International Institute for
Applied Systems Analysis
IIASA www.iiasa.ac.at

MODEL: A General Program for Estimating Parametrized Model Schedules of Fertility, Mortality, Migration, and Marital and Labor Force Status Transitions

Rogers, A. and Planck, F.

IIASA Working Paper

WP-83-102

October 1983

WORKING PAPER

MODEL: A GENERAL PROGRAM FOR ESTIMATING PARAMETRIZED MODEL SCHEDULES OF FERTILITY, MORTALITY, MIGRATION, AND MARITAL AND LABOR FORCE STATUS TRANSITIONS

Andrei Rogers
Friedrich Planck

October 1983
WP-83-102

IIASA
International Institute for Applied Systems Analysis

NOT FOR QUOTATION
WITHOUT PERMISSION
OF THE AUTHORS

MODEL: A GENERAL PROGRAM FOR ESTIMATING PARAMETRIZED MODEL SCHEDULES OF FERTILITY, MORTALITY, MIGRATION, AND MARITAL AND LABOR FORCE STATUS TRANSITIONS

Andrei Rogers
Friedrich Planck

October 1983
WP-83-102

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

Low fertility levels in IIASA countries are creating aging populations whose demands for health care and income maintenance (social security) will increase to unprecedented levels, thereby calling forth policies that will seek to promote increased family care and worklife flexibility. The new Population Program will examine current patterns of population aging and changing life-styles in IIASA countries, project the needs for health and income support that such patterns are likely to generate during the next several decades, and consider alternative family and employment policies that might reduce the social costs of meeting these needs.

An important feature of the Population Program's research agenda is the development of computer program packages that will allow members of its collaborative network to work with comparable and consistent analytical methods. The program described in this paper is part of the set of POPNET programs that have been created for this purpose. It should enable a user to fit observed data with parametrized model schedules and, in this way, to provide a means for data smoothing and inference.

Andrei Rogers
Leader
Population Program

ACKNOWLEDGMENTS

A number of individuals contributed to the writing of the program described in this paper. The first version focused only on migration and was prepared by Richard Raquillet as far back as 1977. Luis Castro greatly expanded and refined this version in subsequent years and made a major contribution to the program's evolution. Walter Kogler further extended it to include fertility and mortality schedules and introduced a number of improvements. To all three go our sincere thanks.

CONTENTS

MODEL: A GENERAL PROGRAM FOR ESTIMATING PARAMETRIZED MODEL SCHEDULES OF FERTILITY, MORTALITY, MIGRATION, AND MARITAL AND LABOR FORCE STATUS TRANSITIONS

Demography has been characterized as the quantitative study of fundamental demographic processes, such as mortality, fertility, migration, and marriage (Bogue 1968). These processes may be viewed as transitions that individuals experience during the course of their life cycle. Individuals are born, age with the passage of time, enroll in school, enter the labor force, get married, reproduce, migrate from one region to another, retire, and ultimately die. These transitions contribute to changes in various population stocks through simple accounting identities. For example, the number of married people at the end of each year is equal to the number at the beginning of the year plus new marriages and arrivals of married migrants less divorces, deaths, widowings, and the outmigration of part of the married population.

The study of transition patterns generally begins with the collection of data and the estimation of missing observations, continues with the calculation of the appropriate rates and corresponding probabilities, and often ends with the generation of simple projections of the future conditions that would arise were these probabilities to remain unchanged.

I. PARAMETRIZED MODEL SCHEDULES

The use of mathematical functions, expressed in terms of a small set of parameters, to smooth and describe parsimoniously schedules of age-specific rates is a common practice in demography. A large number of such functions have been proposed and fitted to mortality and fertility data, for example, and the results have been widely applied to data smoothing, interpolation, comparative analysis, data inference, and forecasting. The relevant literature is vast and entry into it can be made from such representative publications as Brass (1971), Coale and Demeny (1966), Coale and Trussell (1974), Heligman and Pollard (1979), Hoem et al. (1981), and United Nations (1967).

More recently, the range of parametrized schedules has been expanded to include interstate transfers such as migration (Rogers, Raquillet, and Castro 1978; Rogers and Castro 1981) and changes in marital status other than first marriage (Williams 1981). Thus it is now possible to define a model (hypothetical) multistate dynamics that describes the evolution of a single-sex population exposed to parametrized schedules of mortality, fertility, migration, and several forms of marital status change (that is, first marriage, widowhood, divorce, and remarriage).

The role of model schedules in parametrized multistate population dynamics is two-fold. *First*, model schedules allow one to condense an enormous amount of information about transitions between states of existence or the occurrences of vital events in each year into a few parameters. *Second*, model schedules provide a manageable number of interpretable descriptive statistics, for each demographic transition or vital event in each year, the time series of which can be the basis for econometric estimation. The criteria for the selection of appropriate model schedules should emphasize the interpretability of the parameters, their success in characterizing the important features of demographic behavior, and the goodness-of-fit of the parametrized schedules to available data.

Rates and Schedules of Rates

The basic initial measure for most demographic analysis is a central rate, defined for a population in a given state during a particular time span. Its numerator is a count of occurrences of an event; its denominator describes person-years of exposure (number of people times the duration of their exposure to the event in question). In the demographic literature such rates have been called *occurrence/exposure* rates.

In mortality studies, for example, occurrence/exposure rates associate the number of deaths during a given interval with the number of person-years of exposure to death experienced by the population at risk. In labor force studies they take the form of accession and separation rates that relate entrances and exits to the at-risk inactive and active populations, respectively. Analogous principles apply to the construction of fertility rates, nuptiality rates, and divorce rates.

Empirical schedules of age-specific occurrence/exposure rates exhibit remarkably persistent regularities in age pattern. Mortality schedules, for example, normally show a moderately high death rate immediately after birth, after which the rates drop to a minimum between ages 10 to 15, then increase slowly until about age 50, and thereafter rise at an increasing pace until the last years of life. Fertility rates generally start to take on nonzero values at about age 15 and attain a maximum somewhere between ages 20 and 30; the curve is unimodal and declines to zero once again at some age close to 50. Similar unimodal profiles may be found in schedules of first marriage, divorce, and remarriage. The most prominent regularity in age-specific schedules of migration is the high concentration of migration among young adults; rates of migration also are high among children, starting with a peak during the first year of life, dropping to a low point at about age 16, turning sharply upward to a peak near ages 20 to 22, and declining regularly thereafter except for a possible slight hump or upward slope at the onset of the principal ages of retirement. Although data on rates of labor force entry and exit are very scarce, the few published studies that are available indicate that regularities in age pattern also may be found in such schedules. Figure 1 illustrates a number of typical age profiles exhibited by occurrence/exposure rates in multistate demography.

The shape or *profile* of a schedule of age-specific occurrence/exposure rates is a feature that may be usefully examined independently of its intensity or *level*. This is because there are considerable empirical data showing that although the latter tends to vary significantly from place to place, the former remains remarkably similar.

The level at which occurrences of an event or a flow take place in a multistate population system may be represented by the area under the curve of the particular schedule of rates. In fertility studies, for example, this area is called the gross reproduction rate if the rates refer to parents and babies of a single sex. By analogy, therefore, we shall refer to areas under all schedules of occurrence/exposure rates as *gross production rates*, inserting the appropriate modifier when dealing with a particular event or flow, for example, gross mortality production rate and gross accession production rate. The term "production" is retained throughout in order to distinguish this aggregate measure of level from the other more common gross rates used in demography, such as the directional gross (instead of net) rate of migration.

Examples

Fertility

Among the relatively large number of different parametric functions that have been recently proposed for representing schedules of age-specific fertility, the formula put forward by Coale and Trussell (1974) has assumed a certain pre-eminence. This formula can be viewed as the product of two component schedules: a model nuptiality schedule and a model marital fertility schedule. The former adopts the double-exponential first marriage function of Coale and McNeil (1972):

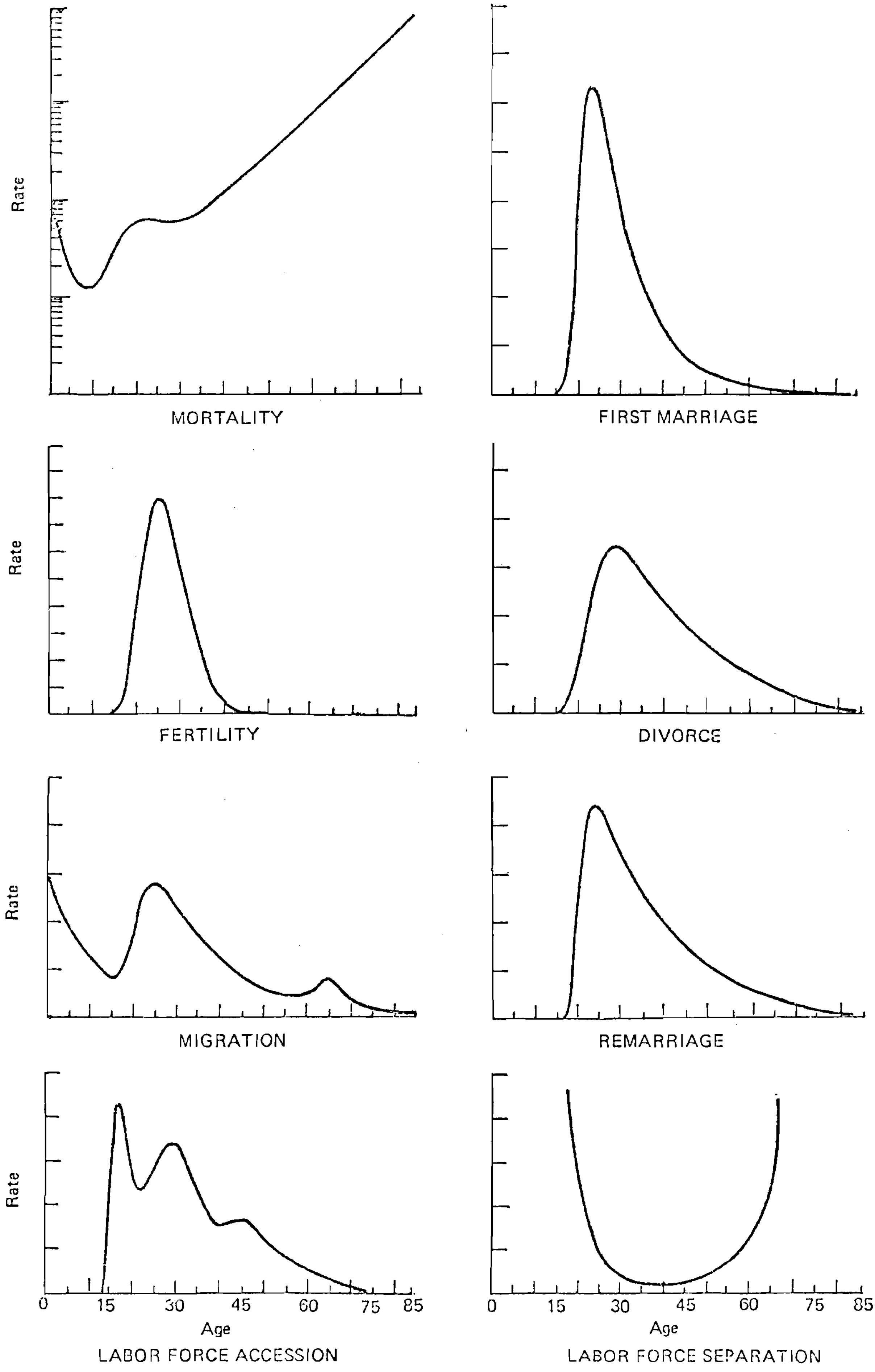

Figure 1. Multistate schedules.

$$g(x) = \frac{0.19465}{k} e^{-\frac{0.174}{k}(x - x_0 - 6.06k) - e^{-\frac{0.2881}{k}(x - x_0 - 6.06k)}} \qquad (1)$$

where x_0 is the age at which a consequential number of first marriages begin to occur, and k is the number of years in the observed population into which one year of marriage in the standard population is transformed. Integrating, one finds

$$G(x) = \int_0^x g(a)da$$

which when multiplied by the proportion who will ever marry, represents the proportion married at each age.

Coale argues that marital fertility either follows a pattern that Henry (1961) called *natural* fertility or deviates from it in a regular manner that increases with age, such that the ratio of marital fertility to natural fertility can be expressed by

$$\frac{r(x)}{n(x)} = M\ e^{mv(x)}$$

where M is a scaling factor that sets the ratio $r(x)/n(x)$ equal to unity at some fixed age, and m indicates the degree of control of marital fertility. The values of v(x) and n(x) are specified by Coale, and they are assumed to remain invariant across populations and over time.

Multiplying the two-parameter model schedule of proportions ever married at each age by the one-parameter model schedule of marital fertility, Coale and Trussell (1974) generated an extensive set of model fertility schedules that have been shown to describe empirical schedules with surprising accuracy. Their representation

$$f(x) = G(x) \cdot r(x) = G(x) n(x) e^{mv(x)} \tag{2}$$

allows one to obtain the age profiles (but not the levels) of fertility, which depend only on the fixed single-year values of the functions $n(x)$ and $v(x)$, and on estimates for x_0, k, and m.

A double-exponential function [similar to that used by Coale and McNeil (1972) for first marriages] also may be used to describe fertility rates at age x:

$$f(x) = gae^{-\alpha(x-\mu) - e^{-\lambda(x-\mu)}} \tag{3}$$

where the shape of the curve is defined by the three parameters, α, μ, and λ, and the level of the curve is defined by the scaling parameter a, and g, the gross fertility rate, which is the sum of all the age-specific fertility rates. Although these parameters (apart from g) are not easily interpretable, it is possible to derive the propensity, mean, variance, and mode of the double-exponential function in terms of them (Coale and McNeil 1972; Rogers and Castro 1981; and Sams 1981).

Marital Status

Coale and McNeil's (1972) double-exponential model schedule of first marriages, discussed above, was introduced a decade ago. Parametrized schedules of other changes in marital status, however, seem to have been first used only recently, in a study carried out by the IMPACT Project in Australia (Powell 1977). Working with a detailed demographic data bank produced by Brown and Hall (1978), Williams (1981) fitted gamma distributions to Australian rates of first marriage, divorce, remarriage of divorcees, and remarriage of widows, for each year from 1921 to 1976. These model schedules provided adequate descriptions of Australian marital status changes, although some difficulties arose with age distributions that exhibited steep rises in early ages; in particular, the age distributions of first marriages. This difficulty was overcome by the addition of a second time-invariant gamma distribution.

Functions based on the Coale-McNeil double-exponential distribution, given in Equation 3, seem better able to cope with the problem of steeply rising age distributions than the gamma distribution. Although the parameters of both functions can be expressed in terms of a propensity, mean age and variance in age, the double-exponential function requires a further parameter —the modal age—whose movements over time may be more difficult to model and project.

Mortality

Three principal approaches have been advanced for summarizing age patterns of mortality: *functional descriptions* in the form of mathematical expressions with a few parameters (Benjamin and Pollard 1980), *numerical tabulations* generated from statistical summaries of large data sets (Coale and Demeny 1966), and *relational procedures* associating observed patterns with those found in a standard schedule (Brass 1971).

The search for a "mathematical law" of mortality, has, until very recently, produced mathematical functions that were successful in capturing empirical regularities in only parts of the age range, and numerical tabulations have proven to be somewhat cumbersome and inflexible for applied analysis. Consequently, the relational methods first proposed by William Brass have become widely adopted. With two parameters and a standard life table, it has become possible to describe and analyze a large variety of mortality regimes parsimoniously.

In 1979 Heligman and Pollard published a paper setting out several mathematical functions that appear to provide satisfactory representations of a wide range of age patterns of mortality. Their function describes the variable q(x), the probability of dying within one year for an individual at age x. We have found it more useful to focus instead on d(x), the annual death rate at age x. Thus we adopt the slightly modified Heligman and Pollard formula suggested by Brooks et al. (1980) of the IMPACT Project:

$$d(x) = d_I(x) + d_A(x) + d_S(x) \qquad \text{for } x = 0,1,\ldots,100+ \qquad (4)$$

where

$$d_I(x) = \begin{cases} Q_0 & \text{for } x = 0 \\ Q_1^{x^\gamma} & \text{for } x > 0 \end{cases}$$

$$d_A(x) = Q_A\, e^{-\left(\frac{\ln x - \ln X_A}{\sigma}\right)^2} \qquad \text{for } x \geq 0$$

and

$$d_S(x) = Q_S \frac{e^{x/X_S}}{1 + Q_S K e^{x/X_S}} \qquad \text{for } x \geq 0$$

Heligman and Pollard interpreted the three terms in their formula as representing infant and childhood mortality (I), mortality due to accidents (A), and a senescent mortality (S) component which reflects mortality due to aging.

Migration

A recent study of age patterns in migration schedules (Rogers and Castro 1981) has shown that such patterns exhibit an age profile that can be adequately described by the mathematical expression:

$$m(x) = a_1 e^{-\alpha_1 x} + a_2 e^{-\alpha_2 (x-\mu_2) - e^{-\lambda_2 (x-\mu_2)}} + R + c \qquad (5)$$

where

$$R = a_3 e^{-\alpha_3 (x-\mu_3) - e^{-\lambda_3 (x-\mu_3)}}$$

if the curve has a retirement peak,

$$R = a_3 e^{\alpha_3 x}$$

if the curve has an upward retirement slope, and

$$R = 0$$

if the curve has neither and is approximately horizontal at the post-labor force ages. The migration rate, m(x), therefore, depends on values taken on by 11, 9, or 7 parameters, respectively.

The shape of the second term, the labor force component of the curve, is the double exponential formula put forward by Coale and McNeil (1972). The first term, a simple negative exponential curve, describes the migration age profile of children and adolescents. Finally, the post-labor force component is a constant, another double-exponential, or an upward sloping positive exponential.

In addition to the parameters and derived variables defined previously in the discussion of model double-exponential fertility schedules, we now introduce three additional measures useful for the study of migration age profiles: the index of *child dependency*

$$\delta_{12} = \frac{a_1}{a_2}$$

the index of *parental-shift regularity*

$$\beta_{12} = \frac{\alpha_1}{\alpha_2}$$

and the low point x_ℓ. The first measures the pace at which children migrate with their parents, the second indicates the degree to which the pattern of the migration rates of children mirrors that of their parents, and the third identifies the age at which the lowest migration rate occurs among teenagers.

Other Transitions

The notion of model schedules may be used to describe a wide range of demographic transitions. We have considered mortality, fertility, migration, marriage, divorce, and remarriage. We could as easily have focused on flows between different states of, for example, income, education, health, and labor force activity.

Consider, for example, the flows between active and inactive statuses in studies of labor force participation. Rates of entry into the labor force, called accession rates, exhibit an age profile that can be described as the sum of three double exponential distributions. Rates of exit from the labor force, called separation rates, may be described by a U-shaped curve defined as:

$$h(x) = a_1 e^{-\alpha_1 x} + a_3 e^{\alpha_3 x} + c \qquad (6)$$

A Computer Program: Model

The rest of this paper describes a computer program that may be used to fit model schedules to observed data. The heart of this program is a nonlinear algorithm that iterates to the data until it in some sense (e.g., in the least-squares sense) produces an optimal fit of model to data. Two such algirithms are embedded in the program. The first is a routine based on the Levenberg-Marquand algorithm that is available from IMSL (International Mathematics and Statistics Library). This routine is copyrighted; consequently, it is not reproduced here. It is, however, available at many university computing centers.

An alternative, and in many respects a better, routine is one based on a Gauss-Newton type of algorithm developed in the Soviet Union at VNIISI, the All Union Research Institute of Systems Studies. This routine, written by Drs. V. Golubkov and S. Scherbov, was brought to IIASA by Dr. Scherbov and was added to the program as an alternative parameter estimation procedure. Since the algorithm is not copyrighted we are able to include it in the program listing contained in this paper.

II. BASIC USER'S MANUAL

Introduction

This manual describes the basic steps for using IIASA's general model schedule parameter estimation program. As the user becomes better acquainted with it, he will find it helpful to examine the Advanced User's Manual, presented in sections III and IV, which offers more general ways of using the program and additional facilities, and which also contains more detailed descriptions.

What the program does

The model schedule program has built in to it several mathematical functions that may be used to describe the age patterns of various demographic variables such as mortality, fertility, marriage, divorce, migration, etc. The parameters of these mathematical functions are estimated by an iterative procedure in which the program tries to fit the model as closely as possible to a set of observed data. The user has to provide the observations (called 'input data') and to select the mathematical function (called 'model schedule') he wishes to apply.

