<td>$Y^a_1$</td>
</tr>
<tr>
<td>2</td>
<td>$Y^1_2$</td>
<td>$Y^2_2$</td>
<td>$Y^3_2$</td>
<td>$Y^a_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>a+1</td>
<td>$Y^1_{a+1}$</td>
<td>$Y^2_{a+1}$</td>
<td>$Y^3_{a+1}$</td>
<td>$Y^a_{a+1}$</td>
</tr>
<tr>
<td>a+2</td>
<td>$Y^2_{a+2}$</td>
<td>$Y^3_{a+2}$</td>
<td>...</td>
<td>$Y^a_{a+2}$</td>
</tr>
<tr>
<td>a+3</td>
<td>$Y^3_{a+3}$</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>a+a</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>$Y^a_{a+a}$</td>
</tr>
</tbody>
</table>

Then define a vector $y$ containing the unknowns of this table in lexicographic order, i.e., numbered line by line according to

$$y_{\text{index}} = Y^t_j$$

with

$$\text{index} = \begin{cases} 
(j-1) a + t; & 1 \leq j \leq a+1; \quad 1 \leq t \leq a; \\
a^2 + a + t - 1; & j = a+2; \quad 2 \leq t \leq a; \\
a^2 + a + a - 1 + t - 2; & j = a+3; \quad 3 \leq t \leq a; \\
\vdots & \vdots \\
a^2 + \frac{1}{2}a(a-1); & j = a+a; \quad t = a. 
\end{cases}$$
For example for \( a = 3 \) we get the following indexing table.

\[
\begin{array}{c|ccc}
  \text{J} & 2 & 3 \\
  \hline
  2 & 4 & 5 & 6 \\
  3 & 7 & 8 & 9 \\
  4 & 10 & 11 & 12 \\
  5 & 13 & 14 & \text{---} \\
  6 & \text{---} & \text{---} & 15 \\
\end{array}
\]

The unknowns \( r_1, r_2, \ldots, r_{a+1} \) are adjoined according to

\[
y_{a+!(a-1)+j}^{2} = r_{j}; \quad j = 1, 2, \ldots, a+1.
\]

Denote by

\[
n = a^2 + ia(a-1) + a + 1
\]

the total number of unknowns.

1.3 Data structure for the matrix \( \mathbf{Q} \)

The matrix \( \mathbf{Q} \) is symmetric, positive definite and has the following structure of non-zero elements (cf. Appendix 1).
To suit the NAG-routine F01MCF we use the skyline or envelope storage scheme for this matrix. Only half the matrix is needed and only the elements from the first non-zero element of a row to the diagonal element of the same row need to be stored for each row. The data of the matrix is stored in a long vector together with an integer vector that points to the first non-zero element of each row. For further technical details, see the description of the parameters of F01MCF in the NAG library manual.

1.4 Constraints

All the formulas in Section 1.1 shall be exact for polynomials of degree less than \( d \) (a given integer), i.e., if

\[
\hat{\lambda}_j = P(x_j); \quad j = 1, 2, \ldots, m;
\]

with \( P \) a polynomial of degree \( d-1 \) and \( x_j = x_0 + j \cdot \Delta x, j=1,2,\ldots, m \) for some given \( \Delta x \), then

\[
\lambda^*_t = P(x^*_t)
\]

for all \( t = 1, 2, \ldots, m \).

These conditions can be expressed in several equivalent ways. They all amount to \( d \) equations for each of the sets of coefficients

\[
\{Y^*_1, Y^*_2, \ldots, Y^*_{a+1}; \quad t = 1, 2, \ldots, a+1.\}
\]

In all there are \((a+1) \times d\) equations. Due to the symmetry-condition for

\[
y^*_j; \quad j = 1, 2, \ldots, 2a+1; \text{ some of the equations for } \n^*_{a+1}; \quad j = 1, 2, \ldots, a+1
\]

are redundant, see below.

We have chosen to construct sets of orthogonal polynomials
\( \{ \phi_s^t(x) \}_{s=0}^{d-1} \) for each of the intervals \( I_t = [x_1, x_{a+t}] \) \( t = 1, 2, \ldots, a+1 \)
to use as bases for the space of polynomials of degree \( d-1 \) on each interval \( I_t \). The polynomials are orthogonal with respect to the inner product

\[
<f, g>_t = \sum_{i=1}^{a+t} f(x_i) g(x_i)
\]

This special choice of basis is motivated by a wish to get orthogonal rows for the matrix of constraints.

The constraints can be written

\[
\sum_{j=1}^{a+t} \sum_{s=0}^{d-1} \phi_s^t(x_j) \phi_s^t(x_i) = \phi_s^t(x_i); \quad s = 0, 1, \ldots, d-1; \\
\quad t = 1, 2, \ldots, a+1;
\]

i.e.,

\[
V_t \Phi_t = \phi_t; \quad t = 1, 2, \ldots, a+1;
\]

where

\[
\phi_t = \begin{bmatrix}
\phi_0^t(x_t) \\
\phi_1^t(x_t) \\
\vdots \\
\phi_d^t(x_t)
\end{bmatrix};
\]

\[
\Phi_t = \begin{bmatrix}
\phi_0^t(x_t) \\
\phi_1^t(x_t) \\
\vdots \\
\phi_d^t(x_t)
\end{bmatrix};
\]
These are \( a+1 \) uncoupled sets of equations.

For \( t = a+1 \) the symmetry condition reduces the equations to

\[
V_0 \Xi = \Phi^{a+1}
\]

with

\[
\begin{bmatrix}
\mathbf{r}_1 \\
\mathbf{r}_2 \\
\vdots \\
\mathbf{r}_{a+1}
\end{bmatrix}
\]

\[
V_0 = \begin{bmatrix}
\phi_0^{a+1}(x_1) + \phi_0^{a+1}(x_{2a+1}) & \phi_0^{a+1}(x_2) + \phi_0^{a+1}(x_{2a}) & \cdots & \phi_0^{a+1}(x_{a+1}) \\
\phi_1^{a+1}(x_1) + \phi_1^{a+1}(x_{2a+1}) & \phi_1^{a+1}(x_2) + \phi_1^{a+1}(x_{2a}) & \cdots & \phi_1^{a+1}(x_{a+1}) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_{d-1}^{a+1}(x_1) + \phi_{d-1}^{a+1}(x_{2a+1}) & \phi_{d-1}^{a+1}(x_2) + \phi_{d-1}^{a+1}(x_{2a}) & \cdots & \phi_{d-1}^{a+1}(x_{a+1})
\end{bmatrix}
\]

\( V_0 \) is obtained by folding \( V_{a+1} \) along column \( a+1 \) and adding columns.

Due to the fact that the points \( x_1, x_2, \ldots, x_{2a+1} \) are symmetric around \( x_{a+1} \), the rows containing odd numbered polynomials \( \phi_i^{a+1}, i=1,3, \ldots \) are identically zero, also these right hand sides are identically zero.

Thus, these equations read \( 0=0 \) and can be omitted.
The remaining equations are written

$$V^* = \mathbf{r}^*$$

with

$$\mathbf{r}^* = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{a+1} \end{bmatrix}$$

$$\mathbf{r}^* = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{a+1} \end{bmatrix}$$

$$\mathbf{*}^* = \begin{bmatrix} \phi_0^{a+1}(x_{a+1}) \\ \phi_2^{a+1}(x_{a+1}) \\ \vdots \\ \phi_{2p}^{a+1}(x_{a+1}) \end{bmatrix}$$

$$\mathbf{r}^* = \begin{bmatrix} \phi_0^{a+1}(x_1) + \phi_0^{a+1}(x_{2a+1}) & \phi_0^{a+1}(x_2) + \phi_0^{a+1}(x_{2a}) & \cdots & \phi_0^{a+1}(x_{a+1}) \\ \phi_2^{a+1}(x_1) + \phi_2^{a+1}(x_{2a+1}) & \phi_2^{a+1}(x_2) + \phi_2^{a+1}(x_{2a}) & \cdots & \phi_2^{a+1}(x_{a+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{2p}^{a+1}(x_1) + \phi_{2p}^{a+1}(x_{2a+1}) & \phi_{2p}^{a+1}(x_2) + \phi_{2p}^{a+1}(x_{2a}) & \cdots & \phi_{2p}^{a+1}(x_{a+1}) \end{bmatrix}$$

where $p$ is the integer part of $\frac{d-1}{2}$.