How to select a model schedule

Each model schedule is identified by a code. Figure 2/1 lists the available model schedules and their associated codes. The selection of a model schedule is done on a separate input file ('input directives'). If, for example, a user wishes to fit a model schedule to observed (age-specific) fertility rates, he might choose the double exponential function (code=23), giving the input directives:

1st directive (at card/line 1): fit
2nd directive (at card/line 2): mod=23

Writing starts at the first column. The input directives are read from file unit 5 (standard input).

How to provide the input data (observations)

Suppose the user has a data file containing age-specific fertility rates. Provided that this file is written 'normally', i.e. it is not unformatted (binary) or free formatted (*-format), it can be used as an input file with only a few modifications. The user has to add two cards/lines at the beginning of the data file: the first is a title card which may be empty; the second has to contain the age range and the age interval of the input data, and the format with which they are to be read. If the fertility rates were single-year-of-age data ranging from, say, 15 to 50 and were written in format (5f10.6) the first two lines of the input data file would be:

```
1st card/line: fertility rates by age of mother
2nd card/line:  15 50  1(5f10.6)
```

The first card/line is read in format (a60), the second is read in format (3i3,a20). The input data file is read from file unit 1.

Output

The output produced by the example input file is written on file unit 6 (standard output) and consists of three parts:

1st part: General program information telling the user what the program did, step by step.

2nd part: A table of the estimated model schedule parameters, along with additional information, of which the most important is the 'goodness of fit'; small values (e.g. < 10) indicate a good fit, big numbers (e.g. > 25) reflect a poor fit.

3rd part: A table of observed versus estimated rates.

Simple extensions

There are several basic input variations that may be used to modify the parameter estimates and the output to be produced:

(a) The iterative estimation of the model schedule parameters may be started with the initial parameter values explicitly named by the user. Depending on the specific application, the built-in defaults may not be appropriate to reach a solution. In such a case the user should try different initial parameter estimates.

(b) The user may want to modify an available model schedule by excluding certain parts of the mathematical function. This can be done by setting the appropriate parameters equal to zero. It is also possible to fix parameters at values other than zero.

(c) The estimated model schedules usually are normalized, so that the area under the curve will be unity. It is possible, however, to delete the normalization.

(d) The output may be supressed (output parts 2 and/or 3 mentioned above) or it may be extended by a graphical output.

(a) How to enter initial parameter values explicitly

Initial parameter estimates are appended after the code of the model schedule on the input directive 'mod'. The order of parameters can be seen from an output of the program run without parameter specifications, for example. It is important to note that either no initial parameter values should be given or all parameters of the model schedule should be initialized. The input format is free, i.e. the values are separated by blanks or commas. A sample input directive is:

```
mod=23,.1,20,.3,.1
```

if the double exponential function (code 23) is to have initial parameters values a, mu, alpha, and lambda of 0.1, 20.0, 0.3, and 0.1, respectively.

(b) How to keep parameter values fixed

Forcing a parameter to equal a certain value means that the parameter will keep its initial value throughout the estimation procedure. Therefore the fixing of parameter values requires that initial values be set by the 'mod' input directive as described above. Which parameter(s) will remain fixed is defined by the input directive 'fix' which contains a '0' or '1' for each parameter, where '1' stands for 'fix'. Thus if the user wishes to keep parameter mu fixed at 20 the above input directive would need to be enlarged to state:

```
mod=23,.1,20,.3,.1
fix=0,1,0,0
```

(c) How to avoid normalization

Surpressing the normalization of the model schedule may be achieved by specifying the input directive:

nor=0

(d) How to get more or less output

To supress as much output as possible the input directive

out=0

should be used. To get only the table of estimated parameter values but not the table of observed and estimated rates the user should enter:

out=6

If the user wishes to extend the output by a graphical plot of estimated versus observed rates he should use the input directive 'gra':

gra

Figure 2/1: The Available Model Schedules and Their Codes

```
Code   Function                                              Name
--------------------------------------------------------------------------------------
 0     f(x) =  0                                             zero schedule

 1     f(x) =  obs(x)                                        observed schedule

11     f(x) =    p9                       x=0                Heligman-Pollard model
                 p1*(x**p2)               x>0                 mortality schedule
               + p5*exp(-((ln(x)-ln(p6))/p7)**2)
               + p3*exp(x/p4)/(1+p3*exp(x/p4))
               + p3

12     f(x) =    p1*exp(-p2*x)                               Rogers model
               + p5*exp(-p7*(x-p6)-exp(-p8*(x-p6)))           mortality schedule
               + p3*exp(p4*x)
               + p9

21     f(x) =  p1*n(x)*exp(p2*v(x))                          Coale-Trussell model
               *integral from (p4-11.36*p5) to                fertility schedule
                infinity of:                                  with parameters
               (1.2313/p3)                                    S and x-bar
               *exp(-(1.145/p3)*(x-p4-.805*p3)
               -exp(-(1.896/p3)*(x-p4-.805*p3)))

22     f(x) =  p1*n(x)*exp(p2*v(x))                          Coale-Trussell model
               *integral from p4 to infinity of:              fertility schedule.
               (.1946/p3)                                     with parameters
               *exp(-(.174/p3)*(x-p4-6.06*p3)                 k and x0
               -exp(-(.288/p3)*(x-p4-6.06*p3)))

23     f(x) =  p1*exp(-p3*(x-p2)-exp(-p4*(x-p2)))            double exponential

31     f(x) =    p1*exp(-p3*(x-p2)-exp(-p4*(x-p2)))          double exponentials
               + p5*exp(-p7*(x-p6)-exp(-p8*(x-p6)))
               + p9*exp(-p11*(x-p10)-exp(-p12*(x-p10)))
               + p13

32     f(x) =    p1*exp(-p2*x)                               Rogers-Castro model
               + p3*exp(-p5*(x-p4)-exp(-p6*(x-p4)))           migration schedule
               + p7*exp(-p9*(x-p8)-exp(-p10*(x-p8)))          with retirement peak
               + p11

33     f(x) =    p1*exp(-p2*x)                               Rogers-Castro model
               + p3*exp(-p5*(x-p4)-exp(-p6*(x-p4)))           migration schedule
               + p7*exp(p3*x)                                 with retirement slope
               + p9

41     f(x) =  p1*exp(-p3*(x-p2)-exp(-p4*(x-p2)))            double exponential
```

Figure 2/2: The Basic Input Directives and Their Meaning

	Input Directive	Effect/Action
	fit	A model schedule will be fitted to observed data.
	mod=icode	The model schedule with the code 'icode' will be selected.
or:	mod=icode,p(1),p(2),...p(n)	The model schedule with the code 'icode' will be selected and the initial parameter values for the fitting procedure will be p(1), p(2), ...p(n).
optional:	fix=ifix(1),ifix(2),...ifix(n)	Parameter i will remain constant if ifix(i) is 1.
optional:	nor=0	No normalization of the model schedule will be done.
optional:	out=0/out=6	Output will be supressed/ /partly supressed.
optional:	gra	A graphical output will be produced.

III. A D V A N C E D U S E R ' S M A N U A L :

GENERAL REMARKS

Program facilities

The program has several executable facilities. Most of them will only be executed if the user tells the program to do so (by means of 'program directives'). The available facilities are (in the order of their executional sequence):

- Read program directives. Program directives are always read from file unit 5.

- Read input data (obsvervations) (see directive 'dat'). Missing input data do not cause an error. No input data are required if only a model schedule has to be computed.

- Interpolate input data (optional - see directive 'cub'). Model schedule values (results) are always given in one-year age intervals. Observed data, however, often have 5-year intervals. In such a case it is possible to replace the input by interpolated observations by applying cubic splines (IMSL-algorithm).

- Normalize observed data / adjust observed data in level (optional - see directive 'nor'). For reasons of comparison it is often desirable to normalize schedules. This means that the area under the curve, i.e. the gross rate in case of rates, will be one. Similarly the observed data can be adjusted to meet any other gross rate.

- Calculate model schedule (optional - see directive 'mod'). Definition of a model schedule (selection of a mathematical function and its parameter values) causes the computation of functional values for ages x=0,1,...100.

- Fit model schedule to observed data (optional - see directive 'fit'). The main task of the program is the parameter estimation, i.e., the fitting of a model schedule to observed data. It is done by a finite-difference analog of the Gauss-Newton algorithm (VNIISI-routine) or a modified Levenberg-Marquardt algorithm (IMSL-routine). Both algorithms use the residuals between observations and estimations in an iterative process for estimating the functional parameters. The model schedule defined by input directive 'mod' is used as an initial estimate (starting point).

  The VNIISI routine tends to stop too early when a satisfying solution is not yet reached. The IMSL routine on the other hand often does not stop within a reasonable number of iterations; moreover, the quality of fit may decrease when the number of iterations increases when the IMSL routine is applied.

- Cut model schedule (optional - see directive 'cut'). For some purposes it is preferable to have a model schedule covering only a part of the full age range 0 to 100 having zero values outside of the chosen range. Moreover the last age group (w) can be made 'open-ended' comprising ages w and over.

- Normalize model schedule / adjust model schedule in level (optional - see directive 'nor'). In general a fitted model schedule will not have exactly the same gross rate as the observed schedule. To enforce it a normalization can be applied to the model schedule in the same way as for the observed schedule (see above). In addition also the 'level parameters' of the model schedule will be adjusted to maintain the link between function parameters and function values. However this is only possible if level parameters exist (e.g. not (not exact) for the Heligman-Pollard model mortality schedule's senescent part).

- Print table of parameters, table of observed and estimated schedules, and internal summary information (optional - see directive 'out').

- Print observed and estimated schedule and components of model schedule as well as logarithms of observed and estimated schedules (optional - see directive 'out').

- Print goodness-of-fit table (optional - see directive 'out').

- Plot observed versus estimated schedule (optional - see directive 'gra').

- Store results for summary table / print summary table (optional - see directive 'out').

- Store results for table of summary statistics / print table of summary statistics (optional - see directive 'out'). - This facility is only valid for the standard Rogers-Castro model migration schedule with retirement peak (code 32).

General format of program directives

The program directives tell the program what to do. They are read from unit 5 and have the general format:

key=kp1,kp2,... kpn

where

- key: keyword (columns 1-3)
- kp1,kp2,...: keyword parameters (columns 5-130), separated by commas or blanks (read in *-format)

Lines starting with '#' or '*' in column 1 are treated as comments.

Column 4 may contain a '=', a blank, or any other character except an 'x' in which case the keyword is treated as it would have never appeared before (this may be necessary if more than one run is pursued (see below)).

Trailing keyword parameters which are to have values equal to zero may be omitted. (Note that zero values may be replaced by default values.)

Order of program directives

Whereas the order of execution steps is fixed the order of program directives is arbitrary. If a keyword appears more than once without being separated by an 'end'-directive (see below) only the last directive is recognized and the previous one(s) is (are) ignored.

Repeated runs

Passing all executional steps (see Program facilities) is called a 'run'. To have several runs within the same program run (i.e., without reaching the 'stop'-statement of the program) the 'end'-directive has to be used to separate the according sets of program directives and to indicate that a new run has to be pursued.

When a new run is started the program directives of the previous run(s) remain in use as long as they are not replaced by new ones. Thereby it is sufficient to define features that apply to all runs only once. Thus if the user wants to have, say, the same kind of output in each run he has to enter the according 'out'- and/or 'gra'-directive only for the first run.

If the user wants to have a single input file only containing program directives plus input data (see directive 'dat') the 'end'-directive has to be inserted before the input data for that run. Figure 3/1 shows the structure of the input files in case of repeated runs.

Figure 3/1: Structure of Input Files in Case of n Runs

Case 1: Two Input Files (1 and 5)

File 5

.
directives (1)
.
end
.
directives (2)
.
end

.
.
.

.
directives (n)
.
end-of-file

File 1

.
input data (1)
.
.
input data (2)
.

.
.
.

.
input data (n)
.
end-of-file

Case 2: One Input File (5)

File 5

.
directives (1)
.
end
.
input data (1)
.
directives (2)
.
end
.
input data (2)
.

.
.
.

.
directives (n)
.
end
.
input data (n)
.
end-of-file

Input data

The input data for one run consist of 1 or 2 'data sets'. If 2 sets are input, rates will be computed from the 2 sets: the first set is assumed to contain population data, the second set is assumed to contain event or move/transition data. Thus if age-specific female population data (set 1) and birth data (by age of mother; set 2) are input birth rates will be computed and used as 'observed data'.

Whether the input data consist of 1 or 2 data sets may be explicitly defined (see directive 'dat'). Implicitly it is assumed that input data should be rates. Therefore one data set is expected if none of its values is greater than 1.0 (checked by the program if no explicit definition is given), otherwise a second data set is expected.

IV. ADVANCED USER'S MANUAL: INPUT DIRECTIVES

PARAMETER ESTIMATION (CURVE FITTING)

directive 'con'

```
con
```

Required to control parameter estimation by printing the residual sum of squares and the estimated parameter values at each iteration step.

Reference procedure: subroutine RESID

PARAMETER ESTIMATION (CURVE FITTING)

directive 'cub'

```
cub=mi1,ma1,iv1,ic,sm,ipr
```

Required to interpolate observed data (by cubic spline application). (The 'cub'-directive is only valid when the IMSL program library is available.)

- mi1: lowest age to use (for interpolating)
- ma1: highest age to use (for interpolating)
- iv1: age interval to use (for interpolating)
- ic=0: cubic spline is applied to cumulated values
  =1: cubic spline is applied to original values
  (recommended and default value is 0)
- sm: stopping criterion (recommended and default value is 0)
- ipr: for error search purposes set ipr=1

Remark: In most cases no parameter at all needs to be given; mi1, ma1 and iv1 have by default the same values as defined by the input data set or as redefined by the 'dat'-directive, respectively.

Reference procedure: subroutine OBSCUB

PARAMETER ESTIMATION (CURVE FITTING)

```
directive 'cut'
---------------

cut=mi1,ma1,opnend,maxage

Required to cut a model schedule and/or to make the last age group
of a model schedule open-ended.

- mi1: new lowest age group (values for ages below mi1 will be 0)
- ma1: new highest age group (values for ages beyond ma1 will be 0)
       (default is: keep last age group as it presently is).
- opnend=0: the model schedule will not be made open-ended.
        =1: the model schedule will be made open-ended; oprend will
            be set to 2 if the model schedule code (see directive
            'mcd') is less than 10 or greater than 19, and to 3
            otherwise.
        =2: the model schedule will be made open-ended; the popula-
            tion structure for ages higher than ma1 is assumed to
            decrease linearly between ages ma1 and maxage
        =3: the model schedule will be made open-ended; the popula-
            tion structure for ages higher than ma1 is estimated by
            the model schedule rates (which are supposed to be death
            rates in this case)
        (default is 1 if both mi1 and ma1 are 0, and 0 otherwise)
- maxage: last age before population will be zero (default is 100)
          (only required for making last age group open-ended)

Reference procedure: subroutine SCHCUT
```

PARAMETER ESTIMATION (CURVE FITTING)

directive 'dat'

dat=mi,ma,sets,ifile

Required to redefine the way in which input data are read.

- mi,ma: lowest and highest ages to use (redefinition) (if one does not want to use all input data in the program one does not have to change the input data file but one can select a certain age range)
- sets: if sets is 1 only one set of data is read, otherwise the program tries to find out by itself whether one data set (containing rates) or two data sets (containing population data + move/transition data) are input.
- ifile: unit from which input data are read (default is 1)

Reference procedure: subroutine INOBS

PARAMETER ESTIMATION (CURVE FITTING)

directive 'end'

```
end
```

Required to separate two sets of program directives.

Remark: 'end' can be replaced by '---'.

Reference procedure: subroutine INPAR

```
---------------- PARAMETER ESTIMATION (CURVE FITTING) -----------------
I                                                                      I
I                                                                      I
I directive 'fit'                                                      I
I ---------------                                                      I
I                                                                      I
I fit=prog,res,iter,nsig,eps,delta,xdelta,op,ipr                       I
I                                                                      I
I                                                                      I
I Required to fit a model schedule (can be omitted if only defaults    I
I are to be used and if a 'fix', 'min', or 'max' directive is in use). I
I                                                                      I
I                                                                      I
I - prog=1: VNIISI fit routine is used                                 I
I      =2: IMSL fit routine is used (only valid if IMSL is available!)I
I     (default is 1)                                                   I
I - res=0: relative residuals are used: (x-y)/sqrt(y)                  I
I      =1: absolute residuals are used: x-y                            I
I      =2: logarithmic residuals are used: log(x)-log(y)               I
I   If 10 is added (res=10,11,12) the absolute values of the residuals I
I   are used (all residuals positive).                                 I
I - max: maximum number of iterations (default is 1000)                I
I - nsig: the IMSL and the VNIISI routines stop if or two successive   I
I   iterations the parameters agree to nsig digits (default is 3)      I
I - eps: the IMSL routine also stops if on two successive iterations   I
I   the residual sum of squares have a relative difference less than   I
I   or equal to eps (default is .000001)                               I
I - delta: the IMSL routine also stops if the norm of the approximate  I
I   gradient is less than or equal to delta (default is .000001)       I
I - xdelta: when starting the VNIISI routine assumes that the initial  I
I   parameters have variances of xdelta*p*p (where p is the initial    I
I   parameter estimate) and initial increments of xdelta*p (xdelta-de- I
I   fault is .01). Note that p is set to 1 by default when the VNIISI  I
I   routine is used (see below)!                                       I
I - op: if op=1 the initial parameter values are used directly by the  I
I   fit routines; if op=-1 each parameter is normalized to 1 first.    I
I   (default is -1 for prog=1 and 1 for prog=2).                       I
I - ipr: for error search set ipr to 1.                                I
I                                                                      I
I                                                                      I
I Reference procedure: subroutine SCHFIT                               I
I                                                                      I
I                                                                      I
I Note: VNIISI is an acronym that represents (in Russian) the All      I
I             Union Research Institute of Systems Studies              I
I       IMSL is an acronym for Interational Mathematical and Statisti- I
I             cal Libraries, Inc.                                      I
I                                                                      I
------------------------------------------------------------------------
```

```
---------------- PARAMETER ESTIMATION (CURVE FITTING) ------------------
I                                                                      I
I                                                                      I
I                                                                      I
I                                                                      I
I directive 'fix'                                                      I
I ---------------                                                      I
I                                                                      I
I                                                                      I
I                                                                      I
I fix=fix1/fix2/... fixn                                               I
I                                                                      I
I                                                                      I
I Required to keep model schedule parameters fixed (constant) during   I
I the estimation process.                                              I
I                                                                      I
I                                                                      I
I - fixi=0: model schedule parameter i will be free (will not remain   I
I           fixed during the estimation process)                       I
I       =1: model schedule parameter i will be kept fixed during the   I
I           estimation process                                         I
I                                                                      I
I                                                                      I
I Reference procedure: subroutine SCHFIT                               I
I                                                                      I
I                                                                      I
I                                                                      I
I                                                                      I
------------------------------------------------------------------------
```

PARAMETER ESTIMATION (CURVE FITTING)

directive 'gra'

gra=ymin,ymax,irow,icol,ifile

Required to produce graphical output (observed versus estimated rates).

- ymin: minimum on ordinate scale (default is 0.0)
- ymax: maximum on ordinate scale (default is .1)
- irow: number of rows used for ordinate scale (default is 50); if the output is to appear on the screen irow=17 is recommended
- icol: number of columns used for abscissa scale (default is 100 for irow greater than 17, and 75 otherwise)
- ifile: output unit (default is 6)

Reference procedure: subroutine OUTGRA

PARAMETER ESTIMATION (CURVE FITTING)

directive 'max' and 'min'

```
min=xmin1,xmin2,... xminn
max=xmax1,xmax2,... xmaxn
```

Required to return to initial model schedule parameter estimates if certain minimum or maximum values are exceeded during the parameter estimation process.

- xmini, xmaxi: minimum, maximum value for model schedule parameter i (defaults are 0.0 and 100.0, respectively)

Reference procedure: subroutine SCHFIT

PARAMETER ESTIMATION (CURVE FITTING)

directive 'mod'

mod=icode,p1,p2,..., pn

Required to calculate a model schedule and/or to estimate model schedule parameters.

- icode: code of model schedule (see figure 2/1)
- pi: value of model schedule parameter i

  Remark: If no pi-values at all are given, then default values are used (the default value depends on the model schedule selected by 'icode' on input directive 'mod').

Reference procedure: subroutine SCHCAL

PARAMETER ESTIMATION (CURVE FITTING)

directive 'nor'

nor=xnorm,ynorm

Required to normalize input data and/or to normalize model schedule data. Note that if one wishes to normalize both input and model schedule data to 1 the 'nor'-directive may be omitted.

- xnorm=0: observations will not be normalized
  #0: observations will be normalized (gross rate = xnorm)
- ynorm=0: model schedule will not be normalized
  #0: model schedule will be normalized (gross rate = ynorm)

Remark: If no 'nor'-directive is given 'nor=1,1' is assumed !

Reference procedure: subroutine OBSNOR and subroutine SCHNOR

```
---------------- PARAMETER ESTIMATION (CURVE FITTING) ------------------
I                                                                       I
I                                                                       I
I                                                                       I
I                                                                       I
I directive 'out'                                                       I
I ---------------                                                       I
I                                                                       I
I                                                                       I
I                                                                       I
I out=ifile1,ifile2,ifile3,ifile4,ifile5,ifile6,ifile7                  I
I                                                                       I
I                                                                       I
I Required to select output and to define output units.                I
I                                                                       I
I                                                                       I
I - ifile1: output unit for table of model schedule parameters         I
I           (default is 6)                                              I
I - ifile2: output unit for table of observed and estimated rates      I
I           /table output (default is 6)                                I
I - ifile3: output unit for internal summary information (default is 0)I
I - ifile4: output unit for table of observed and estimated rates      I
I           /file output (default is 0)                                 I
I - ifile5: output unit for summary table (default is 0);              I
I           if mod is 32 (see directive 'mod') a table of summary      I
I           statistics is also printed                                  I
I - ifile6: output unit for estimated moves/transitions (default is 0);I
I           (only meaningful if input data contain population data,    I
I           i.e. if the input data consist of two data sets)           I
I - ifile7: output unit for goodness-of-fit table (default is 0)       I
I                                                                       I
I                                                                       I
I Remarks: (a) If a unit is equal to 0 (by definition or by default)   I
I              output will be supressed.                                I
I          (b) If no 'out'-directive is given 'out=6,6' is assumed !   I
I                                                                       I
I                                                                       I
I reference procedure: subroutine OUT                                   I
I                                                                       I
I                                                                       I
I                                                                       I
I                                                                       I
-------------------------------------------------------------------------
```

```
---------------- PARAMETER ESTIMATION (CURVE FITTING) -----------------
I                                                                      I
I                                                                      I
I example program directives file                                      I
I -------------------------------                                      I
I                                                                      I
I                                                                      I
I I--------- --------- --------- --------- --------- --------- ------I I
I Imod=32                                                            I I
I Ifit                                                               I I
I I---                                                               I I
I Imod=41,.15,20,.14,.33                                             I I
I Inor=1                                                             I I
I Iout=6,6,0,0,0,0,6                                                 I I
I Igra                                                               I I
I I--------- --------- --------- --------- --------- --------- ------I I
I                                                                      I
I                                                                      I
I Explanation:                                                         I
I                                                                      I
I Two runs of parameter estimation will be performed (because there    I
I are two sets of program directives, separated by a '---'-directive   I
I (equivalent to an 'end'-directive)).                                 I
I                                                                      I
I Run 1: (1) Input data are read (default)                             I
I        (2) Input data are normalized (gross rate = 1.0) (default)    I
I        (3) A standard Rogers-Castro model migration schedule is      I
I            calculated (directive 'mod' with icode=32) using standard I
I            (default) parameters                                      I
I        (4) The parameters of the model schedule are estimated, i.e,  I
I            the model schedule is fitted to the input data (directive I
I            'fit'). The VNIISI parameter estimation procedure is used I
I            (default)                                                 I
I        (5) The estimated model schedule is normalized (gross rate    I
I            = 1.0) (default)                                          I
I        (6) The estimated model schedule parameters and rates are     I
I            written on unit 6 (default)                               I
I                                                                      I
I Run 2: (1) Input data are read (default)                             I
I        (2) Input data are normalized (gross rate = 1.0) (directive   I
I            'nor' with xnorm=1)                                       I
I        (3) A single double exponential model schedule is calculated I
I            using model schedule parameters a=0.15, mu=20.0, alpha=   I
I            0.14, lambda=0.33 (directive 'mod' with icode=41 and      I
I            parameter values .15,20,.14,.33)                          I
I        (4) The model schedule parameters are estimated, i.e. the     I
I            model schedule is fitted to the input data (directive     I
I            'fit' is still active) with the VNIISI procedure          I
I        (5) The estimated model schedule parameters and rates and a   I
I            goodness-of-fit table are written on unit 6 (directive    I
I            'out' with ifile1=6,ifile2=6 and ifile7=6)                I
I        (6) A graph of observed versus estimated rates is printed on  I
I            unit 6 (directive 'gra')                                  I
I        Note that the model schedule is not normalized (directive     I
I        'nor' with ynorm=0)                                           I
I                                                                      I
I                                                                      I
I                                                                      I
I                                                                      I
------------------------------------------------------------------------
```

V. PROGRAMMER'S MANUAL

Program language

The program is written for FORTRAN5 (FORTRAN77) compilers.

The IIASA compiler enables one to read from internal files (i.e. character variables) in *-format. This is not standard FORTRAN77. Thus using the program outside IIASA may make it necessary to have formatted input of keyword parameters. In that case the free (*-)format read commands in the reference procedures (named in section IV) have to be changed to formatted input read commands. To keep the advantages of the *-format remove the comment-'C's from the lines just before the read-statement and replace the file name in the read statement by the variable name 'iofile'. E.g. in subroutine INOBS

change

```
c     call iopar(iofile,datin)
      read(datin,*)yes,mi,ma,sets,file
```

to

```
      call iopar(iofile,datin)
      read(iofile,*)yes,mi,ma,sets,file
```

Note that subroutine INPAR does not need to be changed.

Library programs (IMSL)

The program may need the IMSL program library (*). However, if IMSL is not available the program can still be used if some modifications are made:

----

*) IMSL: International Mathematical & Statistical Libraries, Inc.
   Address: IMSL
   Customer Relations
   Sixth Floor, NBC Building
   7500 Bellaire Boulevard
   Houston, Texas 77036-5085
   USA
   Telephone: (713) 722-1927
   Telex: 79-1923 IMSL INC HOU

(1) Without IMSL no cubic spline interpolation is possible. Ignore program directive 'cub'. Remove statement 'call obscub' in main program 'fit'. Subroutine 'obscub' is not needed then. - Alternatively one may write a dummy subroutine named 'icsscu' replacing the IMSL subroutine.

(2) Without IMSL one cannot choose between two parameter estimation algorithms. Do not set parameter 'prog' to 2 in program directive 'fit' (see section IV). Remove the program section called 'IMSL part' in subroutine 'schfit'. - Alternatively one can write a dummy subroutine named 'zxssq' replacing the IMSL subroutine.

Program structure

The program consists of four categories of routines:

(1) Main program; its only purpose is to call in various subroutines.

(2) Special purpose subroutines; they are called directly by the main program and are especially designed for the parameter estimation and related tasks.

(3) General purpose subroutines; they also may serve as subroutines for other than the parameter estimation (main-) program.

(4) Library programs; they are comparable to the general purpose subroutine but they come from 'outside', i.e. they are more or less 'black boxes'.

Figure 5/1 will help to understand the structuring of the program.

Hints for installing new model schedules

Model schedules are computed in subroutine 'func'. So 'func' is the right place to install new model schedules. Some additional information is to be stored in the function subprograms MSNAM (name of model schedule), NOP (number of parameters), PARNAM (names of parameters), DEFPAR (default parameter values), and MK (definition of 'level'-parameters). Note that the actual program versions may include additional model schedules not mentioned in figure 2/1 that should be considered as test versions.

Figure 5/1: Program Structure

```
main            special purp.   general purp. & library
----------------------------------------------------------------

-fit            -inpar
                -inobs          -inpar

                -obscub         -cubic  -icsscu*
                -obsnor         -err

                -schcal         -funcal -func
                -schfit         -zxssq*         -resid  -func
                                -ident  -...    -resid  -func
                                -resid  -func
                                -func
                                -gof
                -schcut
                -schnor         -func
                -schpro         -mimax
                                -minim  -mimax2 -func
                                                -mimax
                                -maxim  -mimax2 -func
                                                -mimax
                -mmspro

                -out
                -outplo         -plodat
                -outgra         -graph  -graini
                                        -gradra
                                        -grapri
                -outgof         -gof
                -outsum
                -outsta         -stat
```

) routines marked with a '' come from the IMSL library

Figure 5/1:   continued

```
----------------------------------------------------------------

...           -ident  -funct  -resid  -func
                      -functa
                      -mxout  -loc
                      -err
                      -dir    -matr
                              -mxout  -loc
                              -array
                      -dir1   -matr
                              -mxout  -loc
                              -array
                              -minv   -err
                      -sergo  -matr
                              -array
                              -minv   -err
                              -ort    -matr
                              -step   -funct  -resid  -func
                                              -constr
                              -err
                              -detreq -matr
                                      -minv   -err
                      -subpq  -array
                              -minv   -err
                              -matr
```