The number of constraint equations is

$$n_c = a \cdot d + p + 1.$$
All the equations can be summarized as

\[ V^*_m = V, \]

where \( V \) is an \( n \times n \) sparse matrix with special structure and

\[
V = \begin{bmatrix}
  1 \\
  \vdots \\
  \vdots \\
  \vdots \\
  m \times n \\
\end{bmatrix}
\]

1.5 Data structure for the constraints

We store the matrices \( V^* \) and \( V_t \), \( t = 1, 2, \ldots, a \), in a rectangular matrix \( VT \) in the following fashion. For our description denote the transpose of \( VT \) by \( U \). Then

\[
U = \begin{bmatrix}
  V_1 & V_2 & V_3 & \ldots & V_a & V^* \\
  \hline
  0 & & & & & \\
\end{bmatrix}
\]

Together with \( VT \) we store six integer vectors that carry information on the structure of the \( n \times n \) matrix \( V \) of constraints

\[ V^*_m = V. \]

For each of the \( n \) columns of \( U \) the components of the vector \( \text{COLIND} \) tell the index of the corresponding unknown in \( V \), i.e., the column number of the corresponding column in the matrix \( V \).
The vectors \( \text{LOWCOL} \) and \( \text{UPCOL} \) tell for each column of \( U \) the lower row-index and upper row-index for that column's location in \( V \).

For most of the columns \( \text{UPCOL}(j) - \text{LOWCOL}(j) = d \), but for the last \( a+1 \) columns of \( U \) the difference is smaller.

There are \( n_c \) rows in \( V \), for each row the vectors \( \text{INDV} \), \( \text{LOWV} \), \( \text{UPV} \) indicate the location in the matrix \( U \) of the coefficients for the actual row. \( \text{INDV} \) gives the row number, \( \text{LOWV} \) and \( \text{UPV} \) the lower column index and the upper column index respectively.
With the aid of COLIND we get the location in the matrix V.

INDV(p) takes values in the set \( \{1, \ldots, d\} \).
1.6 Operations involving V

In the algorithms of Section 2 we need to perform the following operations with V:

(i) Calculate a vector $V_z$.
(ii) Calculate a vector $V^T_w$.
(iii) Extract a column of $V^T$.

(i) The $n_c$ components of the vector $w = V_z$ are given by

$$w(p) = \sum_{j=LOWV(p)}^{UPV(p)} U(INDV(p), J) Z(COLIN(J));$$

for $p = 1, 2, \ldots, n_c$.

(ii) The $n$ components of the vector $z = V^T_w$ are

$$Z(COLIN(I)) = \sum_{J=LOWCOL(I)}^{UPCOL(I)} U(INDV(J), I) w(J);$$

for $I = 1, 2, \ldots, n$.

(iii) The $p$-th column of $V^T$ is the $p$-th row of $V$. Thus the non-zero elements of the column are given by

$$Z(COLIN(J)) = U(INDV(p), J);$$

for $J = LOW(p), LOWV(p) + 1, \ldots, UPV(p)$.

In the program we have used the matrix $VT = U^T$ instead of $U$ so whenever the matrix element $U(p,q)$ appears above we should replace it with $VT(q,p)$. 
2. NUMERICAL SOLUTION

2.1 Basic Problem

The problem
\[
\min \begin{bmatrix} y \end{bmatrix}^T Q y
\]
for
\[
V y = \begin{bmatrix} v \end{bmatrix},
\]
with \( Q \) an \( n \times n \) symmetric positive definite matrix, \( V \) an \( n_c \times n \) matrix, \( y \) an \( n \)-vector, and \( v \) an \( n_c \)-vector, \( n_c < n \), has the solution
\[
y = Q^{-1} v \begin{bmatrix} v \end{bmatrix}^T (Q^{-1} v)^{-1} y.
\]
See e.g. Gill et al. (1981, Section 5.4.1) and Section 2.2 of this paper.

2.1.1 Algorithm

The solution is computed in the following sequence of steps, using subroutines from the NAG-library.

1. Compute the Choleski factorization \( R^T R \) of \( Q \). Here \( R \) is an upper triangular \( n \times n \)-matrix. Due to the special structure of \( Q \) this is best done by the NAG-routine F01MCF.

2. Define \( Z = Q^{-1} v \). \( Z \) is an \( n \times n_c \)-matrix. The \( n_c \) columns of \( Z \) are the solutions of the \( n_c \) systems of linear equations
\[
QZ = v^T.
\]
These systems are solved using the Choleski factorization of \( Q \). This is done by the NAG-routine F04MCF.
3. Form the symmetric, positive definite matrix 
\[ H = VQ^{-1}V^T = Vz. \]

\( H \) is a \( n_c \times n_c \)-matrix. Only the upper triangular part of \( H \) need to be computed.

4. Define \( \tilde{w} = H^{-1}v \). The \( n_c \)-vector \( \tilde{w} \) is the solution of the symmetric "small" linear system

\[ \tilde{H} \tilde{w} = \tilde{v} \]

which is solved by the NAG-routine F04ASF.

5. Form \( y = \mathbf{v}_T \tilde{w} \). \( y \) is an \( n \)-vector.

6. Finally compute \( \mathbf{v} = Q^{-1}u \). This is equivalent to solving the system of linear equations

\[ Q\mathbf{v} = u, \]

which is done by F04MCF.

Note that this algorithm never explicitly computes the inverses of \( Q \) and \( H \). The inverses have no structure even if \( Q \) and \( H \) have, so they should be avoided. The Choleski-factors, however, have the same kind of structure as the original matrices. For numerical calculations it is always advisable to avoid explicit inverses unless you are interested in the individual elements of them. In our algorithm we are only interested in operating with the inverse on certain given vectors.

2.2 Modified problem, some values of \( y \) given

The problem

\[
\begin{aligned}
\min_{\mathbf{v}} & \mathbf{v}^TQ\mathbf{v} \\
\text{for} & \\
V\mathbf{v} = \mathbf{y}, \\
\end{aligned}
\]

\[ y_i = c_i, \quad i = n-a, n-a+1, \ldots, n, \]
can be solved by first enlarging the set of constraints $W\underline{x} = \underline{y}$ with the $k$ constraints $y_i = c_i$ to

$$W\underline{x} = \underline{y},$$

with $W$ an $(n_c + k) \times n$ matrix and $\underline{w}$ an $(n_c + k)$ vector. Then the problem is solved as in Section 2.1. This, however, may be grossly inefficient.

The following transformations give a much simpler problem.

The idea is to eliminate $y_i, i = n-a, \ldots, n$, from both the quadratic form and the constraints.

Introduce the following notation

$$\underline{X} = \begin{bmatrix} \underline{X}_A \\ \underline{X}_B \end{bmatrix} \quad \text{(n-a-1 components)}$$

$$\underline{Y} = \begin{bmatrix} \underline{Y}_A \\ \underline{Y}_B \end{bmatrix} \quad \text{(a+1 components)}.$$

Let $k = a+1$.