APPENDIX A: COMPUTER PROGRAMS: MAIN PROGRAM AND SPECIAL PURPOSE SUBROUTINES (BY ORDER OF EXECUTION)

```
c MAIN-FIT   -  main program to fit model schedules to observed data
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       input:
c
c       (1) parameter cards
c           (see subroutine INPAR)
c
c       (2) one or two observation data set(s)
c           if the observed data are rates one data set is required
c           if the observed data are not rates two data sets are required:
c            - the first data sets has to contain population data
c            - the second data set has to contain the move/transition data
c           (see subroutine INDAT)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        program fit
c      'call userarg' is UNIX-specific and has to be removed on other systems
        call usearg
c
c          READ INPUT
c
1000    call inpar(end)
        call inobs
        if(end.lt.0.)goto 1000
c
c          MODIFY INPUT      (remove 'call obscub' if IMSL is not available!)
c
        call obscub
        call obsnor
c
c          CALCULATE MODEL SCHEDULE / FIT MODEL SCHEDULE TO OBSERVED DATA
c
        call schcal
        call schfit
c
```

```
c             MODIFY SCHEDULE, COMPUTE PROPERTIES
c
              call schcut
              call schnor
              call schpro
              call mmspro
c
                 PRINT OUTPUT
c
              call out
              call outplo
              call outgof
              call outgra
              call outsum(end)
              call outsta(end)
c
2000          continue
c
              if(end.ne.1.)goto 1000
              stop
              end
```

```
c IN-PAR    -     reads parameter cards
c                 IN-PAR calls no subroutines
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   purpose
c       store parameter cards for further use
c
c   usage
c       call inpar(end)
c
c   parameter
c       end     - is set to 1. when end of information is reached (output)
c
c   method
c       parameter cards are read from unit 5 in free order;
c       their general format is
c               nam,par1,par2,par3,...,parn
c       by the parameter card identifier 'nam' IN-PAR identifies the
c       parameter card and stores it for further use; the parameters
c       are read later by the subroutine where they are required;
c       the reading stops when (1) end of information is reached or
c       (2) nam is 'end'; if two or more parameter cards have the
c       same identifier the former parameter card is overwritten.
c
c   valid parameter card identifiers
c       cut     - parameters read by SCH-CUT (stored at 'cutin')
c       con     - read by IN-PAR (no parameters)
c       cop     - parameters read by IN-PAR
c       dat     - parameters read by IN-OBS (stored at 'datin')
c       cub     - parameters read by OBS-CUB (stored at 'cubin')
c       nor     - parameters read by OBS-NOR & SCH-NOR (stored at 'norin')
c       max     - parameters read by SCH-FIT (stored at 'maxin')
c       min     - parameters read by SCH-FIT (stored at 'minin')
c       mod     - parameters read by SCH-CAL & SCH-FIT (stored at 'modin')
c       fit     - parameters read by SCH-FIT (stored at 'fitin')
```

```
c        fix      - parameters read by SCH-FIT (stored at 'fixin')
c        out      - parameters read by OUT & OLT-PLO (stored at 'outin')
c        gra      - parameters read by OUT-GRA (stored at 'grain')
c        end      - parameters read by IN-PAR
c        ---      - parameters read by IN-PAR (same as 'end')
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
         subroutine inpar(endmk)
         character mark*3,zero*130,ignore*1
         character*130 line,cubin,fitin,fixin,modin,norin,cutin
         ,proin,datin,outin,grain,genin,minin,maxin
         common/ccon/cont
         common/ccubin/cubin
         common/cfitin/fitin
         common/cfixin/fixin
         common/cmodin/modin
         common/cmaxin/maxin
         common/cminin/minin
         common/cnorin/norin
         common/ccutin/cutin
         common/cproin/proin
         common/cdatin/datin
         common/coutin/outin
         common/cgrain/grain
         common/cgenin/genin
         if(endmk.ne.0.)goto 9998
         nor=nor+1
         write(6,'("1"/" start run",i2/1x)')nor
         if(nor.gt.1)goto 1000
         do 050 i=1,130,2
050      zero(i:i)='0'
         cubin=zero
         fitin=zero
         fixin=zero
         modin=zero
         maxin=zero
```

```
         minin=zero
         norin=zero
         cutin=zero
         proin=zero
         datin=zero
         outin=zero
         grain=zero
         genin=zero
1000     read(5,'(a3,a1,a128)',end=9999)mark,ignore,line(3:130)
         if(mark(1:1).eq.'*'.or.mark(1:1).eq.'#')goto 1000
         line(1:2)='1 '
         if(ignore.eq.'x'.or.ignore.eq.'-'.or.ignore.eq.'*')line(1:2)='0 '
         do 100 i=52,130,2
100      if(line(i-1:i).eq.'  ')line(i:i)='0'
         if(mark.ne.'con')goto 2000
         read(line(3:130),*)cont
         if(cont.eq.0.)cont=1.
         write(6,'(" set control mode"/)')
         goto 1000
2000     if(mark.ne.'cub')goto 2001
         cubin=line
         goto 1000
2001     if(mark.ne.'fit')goto 2002
         fitin=line
         goto 1000
2002     if(mark.ne.'fix')goto 2003
         fixin=line
         goto 1000
2003     if(mark.ne.'mod')goto 2004
         modin=line
         goto 1000
2004     if(mark.ne.'nor')goto 2005
         norin=line
         goto 1000
2005     if(mark.ne.'pro')goto 2006
         proin=line
         goto 1000
```

```
2006     if(mark.ne.'cut')goto 2007
         cutin=line
         goto 1000
2007     if(mark.ne.'gra')goto 2008
         grain=line
         goto 1000
2008     if(mark.ne.'dat')goto 2009
         datin=line
         goto 1000
2009     if(mark.ne.'out')goto 3000
         outin=line
         goto 1000
3000     if(mark.ne.'---'.and.mark.ne.'end')goto 4000
         return
4000     if(mark.ne.'cop')goto 5000
         read(line(3:130),*)nol,lol,ifil1,ifil2
         if(lol.eq.0)lol=60
         if(ifil1.eq.0)ifil1=5
         if(ifil2.eq.0)ifil2=6
         do 400 i=1,nol
         read(ifil1,'(a)')line(1:lol)
         write(ifil2,'(a)')line(1:lol)
400      continue
         goto 1000
5000     if(mark.ne.'min')goto 5001
         minin=line
         goto 1000
5001     if(mark.ne.'max')goto 5002
         maxin=line
         goto 1000
5002     continue
6000     genin=line
         goto 1000
9998     write(6,'(///" end")')
9999     endmk=1.
         return
         end
```

```
c IN-OBS    -     reads input data (observations) for curve fitting progr
c                 IN-OBS calls IN-DAT
c
        subroutine inobs
        character tit*60,tit2*60,datin*130
        integer file
        real mig(101)
        common/ctit/tit
        common/cage/low,ihi,mi,ma,iv
        common/cpop/pop(101)
        common/cobs/x(101)
        common/cgro/grx,grx1,gry
        common/cdatin/datin
c
c          READ OBSERVATIONS (RATES OR POPULATION + MOVE/TRANSITION DATA)
c
c        call iopar(iofile,datin)
        read(datin,*)yes,mi,ma,sets,file
        mi=mi+1
        ma=ma+1
        if(file.eq.0)file=1
        ier=0
 1000   call indat(file,tit,low,ihi,iv,x,ier)
        if(ier.gt.0)goto 9999
        if(sets.eq.1.)goto 4000
        do 100 i=low,ihi
 100    if(x(i).gt.0..and.x(i).lt.1.)goto 4000
        do 200 i=1,101
 200    pop(i)=x(i)
        if(ier.eq.0)ier=1
        call indat(file,tit2,low,ihi,iv,mig,ier)
        if(ier.gt.0)goto 4000
        tit=tit2
        do 300 i=1,101
 300    if(x(i).ne.0.)x(i)=mig(i)/x(i)
        write(6,62)tit
```

```
            goto 5000
      4000  write(6,61)tit
      c
      c        SET FIELD INDICES MI, MA   (USING INPUT AGES MI, MA)
      c
      5000  if(mi.eq.1.and.ma.eq.1)mi=low
            if(ma.le.mi)ma=ihi
            if(ma.lt.ihi.and.ma.gt.ihi-iv)ma=ma+iv
            write(6,'(11x,"age range:",i5," -",i3)')mi-1,ma-1
      c
      c        COMPUTE GROSS RATE
      c
            grx=vsum(x,101,mi,ma,1)
            grx1=grx
            if(grx.lt.100.)write(6,63)grx
            if(grx.ge.100.)write(6,64)grx
            return
      9999  low=1
            ihi=101
            iv=1
            if(ier.gt.1.or.grx.eq.0.)goto 4000
            rewind file
            goto 1000
      61    format(//' read data:   ',a60//11x,'kind of data:   rates'
         .         ,' (or population data)')
      62    format(//' read data:   ',a60//11x,'kind of data:   population +'
         .         ,' move/transition data')
      63    format(11x,'gross rate:',f9.3)
      64    format(11x,'sum:',f16.3)
            end
```

```
c OBS-CUB   -   interpolates observed rates by cubic spline method
c               OBS-CUB calls CUBIC
c               ! do not use OBS-CUB if IMSL is not abvailable !
c
          subroutine obscub
          character cubin*130
          real xn(101)
          common/cobs/x(101)
          common/cage/low,ihi,mi,ma,iv
          common/cgro/grx,grx1,gry
          common/cfun/mod,p(20),y(101),y123(303)
          common/ccubin/cubin
          common/ccon/cont
c
c          call iopar(iofile,cubin)
          read(cubin,*)yes,mi1,ma1,iv1,ic,sm,ier
          if(yes.eq.0.)return
          write(6,o2)
          mi1=mi1+1
          ma1=ma1+1
          if(mi1.eq.1.and.ma1.eq.1)mi1=mi
          if(ma1.eq.1)ma1=ma
          if(iv1.eq.0)iv1=iv
          j1=0
          call cubic(x,xn,101,101,mi1,ma1,iv1,j1,j2,iv*1,ic,sm,ier)
          if(ier.ne.0)return
c
c            SET NEW AGE INCICES, CALCULATE GOCCNESS OF FIT
c            SET x TO NEW VALUES xn, CALCULATE NEW GROSS RATE grx1
c
          if(mi1.eq.mi)mi=j1
```

```
          if(ma1.eq.ma)ma=j2
          iv=1
          do  200 i=1,101
  200     y(i)=0.
          do 202 i=low,ihi
  202     y(i)=x(i)
          grx1=0.
          do 204 i=j1,j2
          if(xn(i).lt.0.)xn(i)=0.
          x(i)=xn(i)
  204     grx1=grx1+x(i)
          write(6,63)vgof(x,y,101,mi,ma,iv),grx1
          return
c
   62     format(//' interpolate observed rates (cubic spline)')
   63     format(/11x,'goodness of fit of cubic spline:',f7.1
         */11x,'new gross rate (cubic spline curve):',f9.3)
          end
```

```
c OBS-NOR    -    normalizes observed rates (gross rate will be xnorm)
c                 OBS-NOR may call ERR
c
        subroutine obsnor
        character norin*130
        common/cage/low,ihi,mi,ma,iv
        common/cobs/x(101)
        common/cgro/grx,grx1,gry
        common/cnorin/norin
c
c        call iopar(iofile,norin)
        read(norin,*)yes,xnorm,ynorm,ier
        if(yes.eq.1.and.xnorm.eq.0.)return
        write(6,6)
        if(xnorm.eq.0.)xnorm=1.
        grx1=vsum(x,101,mi,ma,1)
        write(6,62)grx1
        if(grx1.eq.0.)goto 9999
        do 200 i=low,ihi
200     x(i)=x(i)*xnorm/grx1
        grx1=vsum(x,101,mi,ma,1)
        write(6,63)grx1
        return
c
9999    call err('obsnor',200,200,'grx1',0,0,grx1)
        return
6       format(//' normalize observed rates'/)
62      format(11x,'old gross rate (observations):',f9.3)
63      format(11x,'new gross rate (observations):',f9.3)
        end
```

```
c SCH-CAL   -    calculates model schedules
c                SCH-CAL calls FUN-CAL
c
          subroutine schcal
          character modin*130,msnam*60
          real y(1),p(20)
          common/cgro/grx,grx1,gry
          common/cmodin/modin
c          call iopar(iofile,modin)
          read(modin,*)yes,model,p,ier
          if(yes.ne.1.)return
          write(6,'(//" calculate ",a60/)')msnam(model)
          n=nop(model)
          if(n.eq.0)goto 2000
          do 100 i=1,n
  100     if(p(i).ne.0.)goto 2000
          write(6,'(11x,"use default parameter values")')
          do 105 i=1,n
  105     p(i)=defpar(model,i)
c         write(modin(5:130),'(13(1x,f8.5))')(p(i),i=1,n)
 2000     if(model.eq.11.and.grx.ne.0.)p(9)=p(9)*grx1/grx
          call funcal(model,p,20,y,1,.5,100.5,1.,ier)
          return
          end
```

```
c SCH-FIT    -    fits model schedule to observed data, calculates model schedule
c                 SCH-FIT calls ZXSSQ (IMSL) or IDENT (VNIISI), RESID, FUNC & SCH-GOF
c
        subroutine schfit
        character msnam*60,fitin*130,fixin*130,minin*130,maxin*130
        integer ii(20)
        real q(20),r(101)
c THE NEXT CARD IS NEEDED ONLY FOR ident (VNIISI)
        real delta1(20),epsi(20),pksi(101),pq(20,20),q00(20)
c THE NEXT 2 NON-COMMENT CARDS ARE NEEDED ONLY FOR zxssq (IMSL)
c MINIMUM DIMENSIONS:    work: 5*n+2*nov+(n+1)*n/2
c                        xjac: n*nov
c                        xjtj: (n+1)*n/2
        real work(512),xjac(2020),xjtj(210),ff(101),parm(4)
        external resid
        common/cage/low,ihi,mi,ma,iv
        common/cobs/x(101)
        common/cfun/model,p(20),y(101),y1(101),y2(101),y3(101)
        common/cfitin/fitin
        common/cfixin/fixin
        common/cminin/minin
        common/cmaxin/maxin
        common/cres/res,pr,iter,p0(20),fix(20),adj(20),pmin(20),pmax(20)
        data eps/.1/,alb/.8/,np/2/,isub/0/,nnp/20/,epsfi/1./,amu/0./
        data vers/.2/
c
c          COMMON PART (INITIALIZING)
c
        iter=0
        infer=0
c        call iopar(iofile,fitin)
        read(fitin,*)fityes,prog,res,maxfn,nsig,eps,delta,xdelta,op,ier
c        call iopar(iofile,fixin)
        read(fixin,*)fixyes,fix
c        call iopar(iofile,minin)
        read(minin,*)minyes,pmin
c        call iopar(iofile,maxin)
```

```
      read(maxin,*)maxyes,pmax
      if(fityes.eq.0..and.fixyes.eq.0.
     .and.minyes.eq.0..and.maxyes.eq.0.)return
      write(6,61)msnam(model)
      if(prog.eq.0.)prog=1.
      if(res.ge.10.)pr=1.
      if(res.ge.10.)res=res-10.
      if(maxfn.eq.0)maxfn=1000
      if(op.eq.0..and.prog.eq.2.)op=1.
      nov=ma-mi+1
      n=0
      do 100 i=1,nop(model)
      p0(i)=p(i)
      if(fix(i).eq.1.)goto 100
      n=n+1
      ii(n)=i
      q(n)=1.
      adj(i)=p(i)
      if(minyes.eq.0.)pmin(i)=0.
      if(maxyes.eq.0.)pmax(i)=100.
      if(op.ne.1.)goto 100
      adj(i)=1.
      q(n)=p(i)
      if(q(n).eq.0.)q(n)=.000001
100   continue
1000  if(maxfn.le.1.or.n.eq.0)goto 9000
      if(nsig.eq.0)nsig=3
      if(prog.eq.2.)goto 3000
c
c        VNIISI PART (CALLING ident)
c
      write(6,63)prog+vers
      ipr=ier
      ier=0
      xepsi=10.**(-nsig)
      if(xdelta.eq.0.)xdelta=.01
      do 200 i=1,n
```

```
           xq=p(ii(i))
           if(xq.eq.0.)xq=.0000001
           epsi(i)=xepsi*q(i)
           delta1(i)=xdelta*q(i)
           pq(i,i)=xdelta*q(i)*q(i)
           q00(i)=q(i)
           if(q00(i).ne.0.)isub=1
      200  continue
           if(ipr.eq.0)ipr=9
           do 205 i=1,nov
      205  pksi(i)=1.
           h0=.1
           call ident(q,n,delta1,epsi,eps,nov,maxfn
           ,ipr,pksi,h0,alb,np,pq,q00,isub,nnp,nfi,epsfi,amu)
           if(h0.gt.0.)infer=h0
           goto 8000
c
c             IMSL PART (CALLING zxssq - remove this part if IMSL is not available!)
c
      3000 write(6,62)prog+vers
           ixjac=nov
           if(eps.eq.0.)eps=.000001
           if(delta.eq.0.)delta=.000001
           call zxssq(resid,nov,n,nsig,eps,delta
            ,maxfn,iopt,parm,q,ssq,ff,xjac,ixjac,xjtj,work,infer,ier)
           if(ier.gt.0.and.ier.ne.133)write(6,69)ier
           goto 8000
c
c             COMMON PART (FINISHING WORK)
c
      8000 nsig=0
      9000 iter=iter-1
           call resid(q,nov,n,r)
           write(6,64)infer,iter
```

```
              call func(1,101,1,.5,1.)
              call gof(x,y,101,101,mi,ma,iv,sem,se,sst,ssq,chi,xm,ym,ier)
              write(6,65)sem,ssq
c
61            format(//' fit ',a60/)
62            format(11x,'use IMSL fit routine   (program version',f4.1,')')
63            format(11x,'use VNIISI fit routine   (program version',f4.1,')')
64            format(11x,'stop criterion',i2,' met'
     .        ,' - fit process stopped after',i5,' iterations')
65            format(11x,'goodness of fit (model schedule):',f5.1
     .        ,'   -   ssq:',f10.6)
69            format(11x,'---ERROR:   schfit-ier =',i4/)
              end
```

```
c SCH-CUT    -    cuts model schedule (eventualy making last age group open ended)
c                 SCH-CUT calls no subroutines
c
          subroutine schcut
          character cutin*130
          real pop(101)
          common/cage/low,ihi,mi,ma,iv
          common/cgro/grx,grx1,gry
          common/cfun/mod,p(20),y(101),y123(303)
          common/ccutin/cutin
c
c          call iopar(iofile,cutin)
          read(cutin,*)yes,mi1,ma1,opnend,maxage
          if(yes.ne.1.)return
          mi1=mi1+1
          ma1=ma1+1
          if(mi1.eq.1.and.ma1.eq.1.and.opnend.eq.0.)opnend=1
          if(ma1.eq.1)ma1=ma
          write(6,60)mi1-1,ma1-1
          if(opnend.eq.0)goto 5000
c
c            MAKE LAST AGE GROUP OPEN-ENDED
c
          write(6,
          '(11x,"age",i3," is open ended   (method=",i1,")")
          )ma1-1,int(opnend)
          maxage=maxage+1
          if(maxage.eq.1)maxage=101
          if(maxage.le.ma1)return
          pop(ma1)=1.
          rpop=1.
          if(opnend.eq.2)goto 2000
          if(opnend.eq.3)goto 1000
          if(mod.lt.10.or.mod.ge.20)goto 2000
```

```
c
1000    do 100 i=ma1+1,maxage
        if(y(i).gt.1.)goto 1005
        pop(i)=pop(i-1)*(1.-y(i))
        rpop=rpop+pop(i)
100     continue
1005    maxage=i-1
        goto 3000
c
2000    decr=1./(maxage-ma1)
        do 200 i=ma1+1,maxage
        pop(i)=pop(i-1)-decr
        rpop=rpop+pop(i)
200     continue
c
3000    sum=0.
        do 300 i=ma1,maxage
        sum=sum+y(i)*pop(i)/rpop
300     y(i)=0.
        y(ma1)=sum
c
c          CUT MODEL SCHEDULE
c
5000    do 500 i=1,mi1-1
500     y(i)=0.
        do 502 i=ma1+1,101
502     y(i)=0.
        gry=vsum(y,101,mi,ma,iv)
        write(6,'(11x,"new gross rate (model schedule):",f9.3)')gry
        return
60      format(//' cut model schedule'
        /11x,'new age range:',i4,' -',i3)
        end
```

```fortran
c SCH-NOR    -   normalizes model schedule (gross rate will be ynorm)
c                SCH-NOR calls SCH-CAL
c
        subroutine schnor
        character norin*130
        common/cage/low,ihi,mi,ma,iv
        common/cfun/model,p(20),y(101),y123(303)
        common/cgro/grx,grx1,gry
        common/cnorin/norin
c
c        call iopar(iofile,norin)
        read(norin,*)yes,xnorm,ynorm,ier
        if(yes.eq.1.and.ynorm.eq.0.)return
        gry=vsum(y,101,mi,ma,1)
        if(gry.eq.0.)return
        if(ynorm.eq.999)ynorm=grx1
        if(ynorm.eq.0.)ynorm=xnorm
        if(ynorm.eq.0.)ynorm=grx1
        if(ynorm.eq.0.)ynorm=1.
        write(6,61)
        write(6,62)gry
        mkcnt=0
        do 200 i=1,20
        if(mk(model,i).ne.1)goto 200
        p(i)=p(i)*ynorm/gry
        mkcnt=mkcnt+1
 200    continue
        if(mkcnt.eq.0)write(6,64)
        do 300 i=1,101
 300    y(i)=y(i)*ynorm/gry
        gry=vsum(y,101,mi,ma,1)
        write(6,63)gry
        return
 61     format(//' normalize model schedule'/)
 62     format(11x,'old gross rate (model schedule):',f9.3)
 63     format(11x,'new gross rate (model schedule):',f9.3)
 64     format(11x,'*** schedule parameters not normalized ***')
        end
```

```
c SCH-PRO    -    computes properties of model schedule y
c                 SCH-PRO calls MIMAX, MINIM & MAXIM
c
          subroutine schpro
          real yy(101),yy1(101),yy2(101),yy3(101)
          common/cage/low,ihi,mi,ma,iv
          common/cfun/model,p(20),y(101),y1(101),y2(101),y3(101)
          common/cmim/xmin(6),ymin(6),xmax(6),ymax(6)
          common/cpro/xm,pc1,pc2,pc3,xl,xh,xr,x,b
          common/cmms/yh,ratio(6)
c
2000      continue
          do 200 i=1,101
          yy(i)=y(i)
          yy1(i)=y1(i)
          yy2(i)=y2(i)
          yy3(i)=y3(i)
200       continue
          xm=0.
          do 300 i=mi,ma
300       xm=xm+(i-1.+.5*iv)*yy(i)
          pc1=0.
          pc2=0.
          pc3=0.
          do 401 i=mi,15
401       pc1=pc1+yy(i)
          do 402 i=16,65
402       pc2=pc2+yy(i)
          do 403 i=66,ma
403       pc3=pc3+yy(i)
```

```
      pc=pc1+pc2+pc3
      if(pc.eq.0.)return
      xm=xm/pc
      pc1=100.*pc1/pc
      pc2=100.*pc2/pc
      pc3=100.*pc3/pc
c
      if(model.lt.20)goto 6000
      call mimax(y,mi,ma,iv)
      xl=xmin(2)-1.+.5*iv
      yl=ymin(2)
      xh=xmax(2)-1.+.5*iv
      yh=ymax(2)
      if(model.eq.21.or.model.eq.22)goto 5005
      xr=xmax(3)-1.+.5*iv
      if(xl-iv.lt.0.)xl=0.
      if(xr.lt.xh)xr=0.
      if(xl.gt.0.)call minim(xl,yl,iv)
      call maxim(xh,yh,iv)
      if(xr.gt.xh)call maxim(xr,yr,iv)
5005  x=xh-xl
      b=yh-yl
6000  continue
      do 600 i=1,101
      y(i)=yy(i)
      y1(i)=yy1(i)
      y2(i)=yy2(i)
      y3(i)=yy3(i)
600   continue
      return
```

```
c MMS-PRO   -   calculate model migration schedule (mms) properties
c               MMS-PRO calls no subroutines
c
        subroutine mmspro
        common/cage/low,ihi,mi,ma,iv
        common/cfun/model,p(20),y(101),y123(303)
        common/cpro/xm,pc1,pc2,pc3,xl,xh,xr,x,b
        common/cmms/a,ratio(6)
        if(model.ne.32)return
c
c           MMS RATIOS
c
        ratio(1)=rate(p(1),p(11))
        ratio(2)=rate(p(1),p(3))
        ratio(3)=rate(p(7),p(3))
        ratio(4)=rate(p(2),p(5))
        ratio(5)=rate(p(6),p(5))
        ratio(6)=rate(p(10),p(9))
c
c           PARENTAL SHIFT a
c
        yh=a
        ixl=xl+1
        ix=xh+1
        ixr=xr+1
        if(ixr.lt.ix)ixr=ma
        k=0
        a=0.
        do 700 i=1,ixl
```

```
      if(y(i).gt.yh.or.ix.lt.1)goto 700
      do 705 j=ix,ixr
      if(j.eq.1)goto 705
      if(y(j).le.y(i))goto 7000
705   continue
      goto 8000
7000  k=k+1
      if(y(j-1)-y(i).lt.y(i)-y(j))j=j-1
      a=a+(j-i)
      ix=j
700   continue
8000  if(k.gt.0)a=a/k
c
      return
      end
c
c RATE  -   function computing x/y
c
      function rate(x,y)
      rate=0.
      if(y.ne.0.)rate=x/y
      return
      end
```

```
c OUT    -   prints parameter table, obs. & est. rates and internal summ. inf.
c             OUT calls no subroutines
c
         subroutine out
         character mark(20)*7,tit*o0,star*1,resid*3,parnam*8
         character*130 cubin,fitin,fixin,modin,outin
         common/ctit/tit
         common/cage/low,ihi,mi,ma,iv
         common/cobs/x(101)
         common/cpop/pop(101)
         common/cfun/model,p(20),y(101),y1(101),y2(101),y3(101)
         common/cgro/grx,grx1,gry
         common/cres/res,pr,iter,p0(20),fix(20),a(20),pmin(20),pmax(20)
         common/ccubin/cubin
         common/cfitin/fitin
         common/cfixin/fixin
         common/cmodin/modin
         common/coutin/outin
         common/ccon/cont
c
c         call iopar(iofile,outin)
         read(outin,*)yes,ifil1,ifil2,ifil3,ifil4,ifil5,ifil6,ifil7
         write(6,'(//" print results"/)')
         if(yes.eq.1.)goto 1002
         if(ifil1.eq.0)ifil1=6
         if(ifil2.eq.0)ifil2=6
c         call iopar(iofile,outin)
1002     read(cubin,*)smooth
c         call iopar(iofile,fitin)
         read(fitin,*)yes,prog
c         call iopar(iofile,fixin)
         read(fixin,*)yes,fix
c         call iopar(iofile,modin)
         read(modin,*)yes,model0,p0
         if(ifil1.gt.0)write
```

```
      .  (6,'(11x,"unit",i2,":   parameter table")')ifil1
         if(ifil2.gt.0)write
      .  (6,'(11x,"unit",i2,":   observed & estimated rates")')ifil2
         if(ifil3.gt.0)write
      .  (6,'(11x,"unit",i2,":   internal summary information")')ifil3
         if(ifil4.gt.0)write
      .  (6,'(11x,"unit",i2,":   rates (for use as plot input)")')ifil4
         if(ifil5.gt.0)write
      .  (6,'(11x,"unit",i2,":   summary table(s)")')ifil5
         if(ifil6.gt.0)write
      .  (6,'(11x,"unit",i2,":   estimated moves/transitions")')ifil6
         if(ifil7.gt.0)write
      .  (6,'(11x,"unit",i2,":   goodness-of-fit table")')ifil7
         sem=vgof(x,y,101,mi,ma,iv)
         if(ifil1.eq.0)goto 4000
         gry=vsum(y,101,mi,ma,1)
         do 200 i=1,20
         mark(i)='        '
         if(fix(i).eq.1.)mark(i)='(fixed)'
 200     continue
         if(res.eq.0.)resid='rel'
         if(res.eq.1.)resid='abs'
         if(res.eq.2.)resid='log'
         write(ifil1,60)tit,model,grx,mi-1,ma-1,gry,resid
         do 300 i=1,nop(model)
         write(ifil1,61)i,parnam(model,i),p(i),mark(i),p0(i)
 300     continue
         write(ifil1,62)sem,iter
         if(ier.gt.0.and.ier.ne.133)write(ifil1,69)ier
 4000    adj=1.
         if(grx.ne.grx1)adj=grx
         if(ifil2.eq.0)goto 5000
         write(ifil2,66)tit,model
         do 400 i=1,10*((max0(ma,ihi)-1)/10+1)
         if(i.gt.101)goto 5000
         star='*'
```

```
            if(i.lt.mi.or.i.gt.ma)star=' '
            pcdev=.0
            if(x(i).ne.0.)pcdev=(y(i)-x(i))*100./x(i)
            if(abs(pcdev).ge.99.95)pcdev=pcdev*abs(99.9/pcdev)
            if(abs(y(i)).ge.99.95)y(i)=y(i)*abs(99.9/y(i))
            if(mod(i,10).eq.1)write(ifil2,6)
            write(ifil2,6)i-1,x(i)*adj,y(i)*adj,x(i),y(i)
     .               ,y(i)-x(i),pcdev,star
  400       continue
 5000       if(sem.gt.66.667)prog=900+prog
            if(ifil3.gt.0)
     .      write(ifil3,7)tit,grx,gry,sem,iter,model,prog,(p(i),i=1,13)
            if(ifil6.gt.0)write(ifil6,'(9f8.0)')(pop(i)*y(i)*adj,i=1,ma)
            return
 6          format(1x,i9,f20.6,f15.6,f20.6,f15.6,f20.6,f15.1,10x,a1)
 7          format('--- ',a60,3f10.6,2i10,10x,f10.6/4x,13f10.6)
 60         format('1'/' ',a47,'(model',i3,' used)'///
     .      /' observed gross rate =',f10.6,16x,'age range:',i3,'-',i2
     .      /' schedule gross rate =',f10.6,16x,a3,' resid.'
     .      //' estimated parameters:',26x,'(initial values)'/1x)
 61         format(1x,i7,3x,a8,2x,f11.6,3x,a7,10x,f11.6)
 62         format(///' goodness of fit =',f7.3
     .      //' n of iterations =',i7/1x)
 66         format('1',a60,50x,' (model',i3,' used)'
     .       //7x,'AGE',12x,'OBSERVED',6x,'ESTIMATED'
     .       ,12x,'OBSERVED',6x,'ESTIMATED'
     .       ,11x,'EST - OBS',3x,'100(EST-OBS)'
     .       /26x,'(not normalized)',20x,'(normalized)'
     .       ,18x,'(norm.)',12x,'/OBS')
 69         format(//'  ZXSSQ-error ',i3)
            end
```

```
c OUT-PLO    -    prints observed and schedule rates as PLOT-input
c                 OUT-PLO calls PLO-DAT
c
        subroutine outplo
        character*130 outin
        common/cage/low,ihi,mi,ma,iv
        common/cobs/x(101)
        common/cfun/model,p(20),y(101),y1(101),y2(101),y3(101)
        common/coutin/outin
c        call iopar(iofile,outin)
        read(outin,*)yes,if1,if2,if3,if4
        if(if4.eq.0)return
        call plodat(x,y,y1,y2,y3,1,max0(ihi,ma),1,-1,if4,6)
        return
        end
```

```
c OUT-GOF    -    computes goodness-of-fit table
c                 OUT-GOF calls GOF
c
      subroutine outgof
      character outin*130,opein*130,age*6,tit*60
      common/ctit/tit
      common/cage/low,ihi,mi,ma,iv
      common/cobs/x(101)
      common/cfun/mod,p(20),y(101),y123(303)
      common/coutin/outin
      common/copein/opein
c
      read(outin,*)yes,i1,i2,i3,i4,i5,i6,ifil7
      if(ifil7.le.0)return
      read(opein,*)open
      write(ifil7,60)tit
      mi1=5*((mi-1)/5)+1
      ma1=5*((ma-1)/5)+1
      do 100 k=mi1,ma1+5,5
      kmi=k
      kma=k+4
      if(k.eq.mi1.or.k.gt.ma)kmi=mi
      if(k.eq.ma1.or.k.gt.ma)kma=ma
      call gof(x,y,101,101,kmi,kma,1,sem,se,sst,ssq,chi,xm,ym,ier)
      d=ym-xm
```

```
        dpc=0.
        if(xm.ne.0.)dpc=100.*d/xm
        if(xm.gt.99.)xm=99.
        if(ym.gt.99.)ym=99.
        if(d.gt.99.)d=99.
        if(d.lt.-99.)d=-99.
        if(dpc.gt.99.)dpc=99.
        if(dpc.lt.-99.)dpc=-99.
        write(age,'(1x,i2,"-",i2)')kmi-1,kma-1
        if(k.eq.ma1.and.open.eq.1.)age(4:6)='+
        if(k.gt.ma)age='-total'
        write(ifil7,6)age,xm,ym,d,dpc,se,sem,ssq
100     continue
        return
6       format(a6,3f10.6,f6.1,f10.6,f6.1,f10.6)
60      format('1'//1x,a60///' Goodness-of-Fit Table'
     .  //'    age       mean       mean      diff. diff%'
     .  ,'  abs.err. mae%m        ssq'
     .  /' group  obs.rate  est.rate'/)
        end
```

```
c OUT-GRA    -    prints graphic
c                 OUT-GRA calls GRAPH
c
      subroutine outgra
      character tit*60,grain*130
      common/ctit/tit
      common/cobs/x(101)
      common/cfun/mod,p(20),y(101),y123(303)
      common/cgrain/grain
c
c      call iopar(iofile,grain)
      read(grain,*)yes,ymin,ymax,lines,icols,ifile,ier
      if(yes.eq.0.)return
      if(ymax.le.ymin)ymax=.1
      if(lines.le.0)lines=50
      if(lines.le.17.and.icols.le.0)icols=75
      if(icols.le.0)icols=100
      if(ifile.le.0)ifile=6
      if(lines.le.17)write(6,6)ifile,ymin,ymax,lines,icols
      call graph(x,y,101,101,ymin,ymax,lines,icols,'01',tit,ifile)
      return
6     format(//' print graphic on unit',i2
     . //11x,'scale ranges from',f10.6,' to',f10.6
     . ,' - page size',i3,' x',i4)
      end
```

```
c OUT-SUM    -    prints summary table
c                 OUT-SUM calls GRA-INI, GRA-DRA & GRA-PRI
c
        subroutine outsum(end)
        character tit*80,msnam*80,parnam*8,outin*130,grain*130
     * ,page(60)*132,char*1
        real store(101,6)
        common/ctit/tit
        common/cage/low,ihi,mi,ma,iv
        common/cgro/grx,grx1,gry
        common/cobs/x(101)
        common/cfun/model,p(20),y(101),y12(202),y3(101)
        common/cpro/xm,pc1,pc2,pc3,xl,xh,xr,xdiff,b
        common/coutin/outin
        common/cgrain/grain
        data page/'1',59*' '/
        data index/0/,m/-20/
        index=index+1
        m=m+22
        n=m+20
        nn=m+15
        write(page(2)(m:m+10),'("schedule",i2)')index
        write(page(4)(m:n),'("gr (obs)  ",f10.6)')grx
        write(page(5)(m:n),'("gr (est)  ",f10.6)')gry
        write(page(6)(m:nn),'("mae%m",5x,f5.1)')vgof(x,y,101,mi,ma,iv)
        do 100 i=1,13
100     write(page(6+i)(m:n),'(a8,f12.6)')parnam(model,i),p(i)
        write(page(20)(m:nn),'("xmean",5x,f5.1)')xm
        write(page(21)(m:nn),'("%( 0-14)  ",f5.1)')pc1
        write(page(22)(m:nn),'("%(15-64)  ",f5.1)')pc2
        write(page(23)(m:nn),'("%(65+  )  ",f5.1)')pc3
        write(page(24)(m:nn),'("x low",5x,f5.1)')xl
        write(page(25)(m:nn),'("x high",4x,f5.1)')xh
        write(page(26)(m:nn),'("x high2",3x,f5.1)')xr
        write(page(27)(m:nn),'("x (diff)  ",f5.1)')xdiff
```

```
            write(page(28)(m:n),'("b (y-diff)",f10.6)')b
            write(page(31+index),'(1x,i1":   ",a60,3x,a60)')
           . index,tit,msnam(model)
            do 200 i=1,101
  200       store(i,index)=x(i)
            if(index.lt.6.and.end.eq.0.)return
c
c            call iopar(iofile,outin)
            read(outin,*)yes,ifil1,ifil2,ifil3,ifil4,ifil5
            if(yes.eq.0..or.ifil5.eq.0)goto 9999
            write(ifil5,'(a130)')page
            do 900 i=2,60
            do 900 j=1,132,12
  900       page(i)(j:j+11)='            '
c
            read(grain,*)yes,ymin,ymax,lines,icols,ifile
            if(yes.eq.0.)goto 9999
            if(ymax.le.ymin)ymax=.1
            if(lines.eq.0)lines=50
            if(icols.eq.0.and.lines.le.17)icols=75
            if(icols.eq.0)icols=100
            if(ifile.le.0)ifile=6
            call graini(ymin,ymax,lines,icols,'summary')
            do 905 j=1,index
            do 907 i=1,101
  907       y3(i)=store(i,j)
            write(char,'(i1)')j
            call gradra(y3,101,char)
  905       continue
            call grapri(ifile)
c
 9999       index=0
            m=-20
            return
            end
```

```
c OUT-STA    -    prints summary statistics
c                 !!!   only for model schedule 3-2 so far   !!!
c                 OUT-STA calls STAT
c
          subroutine outsta(end)
          parameter (nrun=100,nstat=30)
          character outin*130,col(7,2)*12,row(nstat)*12
          integer nobs(nstat)
          real s(nrun,nstat),xmin(nstat),xmax(nstat),xmea(nstat)
     .    ,xmed(nstat),xmod(nstat),xvar(nstat),xstd(nstat),xstn(nstat)
          common/cage/low,ihi,mi,ma,iv
          common/cgro/grx,grx1,gry
          common/cfun/model,p(20),y0123(404)
          common/cpro/xmpc(4),xlhrs(4),b
          common/cmms/a,ratio(6)
          common/coutin/outin
          data col/'     lowest','     highest',4*' ','   std. dev.'
     .    ,2*'     value',' mean value','      median','        mode'
     .    ,'   std. dev.','      / mean'/
          data row/'gmr (obs)','gmr (mms)','mae%m','a1','alpha1','a2'
     .    ,'mu2','alpha2','lambda2','a3','mu3','alpha3','lambda3','c'
     .    ,'mean age','%( 0-14)','%(15-64)','%(65+  )'
     .    ,'delta1c','delta12','delta23','beta12','sigma2','sigma3'
     .    ,'x low','x high','x ret.','x shift','a','b'/
             if(model.ne.32)goto 2000
          index=index+1
c          call iopar(iofile,outin)
          read(outin,*)ign,i1,i2,i3,i4,ifil5
```

```
            if(index.gt.nrun.or.ifil5.lt.1)return
c
            s(index,1)=grx
            s(index,2)=gry
            s(index,3)=vgof(x,y,101,mi,ma,iv)
            do 100 i=1,11
100         s(index,3+i)=p(i)
            do 102 i=1,4
102         s(index,14+i)=xmpc(i)
            do 104 i=1,6
104         s(index,18+i)=ratio(i)
            do 106 i=1,4
106         s(index,24+i)=xlhrs(i)
            s(index,29)=a
            s(index,30)=b
c
2000        if(end.ne.1.)return
               if(index.lt.1.or.ifil5.lt.1)return
            call stat(s,nrun,nstat,index,0.,nobs
     .      ,xmin,xmax,xmea,xmed,xmod,xvar,xstd,xstn)
            write(ifil5,
     .      '("1"///" mms summary statistics"//2(/13x,7a12)//)'
     .      )col
            do 200 j=1,nstat
200         write(ifil5,'(1x,a12,7f12.5)')row(j)
     .      ,xmin(j),xmax(j),xmea(j),xmed(j),xmod(j),xstd(j),xstn(j)
c
            return
            end
```

APPENDIX B: COMPUTER PROGRAMS: GENERAL PURPOSE SUBROUTINES (BY ALPHABETICAL ORDER)

```
c CUBIC      -     cubic spline interpolating
c                  CUBIC calls IMSL-routine ICSSCU
c                  do not use CUBIC if IMSL is not available !
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c subroutine cubic(a,b,na,nb,i1,i2,i3,j1,j2,ni,ic,sm,ier)
c
c parameters       a          - vector of length na containing the original
c                               data (input)
c                  b          - vector of length nb containing the interpolated
c                               data (output)
c                  na         - number of elements in a (input); 1 < na < 102
c                  nb         - number of elements in b (input); 1 < nb < 102
c                  i1,i2,i3-    indices (input): elements a(i), i=i1,i2,i3
c                               will be interpolated (default: i=1,na,1)
c                  j1,j2      - indices (output, j1 also input):
c                               the interpolated values are stored as
c                               b(j),j=j1,j2; (default: j=i if ni=1 or ni=i3;
c                               j=(i-1)*(ni/i3)+1 else)
c                  ni         - number of interpolation points for each value
c                               a(i), i=i1,i2,i3; if ni=1 each value a(i) is
c                               substituted by only one value b(j), i.e. no
c                               interpolation is done
c                  ic         - ic = 0: vector a values will be cumulated
c                                  = 1: vector a values will not be cumulated
c                  sm         - controls the extent of smoothing (input); see
c                               imsl subroutine icsscu; recommended value: 0.
c                  ier        - error parameter (output)
c                               ier =   1, ni is zero
c                               ier =   3, no default for j1 can be found
c                               ier =   5, j2 is greater than nb
c                               ier = 102, na or nb is greater than 101
c                               ier = 129, ic is less than n-1
c                               ier = 130, n is less than 2
c                               ier = 131, vector xaxis is not ordered
```

```
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      subroutine cubic(a,b,na,nb,ii1,ii2,ii3,jj1,jj2,nni,ic,sm,ier)
      real a(na),b(nb),xaxis(103),yaxis(103),weight(103),yaxnew(103)
     * ,coeff(103,3),work(735)
1000  if(na.gt.101.or.nb.gt.101)ier=102
c
c        SET ICSSCU PARAMETER VALUES
c
      i1=ii1
      i2=ii2
      i3=ii3
      j1=jj1
      j2=jj2
      ni=nni
      icc=103
      if(ni.eq.0)ier=1
      if(i1.eq.0)i1=1
      if(i2.lt.i1.or.i2.gt.na)i2=na
      if(i3.eq.0)i3=1
      if(ier.gt.0)goto 9009
      xaxis(1)=0.
      yaxis(1)=0.
      weight(1)=1.
      n=2
      yaxisl=0.
      do 100 i=i1,i2,i3
      xaxis(n)=xaxis(n-1)+ni
      yaxis(n)=yaxisl+a(i)
      if(ic.ne.1)yaxisl=yaxis(n)
      weight(n)=1.
100   n=n+1
      xaxis(n)=xaxis(n-1)+ni
      yaxis(n)=yaxisl
      weight(n)=1.
```

```
c
1005    call icsscu(xaxis,yaxis,weight,n,sm,yaxnew,coeff,icc,work,ier)
1007    if(ier.ne.0)goto 9010
c
c          CALCULATE INTERPOLATED CURVE work
c
        if(j1.gt.0)goto 2000
        j1=i1
        if(ni.eq.1.or.ni.eq.i3)goto 2000
        if(i3.gt.ni.and.mod(i3,ni).ne.0)ier=3
        if(i3.lt.ni.and.mod(ni,i3).ne.0)ier=3
        if(i3.lt.ni)j1=(i1-1)*(ni/i3)+1
        if(ier.gt.0)goto 9009
2000    j=j1+ni/2
        dd=0.
        if(mod(ni,2).eq.0)dd=.5
        do 200 i=1,n-1
        d=dd
        do 205 k=1,ni
        if(j.gt.0)
     .  work(j)=yaxnew(i)+((coeff(i,3)*d+coeff(i,2))*d+coeff(i,1))*d
        j=j+1
        d=d+1.
205     continue
200     continue
c
c          CALCULATE INTERPOLATED CURVE b
c
        j2=j1+(n-2)*ni-1
        if(j2.gt.nb)j2=nb
        if(ic.eq.1)goto 3005
        do 301 j=j1,j1+ni/2-1
```

```
301     b(j)=(work(j+ni/2+1)-work(j+ni/2))*ni
        do 302 j=j1+ni/2,j2
302     b(j)=work(j+ni)-work(j)
        goto 3009
3005    do 305 j=j1,j2
305     b(j)=work(j+ni)
3009    if(jj1.eq.0)jj1=j1
        jj2=j2
        return
c
9009    write(6,69)ier
        return
9010    write(6,70)ier
        return
69      format(' ---ERROR:   cubic-ier =',i4/)
70      format(' ---ERROR:   icsscu-ier =',i4/)
        end
```

```
c err   -       prints error messages
c
        subroutine err(prog,label1,label2,var,i,j,value)
        character prog*(*),var*(*)
        write(6,*)
     .  ' --- error in',prog,' between label',label1,'and',label2,':'
        if(i.le.0)then
         write(6,*)'      variable',var,' =',value
        else if(j.le.0)then
         write(6,*)'      variable',var,'(',i,') =',value
        else
         write(6,*)'      variable',var,'(',i,',',j,') =',value
        endif
        return
        end
```

```
c FUNC   -       calculates model schedule functions
c                (see also function subprograms DEF-PAR, MK, MS-NAM, NOP & PAR-NAM)
c                FUNC may call ERR
c
      subroutine func(mi,ma,iv,xx,dx)
      real n(101),v(101),mu
      common/cfun/model,p(20),y(101),y1(101),y2(101),y3(101)
      common/cobs/obs(101)
      data n /.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.175,.225,.275
     .        ,.325,.375,.421,.460,.475,.477,.475,.470,.465,.460
     .        ,.455,.449,.442,.435,.428,.420,.410,.400,.389,.375
     .        ,.360,.343,.325,.305,.280,.247,.207,.167,.126,.087
     .        ,.055,.035,.021,.011,.003,51*.0/
      data v /.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0,.0
     .        ,.0,.0,.0,.0,.0,-.004,-.03,-.06,-.10,-.15
     .        ,-.20,-.25,-.31,-.37,-.44,-.52,-.60,-.68,-.76,-.83
     .        ,-.90,-.97,-1.04,-1.11,-1.18,-1.25,-1.32,-1.39,-1.46
     .  ,-1.53,-1.59,-1.64,-1.67,-1.69,-1.70,-1.70,50*.0/
c
      x=xx
      if(model.eq.0.)goto 9000
      if(model.eq.1.)goto 9100
      if(model.eq.11)goto 1100
      if(model.eq.12)goto 1200
      if(model.eq.21)goto 2100
      if(model.eq.22)goto 2200
      if(model.eq.23)goto 4100
      if(model.eq.31)goto 3100
      if(model.eq.32)goto 3200
      if(model.eq.33)goto 3300
      if(model.eq.41)goto 4100
      if(model.eq.51)goto 5100
      write(6,60)model
      goto 9000
c
```

```
c           HELIGMAN-POLLARD MORTALITY FUNCTION   (MODEL 1-1)
c
1100     y4=p(8)
         do 110 i=mi,ma,iv
         if(x.le.0.)y1(i)=1.
         if(p(1).le.0.)y1(i)=0.
         if(p(1).gt.0..and.x.gt.0.)
         y1(i)=ee(1.,0.,0.,alog(p(1)),p(2),0.,alog(x))
         y2(i)=0.
         if(p(6).gt.0..and.p(7).gt.0..and.x.gt.0.)
         y2(i)=-((alog(x)-alog(p(6)))/p(7))**2.
         y2(i)=e(p(5),y2(i),1.)
         if(p(4).ne.0.)y3(i)=e(p(3),1./p(4),x)
         y3(i)=y3(i)/(1.+y3(i))
         if(p(4).eq.0.)y3(i)=1.
         y(i)=y1(i)+y2(i)+y3(i)+y4
110      x=x+dx
         if(mi.eq.1..and.dx.eq.1.)y(1)=p(9)
         if(mi.eq.0..and.dx.lt.1.)y(1)=p(9)
         return
c
c           ROGERS MORTALITY FUNCTION    (MODEL 1-2)
c
1200     y4=p(9)
         do 120 i=mi,ma,iv
         y1(i)=e(p(1),-p(2),x)
         y2(i)=ee(p(5),-p(7),p(6),-1.,-p(8),p(6),x)
         y3(i)=e(p(3),p(4),x)
         y(i)=y1(i)+y2(i)+y3(i)+y4
120      x=x+dx
         return
c
c           COALE & TRUSSELL FERTILITY FUNCTION    (MODEL 2-1 & 2-2)
c
2100     continue
         if(p(1).gt.1.)p(1)=1.
```

```
          if(p(3).gt.15.)p(3)=15.
          if(p(4).gt.35.)p(4)=35.
          p(5)=p(3)/sqrt(43.34)
          p(o)=p(4)-11.36*p(5)
          px0=p(6)
          p1=1.2313/p(3)
          p2=p(4)-.305*p(3)
          p3=1.145/p(3)
          p4=1.896/p(3)
          goto 2201
 2200     if(p(1).gt.1.)p(1)=1.
          p(5)=p(3)*sqrt(43.34)
          p(6)=p(4)+11.36*p(3)
          px0=p(4)
          p1=.1946/p(3)
          p2=p(4)+6.06*p(3)
          p3=.174/p(3)
          p4=.288/p(3)
 2201     shift=mi-(x+dx)
          mi0=(px0-shift)+1
          if(dx.lt.1.)goto 2999
c            Warning: y2 is an integral starting at x0 (parameter px0)
c            initial x is set to px0 and mi to mi0 therefore if mi .gt. mi0
          if(mi.le.mi0)mi0=mi
          x=x-(mi-mi0)
 2202     y4=p(7)
          do 210 i=mi0,ma,iv
          y1(i)=e(p(1)*n(i),p(2),v(i))
          y3(i)=ee(p1,-p3,p2,-1.,-p4,p2,x)
 210      x=x+dx
          y2(mi0)=0.
          y2(mi0+iv)=(y3(mi0)+(y3(mi0+iv)-y3(mi0))*(px0-mi0+shift)/dx
     .                +y3(mi0+iv))*(mi0+iv-shift-px0)*.5
          y(mi0)=y4
          y(mi0+iv)=y1(mi0+iv)*y2(mi0+iv)+y4
```

```
            do 215 i=mi0+2*iv,ma,iv
            y2(i)=y2(i-iv)+.5*dx*(y3(i-iv)+y3(i))
            y(i)=y1(i)*y2(i)+y4
215         continue
            return
2999        call err('func',2201,2202,'dx',0,0,dx)
            return
c
c              MIGRATION FUNCTION (ROGERS)   (MODEL 3-1)
c
3100        y4=p(13)
            do 310 i=mi,ma,iv
            y1(i)=ee(p(1),-p(3),p(2),-1.,-p(4),p(2),x)
            y2(i)=ee(p(5),-p(7),p(6),-1.,-p(8),p(6),x)
            y3(i)=ee(p(9),-p(11),p(10),-1.,-p(12),p(10),x)
            y(i)=y1(i)+y2(i)+y3(i)+y4
310         x=x+dx
            return
c
c              MIGRATION FUNCTION (ROGERS & CASTRO)   (MODEL 3-2)
c
3200        y4=p(11)
            do 320 i=mi,ma,iv
            y1(i)=e(p(1),-p(2),x)
            y2(i)=ee(p(3),-p(5),p(4),-1.,-p(6),p(4),x)
            y3(i)=ee(p(7),-p(9),p(8),-1.,-p(10),p(8),x)
            y(i)=y1(i)+y2(i)+y3(i)+y4
320         x=x+dx
            return
c
c              MIGRATION FUNCTION WITH RETIREMENT SLOPE   (MODEL 3-3)

3300        y4=p(9)
            do 330 i=mi,ma,iv
            y1(i)=e(p(1),-p(2),x)
            y2(i)=ee(p(3),-p(5),p(4),-1.,-p(6),p(4),x)
```

```
            y3(i)=e(p(7),p(8),x)
            y(i)=y1(i)+y2(i)+y3(i)+y4
330         x=x+dx
            return
c
c              SIMPLE DOUBLE EXPONENTIAL FUNCTION   (MODEL 4-1)
c
4100        do 410 i=mi,ma,iv
            y(i)=ee(p(1),-p(3),p(2),-1.,-p(4),p(2),x)
410         x=x+dx
            return
c
c              GENERAL DOUBLE EXP. FUNCTION (II)   (MODEL 5-1)
c
5100        y4=p(13)
            do 510 i=mi,ma,iv
            xmu=mu(p(2),p(3),p(4))
            y1(i)=ee(p(1),-p(3),xmu,-1.,-p(4),xmu,x)
            xmu=mu(p(6),p(7),p(8))
            y2(i)=ee(p(5),-p(7),xmu,-1.,-p(8),xmu,x)
            xmu=mu(p(10),p(11),p(12))
            y3(i)=ee(p(9),-p(11),xmu,-1.,-p(12),xmu,x)
            y(i)=y1(i)+y2(i)+y3(i)+y4
510         x=x+dx
            return
c
c              ZERO FUNCTION   (MODEL 0-0)
c
9000        do 900 i=mi,ma,iv
900         y(i)=0.
            return
c
c              ORIGINAL (OBSERVED) FUNCTION   (MODEL 0-1)
c
9100        do 910 i=mi,ma,iv
910         y(i)=obs(i)
            return
c
```

```
60      format(/11x,'--- ERROR: unknown model schedule',i3/)
        end
c
c FUNCTION E    -        calculates 'a * exp(alpha * x)'
c
        function e(a,alpha,x)
        real max
        data max/85./
        e=0.
        absa=abs(a)
        if(absa.eq.0.)return
        e=alog(absa)+alpha*x
        if(e.gt.max)e=max
        if(e.lt.-max)e=-max
        e=exp(e)
        if(a.lt.0.)e=-e
        return
        end
c
c FUNCTION EE   -        caluculates double exponential
c                        'a * exp[alpha(x-mu)+a2*exp(lambda(x-mu2))]'
c
        function ee(a,alpha,mu,a2,lambda,mu2,x)
        real max,mu,lambda,mu2
        data max/85./
        ee=0.
        absa=abs(a)
        if(absa.eq.0.)return
        ee=e(a2,lambda,x-mu2)
```

```
      ee=alog(absa)+alpha*(x-mu)+ee
      if(ee.gt.max)ee=max
      if(ee.lt.-max)ee=-max
      ee=exp(ee)
      if(a.lt.0.)ee=-ee
      return
      end
c
c FUNCTION MU   -       calculates double exponential mu as:
c                       'max + ln[alpha/lambda]/lambda'
c
      real function mu(max,alpha,lambda)
      real max,lambda
      mu=max
      if(lambda.eq.0.)return
      z=alpha/lambda
      if(z.le.0.)return
      mu=max+alog(z)/lambda
      return
      end
```

```
c FUN-CAL    -    computes model schedule y(x), age x=xmi,xma,dx
c                 field indices i=mi,ma,iv with i approximately = age
c                 if dx=n (1,2...) and i=1,2,... if dx < 1 (e.g. dx=0.1)
c                 FUN-CAL calls FUNC
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c subroutine funcal(mod,p,np,y,ny,xmi,xma,dx,ier)
c
c parameters:     mod      - model schedule code (input)
c                 p        - model schedule parameters (input)
c                 np       - length of vector p (input)
c                 y        - model schedule (output)
c                 ny       - length of vector y (input)
c                 xmi,xma,dx
c                          - minimum, maximum age, age interval for which
c                            the model schedule will be calculated
c                 ier      - error flag
c                            ier =  19: np less than n of mod.sched.param.
c                            ier =  20: np greater than 20
c                            ier = 101: ny greater than 101
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        subroutine funcal(mod,pp,np,yy,ny,xxmi,xxma,ddx,ier)
        real pp(np),yy(ny)
        common/cfun/model,p(20),y(101),y123(303)
1000    if(np.lt.nop(model).and.np.ne.1)ier=19
        if(np.gt.20)ier=20
        if(ny.gt.101)ier=101
        if(ier.gt.0)goto 9000
        model=mod
        xmi=xxmi
        xma=xxma
        dx=ddx
        if(xmi.eq.0..and.xma.eq.0..and.dx.eq.0.)xmi=.5
```

```
          if(xma.lt.xmi)xmi=float(ny)-.5
          if(dx.eq.0.)dx=1.
          mi=(xmi+1)/1
          ma=(xma+1)/1
          iv=dx
          x=xmi
          if(dx.ge.1.)goto 2000
          mi=1
          ma=(xma-xmi)/dx+1
          iv=1
 2000     if(np.eq.0)goto 3000
          do 200 i=1,np
  200     p(i)=pp(i)
          do 205 i=1,101
  205     y(i)=0.
 3000     call func(mi,ma,iv,x,dx)
          if(ny.eq.1)return
          do 300 i=1,ny
  300     yy(i)=y(i)
          return
 9000     write(6,69)ier
          return
   69     format(' ---ERROR:   funcal-ier = ',i4/)
          end
```

```
c FUNCTS    -     function subprograms: model schedule informations
c                 (1) no of parameters (function nop)
c                 (2) model schedule name (function msnam)
c                 (3) model schedule paramter names (function parnam)
c                 (4) definition of scaling parameters (function mk)
c                 (5) model schedule parameter defaults (function defpar)
c                 FUNCTS call no subroutines
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c model schedule parameters:
c
c       --------------------------- model ----------------------------
c param 1-1     1-2     2-1     2-2     2-3     3-1     3-2     3-3     4-1
c
c 1     Q1      a1      M*c     M*c     a       a1      a1      a1      (as 2-3)
c 2     gamma   alpha1  m       m       mu      mu1     alpha1  alpha1
c 3     Qs      a3      S       k       alpha   alpha1  a2      a2
c 4     Xs      alpha3  xbar    x0      lambda  lambda1 mu2     mu2
c 5     Qa      a2      (k)     (S)     -       a2      alpha2  alpha2
c 6     Xa      mu2     (x0)    (xbar)  -       mu2     lambda2 lambda2
c 7     sigma   alpha2  -       -       -       alpha2  a3      a3
c 8     c       lambda2 -       -       -       lambda2 mu3     alpha3
c 9     Q0      c       -       -       -       a3      alpha3  c
c 10    -       -       -       -       -       mu3     lambda3 -
c 11    -       -       -       -       -       alpha3  c       -
c 12    -       -       -       -       -       lambda3 -       -
c 13    -       -       -       -       -       c       -       -
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        function nop(model)
        integer m(13),n(14)
        data nom/13/,m/0,1,11,12,0,21,22,23,31,32,33,41,51/
        data n/0,0,9,9,20,4,4,4,13,11,9,4,13,20/
        do 100 i=1,nom
```

```
             if(model.eq.m(i))goto 2000
  100        continue
  2000       nop=n(i)
             return
             end
  c
             character*60 function msnam(model)
             integer m(13)
             character msn(14)*60
             data nom/13/,m/0,1,11,12,0,21,22,23,31,32,33,41,51/
             data msn
        .    /'zero schedule   (model schedule 0-C)'
        .    ,'observed schedule   (schedule 0-1)'
        .    ,'Heligman-Pollard mortality schedule   (model schedule 1-1)'
        .    ,'Rogers mortality schedule   (model schedule 1-2)'
        .    ,'noname model schedule   (model schedule ?-?)'
        .    ,'Coale-Trussell fertility schedule (S, xbar)   (mod.sch. 2-1)'
        .    ,'Coale-Trussell fertility schedule (k, x0)   (mod.sched. 2-2)'
        .    ,'double exponential fertility schedule   (model schedule 2-3)'
        .    ,'general double exponential schedule   (model schedule 3-1)'
        .    ,'Rogers & Castro migration schedule   (model schedule 3-2)'
        .    ,'Rogers & Castro migr. sched. with ret.slope   (mod.sch. 3-3)'
        .    ,'single double exponential schedule   (model schedule 4-1)'
        /    ,'general double expon. schedule (II)   (model schedule 5-1)'
        .    ,'unknown model schedule !'/
             do 100 i=1,nom
             if(model.eq.m(i))goto 2000
  100        continue
  2000       msnam=msn(i)
             return
             end
  c
             character*8 function parnam(model,iop)
             integer m(13)
             character pn(20,14)*8
             common/cfun/ignore,p(20),y0123(404)
             data nom/13/,m/0,1,11,12,0,21,22,23,31,32,33,41,51/
             data pn
```

```
         .  /20*'-',20*'-'
         .  ,'Q.1','gamma','Q.s','X.s','Q.a','X.a','sigma','c','Q.0',11*'-'
         .  ,'a.1','alpha.1','a.3','alpha.3','a.2','mu.2','alpha.2'
         .  ,'lambda.2','c',11*'-'
         .  ,20*'-'
         .  ,'M*c','m','S','xbar','(k)','(x0)',14*'-'
         .  ,'M*c','m','k','x0','(S)','(xbar)',14*'-'
         .  ,'a','mu','alpha','lambda',16*'-'
         .  ,'a.1','mu.1','alpha.1','lambda.1','a.2','mu.2','alpha.2'
         .  ,'lambda.2','a.3','mu.3','alpha.3','lambda.3','c',7*'-'
         .  ,'a.1','alpha.1','a.2','mu.2','alpha.2','lambda.2','a.3'
         .  ,'mu.3','alpha.3','lambda.3','c',9*'-'
         .  ,'a.1','alpha.1','a.2','mu.2','alpha.2','lambda.2','a.3'
         .  ,'alpha.3','c',11*'-'
         .  ,'a','mu','alpha','lambda',16*'-'
         .  ,'a.1','max.1','alpha.1','lambda.1','a.2','max.2','alpha.2'
         .  ,'lambda.2','a.3','max.3','alpha.3','lambda.3','c',7*'-'
         .  ,20*' '/
  1000      do 100 i=1,nom
            if(model.eq.m(i))goto 2000
  100       continue
  2000      parnam=pn(iop,i)
            return
            end
  c
            function mk(model,iop)
            integer m(13),mark(20,14)
            data nom/13/,m/0,1,11,12,0,21,22,23,31,32,33,41,51/
            data mark
         .  /20*0,20*0
         .  ,1,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0
         .  ,1,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
         .  ,20*0
         .  ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