Partition $Q$ as follows (the same partition as in Appendix 1):

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21}^T & Q_{22} \end{bmatrix}$$

and $V$ as

$$V = [V_A \quad V_B].$$

The quadratic form is then

$$\underline{w}^TQ\underline{w} = (\underline{w}_A \underline{w}_B)^T \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21}^T & Q_{22} \end{bmatrix} \begin{bmatrix} \underline{X}_A \\ \underline{X}_B \end{bmatrix}$$

$$= \underline{x}_A^TQ_{11}\underline{x}_A + 2\underline{x}_A^T\underline{b}_A + \alpha$$
with
\[ b = Q_{12}r \quad \text{and} \quad a = x^T Q_{22} r. \]

The constraints become
\[ V y = \begin{bmatrix} V_A & V_B \end{bmatrix} \begin{bmatrix} x \\ \bar{r} \end{bmatrix} = V_A x + V_B r = y, \]
i.e.,
\[ V_A x = y - V_B r. \]

Due to the special form of \( V \) (cf. Sec. 1.4 - 1.5) this is equivalent to
\[ W x = \bar{m}, \]
where \( w \) is an \( n_w \times (n-k) \) matrix consisting of the first \( n_w \) rows and \( (n-k) \) first columns of \( V \), \( \bar{m} \) consists of the \( n_w \) first components of \( \bar{y} \), and \( n_w = a \cdot d \). Thus we have dropped the constraints on \( \bar{y} \) and incorporated the known values of \( \bar{r} \) into the linear and constant terms of the quadratic form.

Our problem is now
\[
\begin{aligned}
\min_{x, \lambda} \quad & \frac{1}{2} x^T Q_{11} x + x^T b + \frac{1}{2} a \\
\text{for} \quad & W x = \bar{m}.
\end{aligned}
\]

The solution is obtained from the unconstrained problem (\( \lambda \) is a vector of Lagrange multipliers)
\[
\min_{x, \lambda} \frac{1}{2} x^T Q_{11} x + x^T b + \frac{1}{2} a - (W x - \bar{m})^T \lambda = \psi(x, \lambda).
\]

The solution is given by the condition
\[
\text{grad}_{\bar{m}, \lambda} \psi = 0,
\]
\[
\begin{cases}
Q_{11} x + b - W_\lambda^T y = 0, \\
W x - y = 0.
\end{cases}
\]

In partitioned form we write
\[
\begin{pmatrix}
Q_{11} & -W^T \\
W & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix} =
\begin{pmatrix}
-b \\
W_\lambda + b
\end{pmatrix}.
\]

Multiply the first partition with \(WQ_{11}^{-1}\) and subtract from the second partition.

This gives
\[
\begin{pmatrix}
Q_{11} & -W^T \\
0 & WQ_{11}^{-1}W^T
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix} =
\begin{pmatrix}
-b \\
W_\lambda + WQ_{11}^{-1}b
\end{pmatrix},
\]

i.e.
\[
\lambda = (WQ_{11}^{-1}W^T)^{-1}(W_\lambda + WQ_{11}^{-1}b).
\]

Insert this into the first partition and solve for \(x\) to get
\[
x = Q_{11}^{-1} [-b + W^T \lambda]
\]
\[
= Q_{11}^{-1} [-b + W^T (WQ_{11}^{-1}W^T)^{-1}(W_\lambda + WQ_{11}^{-1}b)].
\]

Note that for \(b = 0\) we get the same expression as in Section 2.1.
2.2.1 Algorithm

The solution is computed in the following sequence of steps, using subroutines from the NAG-library. The only differences compared to the algorithm of Section 2.1.1 are the steps 0, 1b, 1c and 5b.

0. Compute $b = Q_{12}z$.

1a. Compute the Choleski factorization $R^TR$ of $Q_{11}$. $R$ is an upper triangular $n \times n$-matrix. Due to the special structure of $Q_{11}$, this is best done by the NAG-routine F01MCF.

1b. Solve $Q_{11}x = b$.

1c. Put $z = z + Wz$.

2. Define $Z^{-1} = Q_{11}v_1$. $Z$ is an $n \times n_c$ matrix. The $n_c$ columns of $Z$ are the solutions of the $n_c$ systems of linear equations

$$Q_{11}Z = V_1^T.$$

These systems are solved using the Choleski factorization of $Q$. This is done by the NAG-routine F04MCF.

3. Form the symmetric, positive definite matrix

$$H = V_1Q_{11}^{-1}V_1^T = V_1Z.$$

$H$ is an $n_c \times n_c$ matrix. Only the upper triangular part of $H$ needs to be computed.

4. Define $w = H^{-1}z$. The $n_c$-vector $w$ is the solution of the symmetric "small" linear system

$$Hw = z,$$

which is solved by the NAG-routine F04ASF.

5a. Form $u = V_1^Tw$. $u$ is an $n$-vector.

5b. Put $y = -b + u$. 
6. Finally, compute $z = Q^{-1}_1 u$. This is equivalent to solving the
system of linear equations

$$Q_1 z = u,$$

which is done by F04MCF.
3. USER INSTRUCTION

This instruction is for the installation on the VAX-computer in the Physics Department of the University of Stockholm.

The data to the program are presented via a terminal in an interactive session in one of the following types. The sessions are hopefully self-explained. Underlined items are the user's replies to the computer.

3.1 No coefficients specified

The program is started by the first command.

```
run glid
The no. of points in the main moving average is 2*a+1.
(1<=a<=40)
Give tail length a:
3
The graduation method will be exact for polynomials of degree G (<=5 and <=a).
Give G:
1
The object function contains a difference of order z.
(1 <=z<=5 and z<a)
Give z:
1
The no. of observations is m (>2*a+z).
Give m:
22
Coefficients of the main moving average known?(YES/NO)
For results, see file FRI2107.50
```
The results from the calculation are stored in an automatically generated file with the name

\[ \text{FRI} < \text{G} > < \text{Z} > < 2a+1 > \].

where \(<\>\) means the character of the numerical value of the enclosed quantity.

The session above generated the result

\[
\begin{array}{cccc}
95681 & 31121 & 982 & -7533 \\
12956 & 27224 & 21356 & 9656 \\
-12956 & 31605 & 39794 & 2977 \\
4119 & 30636 & 42668 & 37401 \\
0 & -20586 & 4951 & 2977 \\
0 & 0 & -9751 & 9656 \\
0 & 0 & 0 & -7533
\end{array}
\]

\[ \text{TRACE} = 0.9037 \times 10^1 \]

The table contains the unknowns \( Y^t_j, j = 1, 2, \ldots, a+1, t = 1, 2, \ldots, a+1 \), arranged in the way described in Section 1.2, i.e., the first column contains \( Y^1_1, Y^1_2, \ldots, Y^1_4 \) and the last column contains \( r_1, r_2, \ldots, r_7 \). Also \( \text{TRACE} = \sum Q_i \).

3.2 The coefficients of the main moving average specified

In this case a file containing the coefficients \( r_j, j = 1, 2, \ldots, a+1 \), of the main moving average must be prepared before the program is executed.
In the file the coefficients must be stored as
r<1>
r<2>

r<a+1>
i.e., sequentially with one value on each row.
The numbers must contain a decimal point, i.e., even if the
value is 5500 it must be given as 5500.0.
If r<a+1> is greater than 1 the program assumes that the co-
efficients in the file are scaled by the factor 10**I, where
I is a positive integer. In that case the file should be
terminated with one row containing the integer I.