```
     .   ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     .   ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     .   ,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0
     .   ,1,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0
     .   ,1,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0
     .   ,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
     .   ,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0
     .   ,20*0/
         do 100 i=1,nom
         if(model.eq.m(i))goto 2000
100      continue
2000     mk=mark(iop,i)
         return
         end
c
         function defpar(model,iop)
         integer m(13)
         real dp(20,14)
         data nom/13/,m/0,1,11,12,15,21,22,23,31,32,33,41,51/
         data dp/20*0.0,20*0.0
     .   ,.001,.1,.00001,10.,.0002,20.,.5,0.,.01,11*0.0
     .   ,.005,.5,.00001,.1,.001,18.,.1,.3,0.,11*0.0
     .   ,20*0.0
     .   ,.41,2.,6.6,25.,16*0.0
     .   ,.41,2.,1.,14.,16*0.0
     .   ,.1,30.,.3,.1,16*0.0
     .   ,.05,16.,.1,1.5,.05,25.,.1,.2,.01,50.,.1,.2,.0,7*0.0
     .   ,.02,.1,.06,20.,.1,.4,.0001,80.,.5,.1,.003,9*0.0
     .   ,.02,.1,.06,20.,.1,.4,.0001,.05,.003,11*0.0
     .   ,.1,20.,.1,.5,16*0.0
     .   ,20*0.0
     .   ,20*0.0/
c
         do 100 i=1,nom
         if(model.eq.m(i))goto 2000
100      continue
2000     defpar=dp(iop,i)
         return
         end
```

```
c GRADRA    -     draws graphic
c                 GRADRA calls no subroutines
c
        subroutine gradra(x,n,char)
        character page(55)*112,char*1
        real x(n)
        common/cgra/transf,adjust,np,mx,nx,page
c
c          DRAW GRAPHIC
c
        do 300 i=1,n
        j=i+11
        index=x(i)*transf+adjust
        if(index.gt.np)index=np
        if(index.lt.2)index=2
        write(page(index)(1:10),'(f10.6)')(float(index)-adjust+.5)/transf
        page(index)(j:j)=char
300     continue
c
        return
        end
```

```
c GRAINI   -    initializes graphic page
c               GRAINI calls no subroutines
c
         subroutine graini(ymin,ymax,lines,cols,tit)
         character page(55)*112,tit*(*)
         integer cols
         common/cgra/transf,adjust,np,mx,nx,page
c
c           SET PAGE AND GRAPH PARAMETERS
c
         yrange=ymax-ymin
         prange=lines
         if(lines.le.0.or.lines.gt.50)prange=50.
         transf=prange/yrange
         adjust=2.5
         np=prange+adjust
         adjust=adjust-ymin*transf
         mx=1
         if(cols.gt.0.and.cols.lt.76)mx=12
         nx=cols+12
         if(cols.le.0.or.cols.gt.100)nx=112
c
c           INITIALIZE PAGE
c
         do 200 i=1,55
         do 202 j=1,112
  202    page(i)(j:j)=' '
         if(i.gt.1.and.i.le.np)page(i)(12:12)='I'
  200    continue
         page(np+2)(12:71)=tit
         i=adjust
         if(i.lt.2.or.i.gt.np)i=2
         page(i)(12:12)='+'
         do 205 j=13,112,10
  205    page(i)(j:j+9)='---------I'
         write(page(1),'(2x,11i10)')(i,i=0,100,10)
c
         return
         end
```

```
c GRAPH    -       draws graphics
c                  GRAPH calls GRA-INI, GRA-DRA & GRA-PRI
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   usage
c        call graph(a,b,na,nb,ymin,ymax,lines,cols,ch,tit,file)
c
c   parameters
c        a         - vector of length na (input)
c        b         - vector of length nb (input)
c        na        - number of elements in a (input)
c        nb        - number of elements in b (input)
c        ymin      - minimum value of y-axis (input)
c        ymax      - maximum value of y-axis (input)
c        lines     - number of lines (rows) of graphic (input)
c        cols      - number of columns of graphic (integer) (input)
c        ch        - character variable of length 2 (input)
c                    ch(1:1) is symbol for vector-a-values
c                    ch(2:2) is symbol for vector-b-values
c        tit       - title (character variable of variable length) (input)
c        file      - output file unit (integer) (input)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
         subroutine graph(a,b,na,nb,ymin,ymax,lines,cols,ch,tit,file)
         character ch*2,tit*(*)
         integer cols,file
         real a(na),b(nb)
         call graini(ymin,ymax,lines,cols,tit)
         call gradra(b,nb,ch(2:2))
         call gradra(a,na,ch(1:1))
         call grapri(file)
         return
         end
```

```
c GRAPRI   -    prints graphic page
c               GRAPRI calls no subroutines
c
        subroutine grapri(file)
        character page(55)*112
        integer file
        common/cgra/transf,adjust,np,mx,nx,page
c
c           PRINT PAGE
c
        if(file.le.0)return
        write(file,'("1")')
        do 400 i=np+3,1,-1
400     write(file,'(1x,130a1)')(page(i)(j:j),j=mx,nx)
c
        return
        end
```

```
c IN-DAT    -     reads a 'standard-type' (age-specific) data set
c                 IN-DAT calls no subroutines
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c standard-type data set:
c
c - 1st card:   title   (a60)
c        ('title' should not start with '--- ' or 'end ')
c
c - 2nd card:   low, ihi, iv, fmt    (3i3,a20)
c        - low = lowest age for which observations are available
c        - ihi = highest age for which observations are available
c        - iv  = age intervall of observation data
c        - fmt = format in which observations have to be read
c        e.g. age = 0,1,2 ...85+ :                 low=0, ihi=85, iv=1
c             age = 0,5,10 ...85+ :                low=0, ihi=85, iv=5
c             age = below 15,15,20 ...85+ :       low=10, ihi=85, iv=5
c
c - 3rd and following cards:    observations   (fmt)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        subroutine indat(file,tit,m,n,iv,x,ier)
        dimension x(101)
        character fmt*20,tit*60
        integer file
        do 100 i=1,101
```

```
100     x(i)=0.
1000    read(file,'(a)',end=9999)tit
        if(tit(1:4).eq.'--- '.or.tit(1:4).eq.'end ')goto 9000
        ier=1
        read(file,'(3i3,a20)',end=9999)m,n,iv,fmt
        if(iv.eq.0)iv=1
        m=m+1
        n=n+1
        ier=2
        read(file,fmt,end=9999)(x(i),i=m,n,iv)
        ier=0
        if(iv.eq.1)return
        ivmo=iv-1
        do 200 i=m,n,iv
        do 200 j=i+1,i+ivmo
200     x(j)=x(i)
        n=n+ivmo
        return
9000    if(ier.eq.0)goto 1000
9999    ier=ier+1
        return
        end
```

```
c I/O-PAR    -    stores parameters on intermediate auxilary file
c                 only required when no unformatted reading from
c                 internal files (character variables) is allowed
c
        subroutine iopar(iofile,params)
        character params*130
        if(iofile.le.0)iofile=4
        rewind iofile
        write(iofile,'(a)')params
        rewind iofile
        return
        end
```

```
c MAXIM    -     finds local maximum of model schedule (see common/cfun/)
c                in the interval [xmin-iv,xmin+iv]
c                MAXIM calls MIMAX2
c
        subroutine maxim(xmax,ymax,iv)
        call mimax2(xmax,ymax,iv,2)
        return
        end
```

```
c MIMAX   -   looks for up to 5 mimimum and maximum values (ymin,
c             ymax) of the step function y(x), x=mi,ma,iv;
c             ymin(j)=y(xmin(j)), ymax(j)=y(xmax(j))
c             j=1: absolute mimimum resp. maximum
c             MIMAX calls no subroutines
c
          subroutine mimax(y,mi,ma,iv)
          real y(101)
          common/cmim/xmin(6),ymin(6),xmax(6),ymax(6)
          xmin(1)=mi
          ymin(1)=y(mi)
          xmax(1)=mi
          ymax(1)=y(mi)
          do 100 i=2,6
          xmin(i)=0.
          ymin(i)=0.
          xmax(i)=0.
          ymax(i)=0.
c100      continue
          j=1
          i=mi-iv
 1000     j=j+1
          if(j.gt.6)return
 2000     i=i+iv
          xmin(j)=i
          ymin(j)=y(i)
          if(i.ge.ma)return
          if(y(i).ge.y(i+iv))goto 2000
          if(ymin(j).gt.ymin(1))goto 3000
          xmin(1)=xmin(j)
          ymin(1)=ymin(j)
 3000     i=i+iv
          xmax(j)=i
          ymax(j)=y(i)
          if(i.ge.ma)return
          if(y(i).lt.y(i+iv))goto 3000
          if(ymax(j).lt.ymax(1))goto 1000
          xmax(1)=xmax(j)
          ymax(1)=ymax(j)
          goto 1000
          end
```