Example RC.DAT contains

-7533-0
9656.0
29.77 D
37401.0
5

A session is started with the first command below.

run gid
The no. of points in the main moving average is 2*a+1.
(1<=a<=40)
Give tail length a:
/. The graduation method will be exact for polynomials of
degree G (<=5 and <=a).
Give G:
2
The object function contains a difference of order z.
(1 <=z<=5 and z<a)
Give z:
1
The no. of observations is m (>2*a+z).
Give m:
50
Coefficients of the main moving average known?(YES/NO)

Give the name of the file of the r-coefficients:
rc!'.dat
For results, see file FST2107.KFF
The results are again stored in an automatically generated file with the name
FST<G>Z<2a+1>.KFF
The session above generated the result

\[
\begin{array}{cccc}
95681 & 3121 & 982 & -7533 \\
12956 & 27224 & 21355 & 9656 \\
-12956 & 31605 & 39794 & 29177 \\
4319 & 30636 & 42668 & 37401 \\
0 & -20586 & 4951 & 29177 \\
0 & 0 & -9751 & 9656 \\
0 & 0 & 0 & -7533
\end{array}
\]

with the same organization as in Section 3.1.
4. INSTRUCTIONS FOR INSTALLATION AND MODIFICATION

As the program uses several NAG-library routines, the source files for the main program and all the subroutines should be compiled, linked and loaded together with the appropriate routines from the NAG-library, into an executable module. On the VAX we have named this module GLID.

The program is written in FORTRAN 77, but most of the program is in accordance with the rules of standard FORTRAN.

Only the input routine GIVEDATA and the output routine OUTCOEFF use non-standard FORTRAN. In these routines the character handling facilities of FORTRAN 77 are used. Furthermore, some system dependent file handling is performed there.

If the program is installed on other systems, these two routines may have to be changed.
The main modules of the program interact as follows.

The logical variable GIVENR is used to select between the two algorithm paths described in Sections 2.1.1 and 2.2.1.

The following three pages contain a listing of the main program.
REAL*8 VT (2461, 6), Z (2461, 1), A (128000), DIAG (2461), H (246, 246)
REAL*8 SUM, R (1), WK1 (246), WK2 (246), X (246), B (2461), QSUM, V (246)
INTEGER ND, NA, NZ, M, LOWV (246), UPV (246), INDV (246), COLIN (2461)
INTEGER NROW (2461), UP, LOWCOL (2461), UPCOL (2461)
LOGICAL GIVENR

CALL GIVEDATA (NA, ND, NZ, M, R, GIVENR)
CALL SYSTEM (A, NROW, NA, NZ, M, LAL, NOEKV, GIVENR, R, B)
CALL CONSTRAINT (VT, V, ND, NA, NCON, LOWV, UPV, INDV, LOWCOL, UPCOL,
* COLIN, GIVENR)
CALL F01MCF (NOEKV, A, LAL, NROW, A, DIAG, GIVENR, R, B)

IF (IFAIL .NE. 0) WRITE (5, 91) IFAIL
91 FORMAT ("IFAIL=", I5, "I F01MCF")

IF (.NOT. GIVENR) GOTO 190
C IFTHER-COEFFICIENTS ARE GIVEN WE SOLVE
C Q<1,1> Z=B Q<1,1> IS THE UPPER LEFT PARTITION OF Q
C FOLLOWED BY CALCULATION OF V=(VT)**T*Z+V

DO 110 J = 1, NOEKV
   Z (J, 1) = B (J)
110 CONTINUE
NZDIM = 2461
ISELECT = 1
NR = 1
CALL F04MCF (NOEKV, A, LAL, DIAG, NROW, NR, Z, NZDIM, ISELECT, Z,
C NZDIM, IFAIL)

DO 160 IV = 1, NCON
   SUM = 0.0
   INDEX = INDV (IV)
   LOW = LOWV (IV)
   UP = UPV (IV)
   DO 150 J = LOW, UP
      SUM = SUM + VT (J, INDEX) * Z (COLIN (J), 1)
150 CONTINUE
V (IV) = SUM + V (IV)
160 CONTINUE
190 CONTINUE
C CONSTRUCTION OF PROJECTED MATRIX  H= (VT)**T*Q**(-1)*VT

DO 500 IZ = 1, NCON
   DO 200 J = 1, NOEKV
      Z (J, 1) = 0.0
200 CONTINUE
INDEX = INDV (IZ)
LOW = LOWV (IZ)
UP = UPV (IZ)
DO 300 J = LOW, UP
   Z (COLIN (J), 1) = VT (J, INDEX)
300 CONTINUE

NZDIM = 2461
ISELECT = 1
NR = 1

CALL F04MCF (NOEKV, A, LAL, DIAG, NROW, NR, Z, NZDIM, ISELECT, Z,
             NZDIM, IFAIL)
IF (IFAIL .NE. 0) WRITE (5, 311) IFAIL
311 FORMAT ('IFAIL I F04MCF= ', I5)

DO 400 IV = IZ, NCON
   SUM = 0.0
   INDEX = INDV (IV)
   LOW = LOWV (IV)
   UP = UPV (IV)
   DO 390 J = LOW, UP
      SUM = SUM + VT (J, INDEX) * Z (COLIN (J), 1)
390 CONTINUE
   H (IZ, IV) = SUM
400 CONTINUE
500 CONTINUE

C SOLVE H*Y = V

NHDIM = 246
CALL F04ASF (H, NHDIM, V, NCON, Y, WK1, WK2, IFAIL)
IF (IFAIL .NE. 0) WRITE (5, 591) IFAIL
591 FORMAT ('IFAIL= ', I5 ' I F04ASF ')

C COMPUTE THE VALUE OF THE QUADRATIC FORM

QSUM = 0
DO 600 I = 1, NCON
   QSUM = QSUM + V (I) * Y (I)
600 CONTINUE

C FORM THE VECTOR Z = VT*Y

DO 1500 I = 1, NOEKV
   LOW = LOWCOL (I)
   UP = UPCOL (I)
   SUM = 0
   DO 1400 J = LOW, UP
      INDEX = INDV (J)
      SUM = SUM + VT (I, INDEX) * Y (J)
1400 CONTINUE
   Z (COLIN (I), 1) = SUM
1500 CONTINUE
IF (.NOT.GIVENR) GOTO 1560

C     IF THE R-COEFFICIENTS ARE GIVEN WE CALCULATE Z=Z-B
          DO 1550 I=1,NOEKV
               Z(I,1)=Z(I,1)-B(I)
1550     CONTINUE
1560     CONTINUE

C     SOLVE Q*Z=Z
          ISELECT=1
          NZDIM=2461
          NR=1
          CALL F04MCF (NOEKV,A,LAL,DIAG,NROW,NR,Z,NZDIM,ISELECT,Z,NZDIM,IFAIL)
          IF (IFAIL.NE.0) WRITE(5,1591) IFAIL
1591    FORMAT ('IFAIL= ',I5,' ANROP 2 AV FO4MCF ')

IF (.NOT.GIVENR) GOTO 1800

C     WHEN THE R-COEFFICIENTS ARE GIVEN CONCATENATE THOSE TO THE
C     SOLUTIONVECTOR BEFORE PRINTING.
          ILOW=NOEKV+1
          NOTOT=NOEKV+NA+1
          DO 1700 I=ILOW,NOTOT
               Z(I,1)=R(I-NOEKV)
1700     CONTINUE
1800     CONTINUE

CALL OUTCOEF(Z,QSUM,NA,ND,NZ,M,LOWV,UPV,COLIN,GIVENR)

END
The subroutine SYSTEM

The subroutine SYSTEM and the subprograms it uses are listed below.

SUBROUTINE SYSTEM(A,NROW,NA,NZ,M,LAL,NOEKV,GIVENR,RCOEFO,B)
REAL*8 A (1),B(1),RCOEFO(1),SL
INTEGER NROW ( ),NA,NZ,LAL,NOEKV,M
INTEGER T(40,40),Q21(41,2420),Q22(41,41),ROWN0,ROWT,COLINDEX
LOGICAL GIVENR

CALL TMATRIX(T,NA,NZ)
CALL QMAT21(Q21,NA,NZ)
CALL QMAT22(Q22,NA,NZ,M)
C STORE THE MATRIX Q IN THE VECTOR A ACCORDING TO THE
C STORAGE SCHEME FOR THE NAG-ROUTINE F01.

NEXTirrnx=1
C TREAT THE FIRST NA+1 DIAGONAL BLOCKS OF Q<1,1>
NA1=NA+1
DO 500 IP=1,NA1
   LOW=(IP-1)\*NA
   00 400 I=1,NA
      NSUB=NZ+1
      IF (I.LE.NZ)NSUB=I
      NROW(LOW+I)=NSUB
      JLOW=I-NSUB+1
      DO 300 J=JLOW,I
         A(NEXTINDEX)=2*DFLOAT(T(I,J))
         NEXTINDEX=NEXTINDEX+1
300  CONTINUE
400  CONTINUE
500  CONTINUE

C CONTINUE WITH THE DIAGONAL BLOCKS NA+2,NA+3,... 2*NA OF Q<1,1>
ROWN0=NA
NA2=NA+2
NTWO=2*NA
DO 1500 IP=NA2,NTWO
   LOW=LOW+ROWN0
   ROWN0=ROWN0-1
   ROWT=IP-NA
   DO 1400 I=1,ROWN0
      NSUB=NZ+1
      IF (ROWT.LE.NZ)NSUB=ROWT
      JLOW=ROWT-NSUB+1
      IF (JLOW.LT.(IP-NA))JLOW=IP-NA
      NROW (LOW+I)=ROWT-JLOW+1
      DO 1300 J=JLOW,ROWT
1400  CONTINUE
1300  CONTINUE
1500  CONTINUE
A (NEXTINDEX)=2*DFLOAT (T(ROWT,J))
NEXTINDEX=NEXTINDEX +1

1 300  CONTINUE
   ROWT=ROWT+1
1 400  CONTINUE
1 500  CONTINUE

C REMEMBER THE NUMBER OF STORED ELEMENTS OF Q<1,1>
KLAL=NEXTINDEX-1
C DECODING OF Q21 AND Q22

NOROW=NA+1
NCOL=NA*(3*NA+1)/2
DO 2500 I=1,NOROW
   COLINDEX= 1
   DO 2100 J=1,NCOL
      IF (Q21(I,J).NE.O) GOTO 2101
      COLINDEX=COLINDEX+1
   2100 CONTINUE
2101 CONTINUE
   NROW(NCOL+I)=NCOL+1-COLINDEX+I
   IF (COLINDEX.GT.NCOL) GOTO 2201
   DO 2200 J=COLINDEX,NCOL
      A(NEXTINDEX)=DFLOAT(Q21(I,J))
      NEXTINDEX=NEXTINDEX+1
   2200 CONTINUE
2201 CONTINUE
   DO 2300 J=1,I
      A(NEXTINDEX)=DFLOAT(Q22(I,J))
      NEXTINDEX=NEXTINDEX+1
   2300 CONTINUE
2500 CONTINUE

LAL=NEXTINDEX-1
NOEKV=NCOL+NOROW

IF (.NOT.GIVENR) GOTO 3500
C IFTHE R-COEFFICIENTS ARE GIVEN THE SIZE OF THE MATRIX IS
C REDUCED AND THE VECTOR B=Q<1,2>R IS CALCULATED.
LAL=KLAL
NOEKV=NOEKV- (NA+1 )
NA1=NA+1
IUP=NA*(3*NA+1)/2
DO 3000 I=1,IUP
   SL=0
   DO 2900 J=1,NA1
      SI=SL+Q21(J,I)*RCOEF(J)
   2900 CONTINUE
   B(I)=SL
3000 CONTINUE
3500 CONTINUE

RETURN
END
SUBROUTINE QMAT22(Q22,NA,NZ,M)
INTEGER Q22(41,1),S1(81,81),S2(81,81),K22(81,81),PZ(41),Q,QUP
INTEGER QLOW,NA,NZ,M

CALL FACTZ(NZ,PZ)

IUP=2*NA+1-NZ
DO 500 IP=1,IUP
   IPOS=1
   QUP=NZ-1
   DO 400 Q=O,QUP
      ISUM=O
      JUP=NZ-Q
      DO 300 J=1 ,JUP
         ISUM=ISUM+J*PZ(J)*PZ(J+Q)
      300 CONTINUE
      S1(IP+Q,IP)=IPOS*ISUM
      S1(IP,IP+Q)=S1(IP+Q,IP)
      IPOS=-IPOS
   400 CONTINUE
   QUP=2*NA+1-IP
   DO 450 Q=NZ,QUP
      S1(IP+Q,IP)=O
      S1(IP,IP+Q)=O
   450 CONTINUE
500 CONTINUE

ILOW=2*NA+2-NZ
IUP=2*NA+1
DO 700 IP=ILOW,IUP
   QUP=2*NA+1-IP
   DO 600 Q=O,QUP
      S1(IP+Q,IP)=O
      S1(IP,IP+Q)=O
   600 CONTINUE
700 CONTINUE

CALL FACTORIAL(NZ,PZ)

MFACT=M-2*NA-NZ
IUP=2*NA+1
DO 1500 IP=1,IUP
   DO 1300 Q=1,NZ
      INDEX=IP+Q
      IF (INDEX.GT.IUP) GOTO 1290
      S2(IP+Q,IP)=MFACT*PZ(Q)
      S2(IP,IP+Q)=S2(IP+Q,IP)
   1290 CONTINUE
   S2(IP,IP)=MFACT*PZ(NZ+1)
   QUP=2*UA+1 -IP
   QLOW=NZ+1
   DO 1400 Q=QLOW,QUP
      S2(IP+Q,IP)=O
      S2(IP,IP+Q)=O
   1400 CONTINUE
1500 CONTINUE
SUBROUTINE TMA

INTEGER T(40,1),KF(40)
CALL FACTZ(NZ,KF)
DO 500 I=1,NA
DO 400 J=1,I
ISUM=O
ISIGN=1
IJ=I+J
IF ((IJ/2)*2.NE.IJ) ISIGN=-1
DO 300 K=1,NA
IMK=I-K
IF (IMK.GT.NZ) IFACT1=0
IF (IMK.LT.O) IFACT1=0
IF (IMK.EQ.O) IFACT1=1
IF (IMK.GT.O.AND.IMK.LE.NZ) IFACT1=KF(IMK)
JMK=J-K
IF (JMK.GT.NZ) IFACT2=0
IF (JMK.LT.O) IFACT2=0
IF (JMK.EQ.O) IFACT2=1
IF (JMK.GT.O.AND.JM.X.LE.NZ) IFACT2=KF(JMK)
ISUM=[SlJM+IFACT1 *IFACT2
300 CONTINUE
T(I,J)=ISIGN*ISUM
400 CONTINUE
500 CONTINUE
RETURN
END
SUBROUTINE QMAT21 (Q21, NA, NZ)
INTEGER Q21 (41, 1), U (40, 6), R (40, 15), K12 (2420, 81), PSI (80)
CALL FACTORIAL (NZ, PSI)
CALL UVECTORS (U, NZ, NA, PSI)
CALL RMATRICES (U, R, NZ, NA)
CALL KMAT12 (K12, R, NZ, NA)
IUP = NA * (3 * NA + 1) / 2
DO 200 I = 1, IUP
   DO 100 J = 1, NA
      Q21 (J, I) = 2 * (K12 (I, J) + K12 (I, 2 * NA + 2 - J))
100   CONTINUE
Q21 (NA + 1, I) = 2 * K12 (I, NA + 1)
200 CONTINUE
RETURN
END