```
c MIMAX2   -    working subroutine for minim and maxim
c               MIMAX2 calls FUNC & MIMAX
c
          subroutine mimax2(xm,ym,iv,mima)
          common/cfun/model,p(20),y(101),y123(303)
          common/cmim/xmin(6),ymin(6),xmax(6),ymax(6)
          dx=iv
          n=alog(.0000001/dx)/alog(.02)+1
          x=xm-dx
          dx=dx*2./100.
          do 100 k=1,n
          call func(1,101,1,x,dx)
          call mimax(y,1,101,1)
          if(mima.eq.1)x=x+(xmin(1)-1.)*dx-dx
          if(mima.eq.2)x=x+(xmax(1)-1.)*dx-dx
          if(k.lt.n)dx=dx*2./100.
100       continue
          xm=x+dx
          if(mima.eq.1)ym=ymin(1)
          if(mima.eq.2)ym=ymax(1)
          return
          end
```

```
c MIMIM    -      finds local mimimum of model schedule (see common/cfun/)
c                 in the interval [xmin-iv,xmin+iv]
c                 MINIM calls MIMAX2
c
          subroutine minim(xmin,ymin,iv)
          call mimax2(xmin,ymin,iv,1)
          return
          end
```

```
c PLO-DAT    -   prints data (e.g. as plot input data)
c                PLO-DAT calls no subroutines
c
      subroutine plodat(x,y,y1,y2,y3,mi,ma,iiv,shift,file,dec)
      character fmt*12
      integer shift,file,dec
      real x(*),y(*),y1(*),y2(*),y3(*),lo(101),ly(101)
      data fmt/'(i3,12f10.6)'/
      iv=iiv
      if(iiv.eq.0)iv=1
      if(dec.lt.10)write(fmt(11:11),'(i1)')dec
      do 100 i=mi,ma,iv
      if(i.gt.101)return
      if(x(i).gt.0.)lo(i)=alog10(x(i))
      if(y(i).gt.0.)ly(i)=alog10(y(i))
      if(x(i).lt..0000005)lo(i)=-9.999999
      if(y(i).lt..0000005)ly(i)=-9.999999
      pcdev=.0
      if(x(i).ne.0.)pcdev=(y(i)-x(i))*100./x(i)
      if(abs(pcdev).ge.99.95)pcdev=pcdev*abs(99.9/pcdev)
      if(abs(y(i)).ge.99.95)y(i)=y(i)*abs(99.9/y(i))
      write(file,fmt)i+shift,x(i),y(i),y1(i),y2(i),y3(i)
     .           ,y(i)-x(i),pcdev,lo(i),ly(i)
100   continue
      return
      end
```

```
c RESID     -     calculates residuals
c                 RESID calls FUNC
c
          subroutine resid(pp,mm,nn,r)
          dimension pp(nn),r(mm),ppp(20)
          common/cage/low,ihi,mi,ma,iv
          common/cobs/x(101)
          common/cfun/model,p(20),y(101),y123(303)
          common/ccon/cont
          common/cres/res,pr,iter,p0(20),fix(20),adj(20),pmin(20),pmax(20)
c
          iter=iter+1
          n=0
          do 100 i=1,20
          if(fix(i).eq.1.)goto 100
          n=n+1
          p(i)=pp(n)*adj(i)
          if(p0(i).eq.0..and.pp(n).ne.1.)p(i)=pp(n)*.0000001
          if(p(i).lt.pmin(i))p(i)=p0(i)
          if(p(i).gt.pmax(i))p(i)=p0(i)
          ppp(n)=p(i)
          if(n.eq.nn)goto 2000
100       continue
2000      call func(mi,ma,iv,float(mi)-.5,float(iv))
c
          if(res.ne.0.)goto 6000
5000      do 500 i=mi,ma
          j=i-mi+1
          z=y(i)
          if(z.ne.0.)goto 500
          z=x(i)
```

```
      if(z.eq.0.)z=1.
500   r(j)=(x(i)-y(i))/sqrt(abs(z))
c
9000  if(pr.ne.1.)goto 9500
      do 900 j=1,ma-mi+1
900   r(j)=abs(r(j))
9500  if(cont.eq.0.)return
      ssq=0.
      do 950 j=1,ma-mi+1
      if(ssq.gt.100..or.r(j).gt.10.)goto 9502
950   ssq=ssq+r(j)*r(j)
9502  if(ssq.gt.99.9)ssq=99.9
      write(6,'(1x,i4,":",14f9.6)')iter,ssq,(ppp(i),i=1,n)
      return
c
6000  if(res.ne.1.)goto 7000
      do 600 i=mi,ma
      j=i-mi+1
600   r(j)=x(i)-y(i)
      goto 9000
c
7000  if(res.ne.2)goto 5000
      do 700 i=mi,ma
      j=i-mi+1
      xx=x(i)
      yy=y(i)
      if(xx.eq.yy)goto 700
      yy=alog10(yy)
      if(xx.lt.1.e-15)xx=1.e-15
      xx=alog10(xx)
700   r(j)=xx-yy
      goto 9000
      end
```

```
c STAT   -       calculates standard statistics
c                STAT may call subroutine ERR
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c usage
c       call stat(x,n,m,no,el,nobs
c                                     ,xmin,xmax,xmea,xmed,xmod,xvar,xstd,xstn)
c
c parameters
c       x        - matrix of dimension n x m containing n observations
c                  (maximum) for each of the m variables (input)
c       n        - number of rows of matrix x (input)
c       m        - number of columns of matrix x (input)
c       no       - number of observations (input); no.le.n
c       el       - value standing for 'unknown' or 'ignore' (input)
c       nobs     - vector of length m containing the used number of obser-
c                  vations ('no' minus eliminated observations) (output)
c       xmin     - vector of length m containing lowest values (output)
c       xmax     - vector of length m containing highest values (output)
c       xmea     - vector of length m containing mean values (output)
c       xmed     - vector of length m containing medians (output)
c       xmod     - vector of length m containing modes (output)
c       xvar     - vector of length m containing variances (output)
c       xstd     - vector of length m containing std. deviations (output)
c       xstn     - vector of length m containing std.dev./mean (output)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
        subroutine stat(x0,n,m,no,el,nobs
     .  ,xmin,xmax,xmea,xmed,xmod,xvar,xstd,xstn)
        parameter (maxn=100)
        integer nobs(m)
        real x0(n,m),xmin(m),xmax(m),xmea(m),xmed(m),xmod(m),xvar(m)
     .  ,xstd(m),xstn(m),x(maxn),y(maxn),z(11),nx(10)
        if(n.lt.1.or.n.gt.maxn)call err('stat',0,200,'n',0,0,n)
```

```
          if(no.lt.1.or.no.gt.n)call err('stat',0,200,'no',0,0,no)
          do 200 j=1,m
c
c            LOWEST & HIGHEST VALUE, NUMBER OF OBS, SUM, SUM OF SQUARES
c
          s=0.
          ss=0.
          nob=0
          xmi=999999.
          xma=-999999.
          xmi1=xmi
          xma1=xma
          xme=0.
          xmo=0.
          do 300 i=1,no
          x(i)=x0(i,j)
          if(x(i).lt.xmi)xmi=x(i)
          if(x(i).gt.xma)xma=x(i)
          if(x(i).eq.el)goto 300
          nob=nob+1
          x(nob)=x(i)
          y(nob)=x(i)
          s=s+x(i)
          ss=ss+x(i)*x(i)
          if(x(i).lt.xmi1)xmi1=x(i)
          if(x(i).gt.xma1)xma1=x(i)
300       continue
          if(nob.eq.0)goto 9000
c
c            MEDIAN
c
          nob2=(nob+1)/2
          do 400 i=1,nob
          yma=0.
          ima=0
          do 405 ii=1,nob
          if(y(ii).le.yma)goto 405
          yma=y(ii)
```

```
              ima=ii
      405     continue
              if(i.eq.nob2)xme=yma
              if(ima.gt.0)y(ima)=0.
      400     continue
      c
      c          MODE
      c
              xdi=(xma1-xmi1)/10.
              nx(1)=0
              z(1)=xmi1
              do 500 k=2,10
              nx(k)=0
              z(k)=z(k-1)+xdi
      500     continue
              z(11)=xma1
              do 502 i=1,nob
              do 502 k=1,10
              if(x(i).ge.z(k).and.x(i).lt.z(k+1))nx(k)=nx(k)+1
      502     continue
              nma=0
              ima=0
              do 505 k=1,5
              if(nx(k).lt.nma)goto 505
              nma=nx(k)
              ima=k
      505     continue
              do 507 k=10,6,-1
              if(nx(k).lt.nma)goto 507
              nma=nx(k)
              ima=k
      507     continue
              if(ima.gt.0)xmo=(z(ima)+z(ima+1))/2.
      c
      c          FINISH
      c
```

```
9000    xmea(j)=0.
        xvar(j)=0.
        xstd(j)=0.
        xstn(j)=0.
        nobs(j)=nob
        xmin(j)=xmi
        xmax(j)=xma
        if(nob.gt.0)xmea(j)=s/nob
        xmed(j)=xme
        xmod(j)=xmo
        if(nob.lt.2)goto 200
        xvar(j)=(ss-s*s/nob)/(nob-1)
        if(xvar(j).gt.0.)xstd(j)=sqrt(xvar(j))
        if(s.ne.0.)xstn(j)=xstd(j)/xmea(j)
 200    continue
c
        return
        end
```

```
c V-GOF    -      vector function: goodness-of-fit
c
           function vgof(x,y,n,mi,ma,iv)
           real x(n),y(n)
           vgof=0.
           sx=0.
           se=0.
           do 100 i=mi,ma,iv
           sx=sx+x(i)
           se=se+abs(x(i)-y(i))
  100      continue
           if(sx.ne.0.)vgof=100.*se/sx
           if(vgof.gt.99.9)vgof=99.9
           return
           end
```

```
c V-SUM    -      vector function: sum
c
           function vsum(x,n,mi,ma,iv)
           real x(n)
           vsum=0.
           do 100 i=mi,ma,iv
  100      vsum=vsum+x(i)
           return
           end
```

APPENDIX C: VNIISI SUBROUTINES

```
c*      ident  *  8-apr-80  *  this program was made at VNIISI  *
c*      dec - 80
c*      last revised  *10-jan-81*  and renamed cident
c       version nov'82   (IIASA/Planck)
c*
c*
c*
c*      purpose
c*      to identify the system's parameters
c*
c*      usage
c*      call ident(q0,n,delta,epsi,eps,nksi,nstop,ipr,pksi,h0,alb,np,
c*   *  pq,q00,isub,npp,nfi,epsfi,amu)
c*
c*      description of parameters
c*      q0-vector of parametrs to identify
c*      n - dimension of vector q (less or equal to 19)
c*      delta - vector of initial increments (dimension n)
c*      epsi - vector of precisions to stop the search (dimension n)
c*      eps -parametr of search (recomended value-  .1)
c*      nksi - the number of measurments in least square technique
c*             (less or equal to 1000)
c*      nstop - maximum number of steps
c*              after identification the number of steps that were
c*              made is stored here
c*      ipr - parametr which control the output information
c*      pksi - weight array which indicates the impact of each measur-
c*             ment in least squares technique
c*      h0 - initial step of movement in a direction (recomended
c*           value-  .1-1.)
c*           after identification it may contain the following values
c*           h0=1.- means that the process of identification stoped
c*                  because of cycling in parabola more then 10 times
c*           h0=2.- means end of search because precision was achieved
c*           h0=3.- maximum number of steps was achieved
c*           h0=4.- determinant in parabola search is equal 0.
c*      alb - parametr of search (recommended value-  .8)
c*            after search contains mean deviation
```

```
c*      np - parameter which shows how many times any row in matrix dq
c*           can be renewed, while there are rows that were not  renewed
c*           even once.
c*      pq - covariation matrix for parameters to be identified (n*n)
c*      q00 - the apriory information about parameters (n*1)
c*      isub - parameter which controls the mode of identification
c*             isub=0  nonsubsequental identification
c*             isub=1  sequental identification
c*      nnp - number of rows in pq matrics
c*      nfi - number of constraints
c*
c*      remarks
c*      the user must provide subroutine resid(q,nksi,nx,fksi)
c*      it's purpose is to genarate for each vector q vector fksi
c*      which contains the difference between the measured values of function
c*      and those calculated by model.
c*
c*      the user must provide subroutine constr(q,nx,fi,nfi)
c*      it's purpose is to genarate for each vector q vector
c*      of constraints fi which component fi(i) is equal to 0
c*      if fi(i).ge.0 and fi(i)=abs(fi(i)) if fi(i).lt.0
c*
c*
c*      subroutines and functions required
c*      minv,array,resid,csergo,subpq,cdir,cconstr,mxcut,dir2,detreq,cstep,ort
c*
c*      method
c*      the generalized least squares technique is used. the method of findig
c*      the solution is the finit-difference analog of gauss-newton method.
c*      it needn't computations of derivatives.
c*      constraints are processed as penalty functions
c*
c*     authors     :  GOLUBKOV V.   SCHERBOV S.
c*
c*
c*
        subroutine ident(zq0,n,zdelta,zepsi,eps,nksi,nstop,ipr,zpksi
```

```
     .,h0,alb,np,zpq,zq00,isub,nnp,nfi,epsfi,amu)
      parameter (npar=20,nmeas=1001,ncon=20)
      logical bul
      integer lw1(npar),nnn(npar)
      real ksinp1,max
      real zq0(*),zdelta(*),zepsi(*),zpksi(*),zpq(nnp,*),zq00(*)
      real delta(npar),qnk(npar),ksink(nmeas),fink(ncon)
     .,fi0(ncon),f(npar),aifi(npar,npar),ql(npar)
      common/cid/epsi(npar),lw2(npar),w(npar),finp1(ncon)
     .,fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
     .,ksinp1(nmeas),pksi(nmeas),dksi(nmeas,npar)
     .,q0(npar),q00(npar),qnp1(npar),pq(npar,npar)
     .,dq(npar,npar),dqh(npar)
      do 1234 i=1,n
      q0(i)=zq0(i)
      q00(i)=zq00(i)
      delta(i)=zdelta(i)
      epsi(i)=zepsi(i)
      do 2345 j=1,n
      pq(i,j)=0.
      aifi(i,j)=0.
      dq(i,j)=0.
2345  continue
      pq(i,i)=zpq(i,i)
      qnk(i)=0.
      f(i)=0.
      ql(i)=0.
      lw1(i)=0
      nnn(i)=0
      lw2(i)=0
      w(i)=0.
      qnp1(i)=0.
      dqh(i)=0.
      do 3456 j=1,nfi
      dfi(j,i)=0.
      dfi0(j,i)=0.
```

```
3456    continue
        do 4567 j=1,nksi
        dksi(j,i)=0.
4567    continue
1234    continue
        do 5678 i=1,nksi
        pksi(i)=zpksi(i)
        ksink(i)=0.
        ksinp1(i)=0.
5673    continue
        amu0=1.
        bh=.5
        alph=2.
        hmax=1.
c*
        kstop=0
        krec=0
1100    continue
        krec=krec+1
        kn=1
        do 1 i=1,n
        lw1(i)=0
        lw2(i)=0
1       dq(i,n+1)=q0(i)
        k=0
        call funct(dq(1,n+1),dksi(1,n+1),pksi,dfi(1,n+1),ff,amu
        ,nfi,nksi,n,np)
        if(isub.eq.1) call functa(n,dq(1,n+1),q00,ff,pq,nnp)
        f(n+1)=ff
        fff=ff
c*
        do 20 j=1,n
        do 10 i=1,n
        if(i.eq.j) q0(i)=q0(i)+delta(i)
        dq(i,j)=q0(i)
10    continue
```

```
           call funct(dq(1,j),dksi(1,j),pksi,dfi(1,j),ff,amu,nfi,nksi,n,np)
           if(isub.eq.1) call functa(n,dq(1,j),q00,ff,pq,nnp)
 20       f(j)=ff
          if(ipr.le.4) write (6,*) 'f=',(f(i),i=1,n+1)
c*
           do 30 i=1,n+1
           max=-1000.
           do 40 j=1,n+1
           if(f(j).lt.max) go to 50
           max=f(j)
           jmax=j
           ql(j)=f(j)
 50        continue
 40        continue
           f(jmax)=-1000.
           nnn(i)=jmax
 30        continue
              if(ipr.le.3) write (6,*) '************************************'
              if(ipr.le.3) write (6,*) '************************************'
              if(ipr.le.3) write (6,*) 'nnn=',nnn
              if(ipr.le.3) write (6,*) 'ql=',ql
           ff=fff
           do 600 i=1,n
 600       qnp1(i)=dq(i,n+1)
              if(ipr.le.6) write (6,*) 'qnp1=',(qnp1(i),i=1,n)
           do 610 i=1,nksi
 610       ksinp1(i)=dksi(i,n+1)
           do 620 i=1,nfi
 620       finp1(i)=dfi(i,n+1)
              if(ipr.le.4) write (6,*) 'finp1=',(finp1(i),i=1,nfi)
c*
           do 121 i=1,n
 121       dq(i,n+1)=dq(i,1)-dq(i,n+1)
```

```
      do 120 j=1,n-1
      do 120 i=1,n
120   dq(i,j)=dq(i,j+1)-dq(i,j)
      do 122 i=1,n
122   dq(i,n)=dq(i,n+1)
      do 131 i=1,nksi
131   dksi(i,n+1)=dksi(i,1)-dksi(i,n+1)
      do 130 j=1,n-1
      do 130 i=1,nksi
130   dksi(i,j)=dksi(i,j+1)-dksi(i,j)
      do 132 i=1,nksi
132   dksi(i,n)=dksi(i,n+1)
      do 141 i=1,nfi
141   dfi(i,n+1)=dfi(i,1)-dfi(i,n+1)
      do 140 j=1,n-1
      do 140 i=1,nfi
140   dfi(i,j)=dfi(i,j+1)-dfi(i,j)
      do 142 i=1,nfi
142   dfi(i,n)=dfi(i,n+1)
c*
         if(ipr.le.3)
     .   call mxout('dq',dq,npar,n,n,0,60,132,1)
      do 150 j=1,n
      max=-1000.
      do 160 i=1,n
      if(abs(dq(i,j)).lt.max) go to 160
      max=abs(dq(i,j))
      imax=i
160   continue
         if(ipr.le.3)print*,' max, imax = ',max,imax
      a1=0.
         if(dq(imax,j).eq.0.)
     .   call err('ident',160,170,'dq',imax,j,dq(imax,j))
      do 170 i=1,n
170   a1=a1+dq(i,j)**2/dq(imax,j)**2
      q1(j)=sqrt(a1)*abs(dq(imax,j))
150      if(ipr.le.3)write(6,*)'q1(',j,') = ',q1(j)
```

```
           det=1.
              if(ipr.le.2) call mxout('dq    ',dq,npar,n,n,0,60,132,1)
              if(ipr.le.2) write (6,*) 'dfi=',((dfi(i,j),j=1,n),i=1,nfi)
              if(ipr.le.2) call mxout('dksi  ',dksi,nmeas,n,nksi,0,60,132,1)
              if(ipr.le.2)write (6,*) 'ql=',(ql(i),i=1,n)
           do 190 j=1,n
           if(ql(j).eq.0.)call err('ident',150,180,'ql',j,0,ql(j))
           do 180 i=1,n
  180      dq(i,j)=dq(i,j)/ql(j)
           do 200 i=1,nksi
  200      dksi(i,j)=dksi(i,j)/ql(j)
           do 210 i=1,nfi
  210      dfi(i,j)=dfi(i,j)/ql(j)
  c*
           det=delta(j)/ql(j)*det
  190      continue
  1000     continue
              if(ipr.le.6) write (6,*) '******************************'
              if(ipr.le.2) call mxout('dq    ',dq,npar,n,n,0,60,132,1)
              if(ipr.le.2)write (6,*) 'dfi=',((dfi(i,j),j=1,n),i=1,nfi)
              if(ipr.le.1) call mxout('dksi  ',dksi,nmeas,n,nksi,0,60,132,1)
              if(ipr.le.1.and.nfi.gt.0)
        *     call mxout('dfi ',dfi,ncon,n,nfi,0,60,132,1)
              if(ipr.le.3) write (6,*) 'ql=',(ql(i),i=1,n)
           k=k+1
           kstop=kstop+1
              if(ipr.le.3) write (6,*) 'determinant=',det, 'k=',k
  c        nk=k-(k-1)/n*n
           if(nfi.eq.0) go to 220
           call dir(epsfi,
        *  nksi,n,nfi,nw,amu,spaspb,np,ipr,isub,nnp)
           if(spaspb.lt.epsfi) go to 330
           if(ipr.le.3) write (6,*) 'spaspb=',spaspb
           k31=0
           iii=0
  320      k0=0
           do 240 i=1,nfi
```