SUBROUTINE FACTORIAL (N, KF)
INTEGER N, KF (1)
C CALC OF FACTORIALS (2N OVER N), (2N OVER N-1), ... (2N OVER 0)
C RESULT STORED AS KF (J) = (-1)**J * (2N OVER N-J) FOR J = 1, 2 ... N
C KF (0) = (2N OVER N) IS STORED IN KF (N+1)
NR = (N/2) * 2 - N
ISIGN = 1
IF (NR .EQ. 0) ISIGN = -1
KF (N) = ISIGN
N1 = N - 1
DO 100 K = 1, N1
   INDEX = N - K + 1
   KF (INDEX - 1) = -KF (INDEX) * (N-K+1) / K
100 CONTINUE
KF (N+1) = KF (1) * (N+1) / N
IF (N+1 .LT. 0) KF (N+1) = -KF (N+1)
RETURN
END
SUBROUTINE FACTZ(N,KF)
INTEGER KF(1)
C CALC OF FACTORIALS (N OVER I) I=1,••• N
C KF(I)= (N OVER I) I=1,••• N
KF(1)=N
IF (N.EQ.1) GOTO 200
DO 100 K=2,N
   KF(K)=KF(K-1)*(N-K+1)/K
100 CONTINUE
200 CONTINUE
RETURN
END

SUBROUTINE UVECTORS (U,NZ,NA,PSI)
INTEGER U(40,1),PSI(1)
C CALC OF THE VECTORS U(J) J=1,••• Z
C U(J) = (0,••• 0,PSI(NZ),••• PSI(J) )**T
C THE VECTORS ARE STORED COLUMNWISE IN THE MATRIX U
C AS U= (U(1),U(2),••• U(NZ) )
DO 1000 J=1,NZ
   NUP=NA+J-NZ-1
   DO 100 I=1,NUP
      U(I,J)=0
100     CONTINUE
   NLOW=NUP+1
   INDEX=NZ
   DO 200 I=NLOW,NA
      U(I,J)=PSI(INDEX)
      INDEX=INDEX-1
200     CONTINUE
1000 CONTINUE
RETURN
END
SUBROUTINE RMATRICES(U,R,NZ,NA)
INTEGER U(40,1),R(40,1)
C CALL OF R-MATRIX
C R(p) = (U(p),U(p-1),...,U(1)) p=1,2,... NZ
C THE MATRICES ARE STORED IN THE LARGE R-MATRIX
C R= (R(1),R(2),...,R(NZ))
JCOLR=0
DO 500 JP=1,NZ
   DO 400 J=1,JP
   JCOLR=JCOLR+1
   JCOLU=JP-J+1
   DO 300 I=1,NA
      R(I,JCOLR)=U(I,JCOLU)
   300 CONTINUE
400 CONTINUE
500 CONTINUE
RETURN
END

SUBROUTINE KMAT12(K12,R,NZ,NA)
INTEGER K12(2420,1),R(40,1),RCOL
C CALL OF THE MATRIX K12
IUP=NA*(NA+1)+NA*(NA-1)/2
JUP=2*NA+1
DO 100 I=1,IUP
   DO 100 J=1,JUP
      K12(I,J)=0
100 CONTINUE
DO 500 IP=1,NZ
   KROWLOW=(IP-1)*NA
   RCOL=IP(IP-1)/2
   DO 400 J=1,IP
      RCOL=RCOL+1
      KROW=KROWLOW
      DO 300 I=1,NA
         KROW=KROW+1
         K12(KROW,J)=R(I,RCOL)
300 CONTINUE
400 CONTINUE
500 CONTINUE
IPLOW' = NZ + 1
IPUP = NA + 1
DO 1500 IP = IPLOW, IPUP
   KROWLOW = (IP - 1) * NA
   RCOL = (NZ - 1) * NZ / 2
   JLOW = IP - NZ + 1
   JUP = JLOW + NZ - 1
   DO 1400 J = JLOW, JUP
      RCOL = RCOL + 1
      KROW = KROWLOW
      DO 1300 I = 1, NA
         KROW = KROW + 1
         K12(KROW, J) = R(I, RCOL)
      1300 CONTINUE
   1400 CONTINUE
1500 CONTINUE
IPLOW = NA + 2
IPUP = 2 * NA
DO 2500 IP = IPLOW, IPUP
   NY = IP - (NA + 1)
   KROWLOW = (NA + 1) * NA + (NY - 1) * NA - NY * (NY - 1) / 2
   RCOL = (NZ - 1) * NZ / 2
   JLOW = IP - NZ + 1
   JUP = JLOW + NZ - 1
   DO 2400 J = JLOW, JUP
      RCOL = RCOL + 1
      KROW = KROWLOW
      ILOW = IP - NA
      DO 2300 I = ILOW, NA
         KROW = KROW + 1
         K12(KROW, J) = R(I, RCOL)
      2300 CONTINUE
   2400 CONTINUE
2500 CONTINUE
RETURN
END
5.3 The subroutine CONSTRAINT

SUBROUTINE CONSTRAINT(V, HL, D, A, NOCONSTR, LOW, UP, IND, 
  * LOWCOL, UPCOL, COLIN, GIVENR)
REAL*8 V(2461,1), HL(1) 
INTEGER D, A, LOW(1), UP(1), IND(1), LOWCOL(1), UPCOL(1), COLIN(1) 
LOGICAL GIVENR
INTEGER LOW, LOWHL, UP, I, J, IP
REAL*8 FI(81,6) 
LOWHL=0
UP=0
LOW=1
DO 300 IP=1, A
  NROW=A+IP
  UP=UP+NROW
  CALL ORTBASIS(NROW, D, FI)
  DO 200 I=1, D
    INDEX=1
    DO 190 J=LOW, UP
      V(J, I)=FI(INDEX, I)
      INDEX=INDEX+1
  190 CONTINUE
  LI=LOWHL+I
  HL(LI)=FI(IP, I)
  LOWV(LI)=LOW
  UPV(LI)=UP
  INDV(LI)=I
  200 CONTINUE
  DO 250 J=LOW, UP
    CALL NEWINDEX(J, IP, A, JNEW)
    COLIN(J)=JNEW
    LOWCOL(J)=LOWHL+1
    UPCOL(J)=LOWHL+D
  250 CONTINUE
  LOWHL=LOWHL+D
  LOW=LOW+NROW
  300 CONTINUE
LOW=UP+1
UP=LOW+A-1
NROW=2*A+1

CALL ORTBASIS(NROW,D,FI)

IVCOL=1
DO 500 I=1,D
   INDEX=1
   IF ((I-(I/2)*2).EQ.O) GOTO 495
   DO 490 J=LOW,UP
      V(J,IVCOL)=FI(INDEX,I)+FI(2*A+2-INDEX,I)
      INDEX=INDEX+1
   490 CONTINUE
   LI=LOWHL+IVCOL
   HL(I)=FI(A+1,I)
   V(UP+1,IVCOL)=FI(A+1,I)
   LOWV(LI)=LOW
   UPV(LI)=UP+1
   INDV(LI)=IVCOL
   IVCOL=IVCOL+1
495 CONTINUE
500 CONTINUE

NOCONSTR=LOWHL+IVCOL-1

UP=UP+1
DO 600 J=LOW,UP
   COLIN(J)=J
   LOWCOL(J)=LOWHL+1
   UPCOL(J)=NOCONSTR
600 CONTINUE

IF (GIVENR) NOCONSTR=A*D

RETURN
END

SUBROUTINE ORTBASIS(M,D,FI)
REAL*8 FI(81,1),GAMMA,BETA,CAPPA,SUM,SUMX,TERM,SUMNX
INTEGER D,D1

C FIND AN ORTHOGONAL BASIS FOR THE SPACE OF POLYNOMIALS OF
C DEGREE LESS THAN D. THE ORTOGONALITY IS WITH RESPECT TO
C THE INNERPRODUCT
C <f,g>= \int f(x)g(x)dx + \cdots \int f(t)g(t)
C THE RESULT IS A SET OF ORTHOGONAL VECTORS STORED COLUMNWISE
C IN THE UPPER LEFT CORNER OF THE MATRIX FI.
SUM=0
SUMX=0
SUMNX=0
GAMMA=(M+1)/2DO
CAPPA=SQRT (M/1DO)

DO 100 I=1,M
   FI(I,1)=1 DO/CAPPA
   FI(I,2)=1-GAMMA
   TERM=FI(I,2)*FI(I,2)
   SUM=SUM+TERM
   SUMX=SUMX+I*TERM
   SUMNX=SUMNX+I*FI(I,1)*FI(I,2)
100 CONTINUE

D1=D-1
DO 500 N=2,D1
   CAPPA=SQRT (SUM)
   BETA=SUMX/SUM
   GAMMA=SUMNX/CAPPA
   SUM=0
   SUMX=0
   SUMNX=0
   DO 400 I=1,M
      FI(I,N)=FI(I,N)/CAPPA
      FI(I,N+1)=(I-BETA)*FI(I,N)-GAMMA*FI(I,N-1)
      TERM=FI(I,N+1)**2
      SUM=SUM+TERM
      SUMX=SUMX+TERM*I
      SUMNX=SUMNX+I*FI(I,N+1)*FI(I,N)
400 CONTINUE
500 CONTINUE
RETURN
END

SUBROUTINE NEWINDEX (ININD,T,A,OUTIND)
INTEGER ININD,T,A,OUTIND

.J=LIND-(T-1)*A-T*(T-1)/2
IF(J.LT.(A+2))GOTO 100
OUTIND=(J-1)*A+T-(J-A)*(J-A-1)/2
RETURN
100 CONTINUE
OUTIND=(J-1)*A+T
RETURN
END
5.4 Input and output subroutines

SUBROUTINE GIVEDATA (A,D,Z,M,R,GIVENR)
REAL*8 R(1)
INTEGER A,D,Z,M,AL,G
LOGICAL GIVENR
CHARACTER*1 ANS
CHARACTER*15 INFILE

WRITE (6,11)
11 FORMAT ('The no. of points in the main moving average is 2*a+1 .
* ,/,' (<=a<40),/,' Give tail length a: )
READ (6,15) A
15 FORMAT (I)

WRITE (6,21)
21 FORMAT ('The graduation method will be exact for polynomials of
* ,/,' degree G (<=5 and <=a). ,/,' Give G: )
READ (5,15) G
D=G+1

WRITE (5,31)
31 FORMAT ('The object function contains a difference of order z.,
* ,/,' (<=z<=5 and z<a ),/,' Give z: )
READ (6,15) Z

WRITE (6,41)
41 FORMAT ('The no. of observations is m (2*a+2). ,/,' Give m: )
READ (5,15) M

WRITE (6,101)
101 FORMAT ('Coefficients of the main moving average known? (YES/NO) )
READ (6,102) ANS
102 FORMAT (A)
IF (ANS .EQ. Y .OR. ANS .EQ. Y .OR. ANS .EQ. J .OR. ANS .EQ. J )
* GOTO 200
GOTO 1000
200 CONTINUE

    WRITE(5,208)  
208 FORMAT ('Give the name of the file of the r-coefficients: ')  

C In the file the coefficients must be stored as  
C          r<1>  
C          r<2>  

C          r<a+1>  
C i.e. sequentially with one value on each row.  
C The numbers must contain a decimal point, i.e. even if the  
C value is 5500 it must be given as 5500.0.  
C If r<a+1> is greater than 1 the program assumes that the  
C coefficients in the file are scaled by the factor 10**I  
C where I is a positive integer. In that case the file is  
C terminated with one row containing the integer I.  

READ (5,209) INFILE  
209 FORMAT (A)  

OPEN(UNIT=9,FILE=INFILE,STATUS=OLD)  
GIVENR=.TRUE.  
A1=A+1  
READ (9,302) (R(I),I=1,A1)  
302 FORMAT (F)  

IF(R(A1).LE.1.DO)GOTO400  

READ (9,307) ISCALE  
307 FORMAT (I)  

MSCALE=f  
DO 320I=1,ISCALE  
320 MSCALE=MSCALE*10  

DO 350 I=1,A1  
350 R(I)=R(I)/MSCALE  

400 CONTINUE  
1 000 CONTINUE  

RETURN  
END
SUBROUTINE OUTCOEF(X,QSUM,A,D,Z,M,LOWV,UPV,INDROW,GIVENR)
REAL*8 X(1),QSUM
INTEGER A,D,Z,M,LOWV( ),UPV( ),INDROW( )
INTEGER A1,A2,OUTMAT(81,41)
LOGICAL GIVENR
CHARACTER*1 RFILE
CHARACTER*1 GP
C CREATE A FILENAME AND FILE FOR OUTPUT OF RESULTS
C ! THIS IS NOT STANDARD FORTRAN!

LED=Z*100+2*A+1
ND=D-1
IF (ND.EQ.0) GP=0
IF (ND.NE.0) WRITE(GP,5)ND
5 FORMAT(1)
IF (.NOT.GIVENR) WRITE(RFILE,10)GP,LED,M
IF (GIVENR) WRITE(RFILE,20)GP,LED
10 FORMAT(FRI,A1,I3,.I3)
20 FORMAT(FST,A1,I3,.KFF)
OPEN(UNIT=8,FILE=RFILE,STATUS=NEW)
C PREPARE DATA FOR PRINTING
A1 =A+1
A2=2*A+1
DO 200 I=1,A2
   DO 190 J=1,A1
      OUTMAT(I,J)=0
190 CONTINUE
200 CONTINUE
DO 400 J=1,A1
   IND=(J-1)*D+1
   LOW=LOWV(IND)
   UP=UPV(IND)
   DO 300 I=LOW,UP
      OUTMAT(I-LOW+1,J)=NINT(X(INDROW(I))*100000)
300 CONTINUE
400 CONTINUE
DO 500 I=1,A
   OUTMAT(2*A+2-I,A1)=OUTMAT(I,A1)
500 CONTINUE
DO 700 I=1,A2
   WRITE(8,709)(OUTMAT(I,J),J=1,A1)
700 CONTINUE
709 FORMAT(11I8)
WRITE(S,709)
IF (.NOT.GIVENR) WRITE(6,719)QSUM
719 FORMAT(' '6)TRACE=',E15.
WRITE(S,809)RFILE
809 FORMAT(' For results, see file ',A11)
RETURN
END
APPENDIX 1. Contruction of $Q$

Here is a safe way of stepwise construction of the matrix $Q$. This algorithm was given by Hoem (1983).

Let

$$T_{ij} = (-1)^{i+j} \sum_{k=1}^{a} \frac{z}{i-k} \frac{z}{j-k},$$

for $i = 1, 2, \ldots, a$, and $j = 1, 2, \ldots, a$, and let

$$F_p = T, \quad \text{for } p = 1, 2, \ldots, a+1,$$

$$F_p = \{T_{ij}: i, j = p-a, p-a+1, \ldots, a\} \quad \text{for } p = a+2, \ldots, 2a.$$

Then define

$$K_{11} = \text{diag}\{F_1, F_2, \ldots, F_{2a}\}.$$  

Note that $K_{11}$ has a dimension $\frac{1}{4}a(3a+1) \times \frac{1}{4}a(3a+1)$. Let

$$\psi_j = (-1)^{j} \frac{2z}{z-j}$$

for integer $j$, and note that $\psi_j = 0$ for $j > z$. Define

$$u_j = (\psi_{j+1}, \psi_{j+2}, \ldots, \psi_z)^T \quad \text{for } j = 1, 2, \ldots, z.$$  

Then $u_j$ is a column vector with $a$ components. Also let

$$R_p = \begin{cases} 
(u_p, u_{p-1}, \ldots, u_1), & \text{for } p = 1, 2, \ldots, z-1, \\
(u_z, u_{z-1}, \ldots, u_1), & \text{for } p = z, z+1, \ldots, 2a+1.
\end{cases}$$

Note that

$$u_j = (0, \ldots, 0, \psi_z, \psi_{z-1}, \ldots, \psi_j)^T,$$

with $a+j-z-1$ leading zeros, so a lot of the elements of each $R_p$ are 0.
Now let $C_p = R_p$ for $p = 1, 2, \ldots, a+1$, while for $p > a+1$
we define $C_p$ as $R_p$ with the $p-a-1$ first rows omitted:

$$C_p = \begin{cases} R_p, & \text{for } p = 1, 2, \ldots, a+1, \\ \{\begin{array}{c} (R_p)_{ij} \\
\end{array} \}_{i = p-a, p-a+1, \ldots, a; j = 1, 2, \ldots, z}, & \text{for } p = a+2, \ldots, 2a. \end{cases}$$

Now define

$$C_p = \begin{cases} C_p, & \text{for } p = 1, 2, \ldots, z, \\ \{0(a, 2a+1-p), C_p, 0(a, 2a+1-p)\}, & \text{for } p = z+1, \ldots, a+1, \\ \{0(2a+1-p, p-z), C_p, 0(2a+1-p, 2a+1-p)\}, & \text{for } p = a+2, \ldots, 2a. \end{cases}$$

Here $0(m,n)$ is an $m \times n$ matrix all of whose elements are zero. Then

$C_p$ has $2a+1$ columns.

Let

$$K_{12} = \begin{bmatrix} \bar{C}_1 \\ \bar{C}_2 \\ \vdots \\ \bar{C}_{2a} \end{bmatrix}, \quad \text{and } K_{21} = K_{12}^T.$$ 

Then $K_{12}$ is a $1 \times (3a+1) \times (2a+1)$ matrix. Let $S_1$ be a symmetric

$(2a+1) \times (2a+1)$ matrix with elements

$$(S_1)_{p, p+q} = \begin{cases} (-1)^q \sum_{j=1}^{z} j(z)(j+q), & \text{for } q = 0, 1, \ldots, z-1, \\ 0, & \text{for } q = z, z+1, \ldots, 2a+1-p, \end{cases}$$

for $p = 1, 2, \ldots, a+1.$
when \( p = 1, 2, \ldots, 2a+1-z \). For \( p = 2a+2-z, \ldots, 2a+1 \), and \( q = 0, 1, \ldots, 2a+1-p \), \((S)_{p,p+q}\) is again defined by the above summation formula. Also let \( S_2 \) be a symmetric \((2a+1)\times(2a+1)\) matrix defined by

\[
(S_2)_{p,p+q} = (-1)^q, \\
\text{for } q = 0, 1, \ldots, 2a+1-p, \text{ and } p = 1, 2, \ldots, 2a+1.
\]

Note that \((S_2)_{p,p+q} = 0\) for \( q > z \). As before, the parameter \( m \) represents the number of observation points on the curve which is to be smoothed.

Finally, let \( K_{22} = S_2 + 2S_1 \) and

\[
K = \begin{bmatrix}
K_{11} & K_{12} \\
K_{21} & K_{22}
\end{bmatrix}
\]

The matrix \( Q \) can now be obtained from \( K \) in the following way:

Partition \( Q \) as

\[
Q = \begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{bmatrix}
\]

with \( Q_{21} = Q_{12}^T \), \( Q_{11} \) a symmetric \((a+1)\times (a+1)\) matrix, and \( Q_{22} \) a symmetric \((a+1)\times (a+1)\) matrix. Thus \( Q_{12} \) has dimension \((a+1)\times (a+1)\).

Furthermore,

\[
Q_{11} = 2K_{11},
\]

\[
(Q_{12})_{ij} = \begin{cases} 
2 \{ (0,2)_{i,j} + (12)_{a-2-i,j} \}, & \text{for } J = 1, 2, \ldots, a, \\
2 (12)_{z, a+1}, & \text{for } J = a+1,
\end{cases}
\]

for \( i = 1, 2, \ldots, a(a+1) \).
Also, 

\[
(Q_{22})_{ij} = \begin{cases} 
2(K_{22})_{ij} + (K_{22})_{i,2a+2-j} & \text{for } i = 1, 2, \ldots, a, \\
(K_{22})_{a+1,j} + (K_{22})_{a+1,2a+2-j} & \text{for } j = 1, 2, \ldots, a, \\
(K_{22})_{a+1,a+1} & \text{for } i = a+1, j = a+1, \\
(K_{22})_{a+1,i} + (K_{22})_{a+1,2a+2-i} & \text{for } i = 1, 2, \ldots, a, j = a+1.
\end{cases}
\]
APPENDIX 2. NAG library routines

The following subroutines from the NAG library, Mark 9, have been used:

F01MCF,
F04MCF,
F04ASF.

Descriptions of the routines can be found in the NAG library manuals.
ACKNOWLEDGEMENTS

Many thanks to Jan M. Hoem for suggesting and financing the work of this report.
For her excellent typing I thank Hulda Hjörleifsdóttir.

REFERENCES


SRRD-1  Hoem, Jan M. and Randi Selmer: The interaction between premarital cohabitation, marriage, and the first two births in current Danish cohorts, 1975. (March 1982)


SRRD-3  Hoem, Jan M.: Balancing bias in vital rates due to an informative sampling plan. (April 1982)

SRRD-4  Hoem, Jan M.: Distortions caused by nonobservation of periods of cohabitation before the latest. (May 1982)
Demography 20(4), 491-506.

SRRD-5  Hoem, Jan M.: The reticent trio: Some little-known early discoveries in insurance mathematics by Oppermann, Thiele, and Gram. (July 1982)

SRRD-6  Hoem, Jan M. and Bo Rennermalm: Biases in cohabitational nuptiality rates caused by nonobservation of periods of cohabitation before the latest: An empirical note. (Dec. 1982)

SRRD-7  Funck Jensen, Ulla: An elementary derivation of moment formulas for numbers of transitions in time-continuous Markov chains. (Dec. 1982)


SRRD-10 Hoem, Jan M.: Multistate mathematical demography should adopt the notions of event-history analysis. (Feb. 1983)


SRRD-12 Lyberg, Ingrid: Nonresponse effects on survey estimates in the analysis of competing exponential risks. (April 1983)

SRRD-13 Hartmann, Michael: Past and recent experiments in modeling mortality at all ages. (Nov. 1983)

(Jan. 1984)

SRRD-16  Hoem, Jan M.: Statistical analysis of a multiplicative model 
and its application to the standardization of vital rates: 
A review.  (Feb. 1984)

SRRD-17  Lindberg, Bengt: A numerical algorithm for the calculation of 
coefficients in moving averages.  (March 1984)

Also available on request:

Hoem, Jan M.: Sveriges hundraåriga fruktomsfellsfall. Pp. 185-212 
in Barn? Författare, forskare och skolungdömdiskuterar var-

Universitet 1479-1979, Bind XII. Kbenhavn: G.E.C. Gads 
Forla, 1983.

Hoem, Jan M.: A contribution to the statistical theory of linear 
graduation. Insurance Mathematics and Economics 3(1984), 
1-17.