```
        bul=.true.
        do 250  j=1,n
250     bul=(abs(dfi(i,j)).lt.1.e-30).and.bul
        if(bul) go to 240
        k0=k0+1
        if(iii.eq.0) go to 240
        fi0(k0)=finp1(i)
        do 260 j=1,n
260     dfi0(k0,j)=dfi(i,j)
240     continue
        if(iii.eq.1) go to 270
280     k1=0
        do 290  j=1,n
        bul=.true.
        do 300 i=1,nfi
300     bul=(abs(dfi(i,j)).lt.1.e-30).and.bul
        if(bul) go to 290
        k1=k1+1
        if(iii.eq.0) go to 290
        do 310 i=1,nfi
        aifi(i,j)=0.
310     dfi0(i,k1)=dfi(i,j)
        aifi(k1,j)=1.
290     continue
        if(iii.eq.1) go to 270
        iii=1
        if(k0.eq.0.or.k1.eq.0) go to 220
        if(k0.gt.k1) k31=1
        if(k0-k1) 320,320,280
270     continue
```

```
          n0=k1
          nfi0=k0
             if(ipr.le.1) call mxout('dfi0 ',afi0,ncon,n,nfi,0,60,132,1)
             if(ipr.le.3)write (6,*) '**********entering dir1!*********'
             if(ipr.le.3)write (6,*) 'k31=',k31
          call dir1(gnk,
     *    nksi,n,nfi,nw,mu,spaspb,np,ipr,
     *    isub,nnp,aifi,
     *    nfi0,n0,k31,amu,amu0)
          go to 330
  220     continue
          call dir(epsfi,nksi,n,nfi,nw,amu,spaspb,np,ipr,isub,nnp)
  330     continue
          if(kstop.le.1) go to 552
          bul=.true.
          do 551 i=1,n
          bul=bul.and.(abs(dqh(i)).le.epsi(i))
  551     continue
          if(bul) h0=2.
          if(bul) go to 560
  552     continue
          do 340 i=1,n
  340     qnk(i)=qnp1(i)
          do 341 i=1,nksi
  341     ksink(i)=ksinp1(i)
          do 342 i=1,nfi
  342     fink(i)=finp1(i)
c*
             if(ipr.le.3) write (6,*) 'w=',(w(i),i=1,n)
             if(ipr.le.5) write (6,*) 'dqh=',(dqh(i),i=1,n)
c*
c*        begining of parabola search
c*
          kkk=0.
          if(k.gt.1) go to 350
          h10=h0
```

```
350   h1=0.
      f1=ff
      f0=f1
      hpar=h10
         if(ipr.le.3) write (6,*) 'hpar=',hpar
      do 360 i=1,n
360   qnp1(i)=qnp1(i)+hpar*dqh(i)
      call funct(qnp1,ksinp1,pksi,finp1,fpar,amu,nfi,nksi,n,np)
      if(isub.eq.1) call functa(n,qnp1,q00,fpar,pq,nnp)
         if(ipr.le.2) write (6,*) ' ** 1** '
      if(fpar.ge.f0) go to 370
      if(hpar.gt.0) h10=min(alph*hpar,hmax)
      if(hpar.le.0) h10=h0
380   continue
      ff=fpar
      go to 480
370   continue
      do 390 i=1,n
390   qnp1(i)=qnp1(i)-hpar*dqh(i)
      f2=fpar
      h2=hpar
      hpar=bh*hpar
      h3=hpar
410   continue
      do 400 i=1,n
400   qnp1(i)=qnp1(i)+hpar*dqh(i)
      call funct(qnp1,ksinp1,pksi,finp1,fpar,amu,nfi,nksi,n,np)
      if(isub.eq.1) call functa(n,qnp1,q00,fpar,pq,nnp)
      if(fpar.ge.f0) go to 420
      if(hpar.gt.0) h10=hpar
      if(hpar.le.0) h10=h0
      go to 380
420   continue
      do 430 i=1,n
430   qnp1(i)=qnp1(i)-hpar*dqh(i)
      if(hpar.ge.0) go to 440
      h3=hpar
```

```
             f3=fpar
             go to 450
      440    continue
             if(h3.le.0) go to 460
             h3=h2
             f3=f2
      460    h2=hpar
             f2=fpar
      450    continue
             kkk=kkk+1
             if(kkk.ge.10) h0=1.
             if(kkk.ge.10) go to 900
                if(ipr.le.3) write (6,*) 'h1,h2,h3=',h1,h2,h3
                if(ipr.le.3) write (6,*) 'f1,f2,f3=',f1,f2,f3
             det1=(h1-h3)**2*(h2-h3)-(h2-h3)**2*(h1-h3)
             if(abs(det1).lt.1.e-20) h0=4.
             if(abs(det1).lt.1.e-20) go to 900
             apar=((f1-f3)*(h2-h3)-(f2-f3)*(h1-h3))
             bpar=((h1-h3)**2*(f2-f3)-(h2-h3)**2*(f1-f3))
             cpar=f3
             if(apar.eq.0.)goto 470
             if(apar.le.0) go to 470
             hpar=-bpar/2./apar+h3
             go to 410
      470    hpar=-h0
             go to 410
      480    continue
             max=-1000.
             do 500 i=1,n
             if(abs(dqh(i)).lt.max) go to 500
             max=abs(dqh(i))
             imax=i
      500    continue
             a1=0.
             if(dqh(imax).eq.0.)goto 511
             do 510 i=1,n
```

```
510     a1=a1+dqh(i)**2/dqh(imax)**2
511     ql1=sqrt(a1)*abs(dqh(imax))
c********************
        si=1.
        if(hpar.lt.0.) si=-1.
        imax=0
        kg=0
481     fmax=-1000.
           if(ipr.le.3)write(6,*)' lw1 = ',lw1
           if(ipr.le.3)write(6,*)' w = ',w
        do 482 i=1,n
        if(lw1(i).ge.1) go to 482
        if(fmax.ge.abs(w(i))) go to 482
        fmax=abs(w(i))
        imax=i
482     continue
           if(ipr.le.3)print*,' imax = ',imax
        if(imax.ne.0) go to 483
        kn=1
        do 484 i=1,n
        lw1(i)=0
484     lw2(i)=0
        go to 481
483     if(ql1.ne.0.)detw=det*w(imax)/ql1
           if(ipr.le.3)print*,' detw, det, w(imax), ql1 ='
           ,detw,det,w(imax),ql1
           if(ipr.le.3)print*,' eps = ',eps
        if(abs(detw).lt.eps) go to 485
        det=detw
        nk=imax
           if(ipr.le.3)print*,' kg = ',kg
        if(kg.eq.1) go to 485
        lw1(nk)=lw1(nk)+1
           if(ipr.le.3)print*,' ql1 = ',ql1
        if(ql1.eq.0.)goto 491
        do 489 i=1,n
489     dq(i,nk)=dqh(i)/ql1*si
        go to 491
485     continue
```

```
         if(ipr.le.3)print*,' kn = ',kn
      kn=kn+1
         if(ipr.le.3)print*,' kn = ',kn
      do 486 i=1,n
      lw2(i)=lw2(i)+lw1(i)
  486 lw1(i)=0
         if(ipr.le.3)print*,' lw2 = ',lw2
         if(ipr.le.3)print*,' kn, np = ',kn,np
      if(kn.le.np) go to 481
      if(abs(detw).gt.1.e-3) go to 499
      kg=1
      go to 481
  499 continue
      nk=imax
      lw2(nk)=lw2(nk)+1
      det =detw
      if(ql1.eq.0.)call err('ident',499,490,'ql1',0,0,ql1)
      do 490 i=1,n
  490 aq(i,nk)=aqh(i)/ql1*si
  495 call sergo(nk,n,eps,det,nfi,
     * nksi,hpar,ipr,amu)
      kn=1
      do 488 i=1,n
      lw1(i)=lw2(i)
  488 lw2(i)=0
  491 continue
      bul=.true.
      do 550 i=1,n
      bul=bul.and.(abs(dqh(i)*hpar).le.epsi(i))
     .       .and.(abs(dqh(i)).le.epsi(i))
  550 continue
      if(bul) h0=2.
      ql(nk)=ql1*abs(hpar)
      if(ql(nk).eq.0.)goto 560
      do 530 i=1,nksi
  530 dksi(i,nk)=(ksinp1(i)-ksink(i))/ql(nk)
```

```
          do 540 i=1,nfi
  540     dfi(i,nk)=(finp1(i)-fink(i))/ql(nk)
c*
  560        continue
             if(ipr.le.3) write (o,*) 'nk=',nk
             if(ipr.le.2) call mxout('dq    ',dq,npar,n,n,0,60,132,1)
             if(ipr.le.2.and.nfi.gt.0)
     *       write (6,*) 'finp1=',(finp1(i),i=1,nfi)
             if(ipr.le.2)write (6,*) 'dfi=',((dfi(i,j),j=1,n),i=1,nfi)
             if(ipr.le.2) write (6,*) 'ksinp1=',(ksinp1(i),i=1,nksi)
             if(ipr.le.2) write (6,*) 'dksi(i,nk)=',(dksi(i,nk),i=1,nksi)
c            if(ipr.le.3) write (6,*) 'ql=',(ql(i),i=1,n)
             if(ipr.le.4) write (6,*) 'det=',det
     *       ,'w(imax)=',w(imax),'ql(nk)=',ql(nk)
             if(ipr.le.5) write (6,*) '                  hpar=',hpar
             if(ipr.le.7) write (6,*) 'qnp1=',(qnp1(i),i=1,n),'fpar=',fpar
             if(ipr.le.3) write (6,*) 'eps=',eps
          if(bul) go to 900
          if(kstop.ge.nstop) h0=3.
          if(kstop.ge.nstop) go to 900
          if(abs(det).ge.eps) go to 1000
          do 580 i=1,n
          si=1.
          if(dqh(i)*hpar.lt.0.) si=-1.
          delta(i)=alb*amax1(abs(hpar*dqh(i)),epsi(i))*si
  580     continue
          do 650 i=1,n
  650     q0(i)=qnp1(i)
             if(ipr.le.3) write (6,*) 'delta =',(delta(i),i=1,n)
          h0=h10
          go to 1100
c
  900     continue
          fpar=ff
          nstop=kstop
```

```
      do 800 i=1,n
      if(isub.eq.1)zq00(i)=qnp1(i)
800   zq0(i)=qnp1(i)
         if(ipr.le.4) write (6,*) 'krec=',krec
         if(ipr.le.8) write (6,*) 'qnp1=',(qnp1(i),i=1,n),'fpar=',fpar
         if(ipr.le.4.and.nfi.gt.0)
     *   write (6,*) 'finp1=',(finp1(i),i=1,nfi)
      alb=0.
      if(nksi.gt.n)  alb=sqrt(fpar/(nksi-n))
      if(isub.eq.0) return
      call subpq(n,nksi,ipr,nnp)
         if(ipr.le.3) call mxout('pq    ',pq,nnp,n,n,0,60,132,1)
      do 805 i=1,n
      do 805 j=1,n
805   zpq(i,j)=pq(i,j)
      return
      end
```

```
c
c     ...................................................................
c
c        subroutine array
c
c        purpose
c           convert data array from single to double dimension or vice
c           versa.  this subroutine is used to link the user program
c           which has double dimension arrays and the ssp subroutines
c           which operate on arrays of data in a vector fashion.
c
c        usage
c           call array (mode,i,j,n,m,s,d)
c
c        description of parameters
c           mode - code indicating type of conversion
c                     1 - from single to double dimension
c                     2 - from double to single dimension
c           i    - number of rows in actual data matrix
c           j    - number of columns in actual data matrix
c           n    - number of rows specified for the matrix d in
c                  dimension statement
c           m    - number of columns specified for the matrix d in
c                  dimension statement
c           s    - if mode=1, this vector is input which contains the
c                  elements of a data matrix of size i by j. column i+1
c                  of data matrix follows column i, etc. if mode=2,
c                  this vector is output representing a data matrix of
c                  size i by j containing its columns consecutively.
c                  the length of s is ij, where ij=i*j.
c           d    - if mode=1, this matrix of size n by m is output,
c                  containing a data matrix of size i by j in the first
c                  i rows and j columns. if mode=2, this n by m matrix
c                  is input containing a data matrix of size i by j in
c                  the first i rows and j columns.
c
```

```
c          remarks
c             vector s can be in the same location as matrix d.  vector s
c             is referred as a matrix in other ssp routines, since it
c             contains a data matrix.
c             this subroutine converts only general data matrices (storage
c             mode of 0).
c
c          subroutines and function subroutines required
c             none
c
c          method
c             refer to the discussion on variable data size in the section
c             describing overall rules for usage in this manual.
c
c       ..................................................................
c
        subroutine array (mode,i,j,n,m,s,d)
        dimension s(*),d(*)
c
        ni=n-i
c
c          test type of conversion
c
        if(mode-1) 100, 100, 120
c
c          convert from single to double dimension
c
    100 ij=i*j+1
        nm=n*j+1
        do 110 k=1,j
        nm=nm-ni
        do 110 l=1,i
        ij=ij-1
        nm=nm-1
    110 d(nm)=s(ij)
```

```
      go to 140
c
c        convert from double to single dimension
c
  120 ij=0
      nm=0
      do 130 k=1,j
      do 125 l=1,i
      ij=ij+1
      nm=nm+1
  125 s(ij)=d(nm)
  130 nm=nm+ni
c
  140 return
      end
```

```
c CONSTR   -    generates vector of constraints
c               CONSTR calls no subroutines (is needed by IDENT)
c
        subroutine constr(q,nq,fi,nfi)
        real q(nq),fi(nfi)
        return
        end
```

```
      subroutine detreq(dq0t,dq1t,dq0,a,b,ii,n)
      parameter (npar=20)
      real dq0t(npar,npar),dq1t(npar,npar),dq0(npar,npar)
      ,a(npar),b(npar),g(2,npar),f(npar,2)
      dimension w1(npar,npar),w2(2,2),ll(2),mm(2)
      aa=0.
      do 10 i=1,n
10    aa=a(i)*a(i)+aa
      do 21 i=1,ii
      bb=0.
      do 20 j=1,n
20    bb=bb+dq0(j,i)*a(j)
21    f(i,2)=bb+aa*b(i)
      do 30 i=1,ii
30    f(i,1)=b(i)
      do 40 i=1,ii
      bb=0.
      do 41 j=1,n
41    bb=bb+a(j)*dq0(j,i)
40    g(1,i)=bb
      do 50 i=1,ii
50    g(2,i)=b(i)
c************************************
      call matr(dq0t,npar,f,npar,w1,npar,ii,ii,2,1)
c************************************
      call matr(g,2,w1,npar,w2,2,2,ii,2,1)
c************************************
      do 60 i=1,2
60    w2(i,i)=w2(i,i)+1.
c************************************
```

```
           call minv(w2,2,detd,ll,mm)
c***************************************
           call matr(w1,npar,w2,2,dq1t,npar,ii,2,2,1)
c***************************************
           call matr(dq1t,npar,g,2,w1,npar,ii,2,ii,1)
c***************************************
           call matr(w1,npar,dq0t,npar,dq1t,npar,ii,ii,ii,1)
           do 70 i=1,ii
           do 70 j=1,ii
70         dq1t(i,j)=dq0t(i,j)-dq1t(i,j)
           return
           end
```

```
c*       09-jun-80
c*       last revised 18-dec-80
         subroutine dir(ebsfi,nksi,n,nfi,
     *   nw,mu,spaspb,np,ipr,isub,nnp)
c*
         parameter (npar=20,nmeas=1001,ncon=20)
         real ksi,mu
         dimension w4(npar,npar),w5(npar,npar),ll(npar),mm(npar),
     *   w8(npar)
         common/cid/epsi(npar),lw2(npar),w(npar),fi(ncon)
     .   ,fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
     .   ,ksi(nmeas),pksi(nmeas),dksi(nmeas,npar)
     .   ,zq0(npar),q0(npar),q(npar),pq(npar,npar)
     .   ,dq(npar,npar),dqh(npar)
c*
         n10=npar
         n50=npar
         nw=npar
         if(isub.eq.1) go to 1111
            if(ipr.le.3.or.ipr.eq.11)call ccc(131)
         do 131 i=1,n
         w8(i)=0.
         do 131 j=1,n
131      w4(i,j)=0.
1111     continue
            if(ipr.le.3.or.ipr.eq.11)call ccc(1111)
         if(isub.eq.0) go to 133
         call matr(dq,npar,pq,nnp,w5,npar,n,n,n,2)
            if(ipr.le.3.or.ipr.eq.11)call ccc(121)
         do 121 i=1,n
         aa=0.
            if(ipr.le.3.or.ipr.eq.11)call ccc(120)
         do 120 j=1,n
120      aa=aa+w5(i,j)*(q0(j)-q(j))
121      w8(i)=aa
         if(ipr.le.2) call mxout('w8    ',w8,1,n,1,0,60,132,1)
         call matr(w5,npar,dq,npar,w4,npar,n,n,n,1)
         if(ipr.le.2) call mxout('w4 1  ',w4,npar,n,n,0,60,132,1)
```

```
133   continue
         if(ipr.le.3.or.ipr.eq.11)call ccc(10)
      do 10 i=1,n
      do 10 j=1,n
      aa=0.
      do 20 k=1,nksi
20    aa=aa+dksi(k,j)*dksi(k,i)*pksi(k)
10    w5(j,i)=aa+w4(j,i)
c*
      if(nfi.eq.0) go to 110
      aa=0.
         if(ipr.le.3.or.ipr.eq.11)call ccc(100)
      do 100 i=1,n
100   aa=aa+w5(i,i)
c*
      call matr(dfi,n10,dfi,n10,w4,nw,n,nfi,n,2)
      if(ipr.le.2) call mxout('w4 dfi',w4,npar,n,n,0,60,132,1)
      ab=0.
         if(ipr.le.3.or.ipr.eq.11)call ccc(200)
      do 200 i=1,n
200   ab=ab+mu*w4(i,i)
      spaspb=ab/aa
      if(spaspb.ge.ebsfi) return
c*
         if(ipr.le.3.or.ipr.eq.11)call ccc(300)
      do 300 i=1,n
      do 300 j=1,n
300   w5(i,j)=w5(i,j)+mu*w4(i,j)
110   continue
         if(ipr.le.3.or.ipr.eq.11)call ccc(110)
c*
      call array(2,n,n,n50,n50,w4,w5)
      cnorm=1.
         if(ipr.le.3.or.ipr.eq.11)call ccc(310)
      do 310 i=1,n
      w5ii=abs(w5(i,i))
```

```
          plumin=w5ii/w5(i,i)
          if(w5ii.ne.0.)cnorm=cnorm*sqrt(w5ii)*plumin
  310     continue
          if(ipr.le.2) call mxout('w4     ',w4,n,n,n,0,60,132,1)
          call minv(w4,n,det,ll,mm)
          if(ipr.le.2) call mxout('w4 inv',w4,n,n,n,0,60,132,1)
          call array(1,n,n,n50,n50,w4,w5)
          if(abs(cnorm).lt.1.e-15) go to 111
          sqrdet=sqrt(abs(det))
          if(sqrdet.le.abs(cnorm)*1.e-10.and.ipr.le.8)
          write(6,*)'              ---bad observability'
          if(ipr.le.4) write (6,*) 'dir det/cnorm=',sqrdet/cnorm
  111     continue
             if(ipr.le.3.or.ipr.eq.11)call ccc(111)
          if(ipr.le.2) write (6,*) 'cnorm=',cnorm**2.,'dir det=',det
c*
c*
             if(ipr.le.3.or.ipr.eq.11)call ccc(30)
          do 30 i=1,n
          aa=0.
          do 40 j=1,nksi
  40      aa=aa+dksi(j,i)*ksi(j)*pksi(j)
  30      w4(i,2)=aa-w8(i)
          if(ipr.le.2) call mxout('w4(1,2',w4(1,2),1,n,1,0,60,132,1)
          if(nfi.eq.0) go to 210
c*
          call matr(dfi,n10,fi(1),n10,w4(1,1),nw,n,nfi,1,2)
c*
             if(ipr.le.3.or.ipr.eq.11)call ccc(500)
          do 500 i=1,n
  500     w4(i,2)=w4(i,2)+mu*w4(i,1)
```

```
210     continue
           if(ipr.le.3.or.ipr.eq.11)call ccc(210)
c*
        call matr(w5,n50,w4(1,2),nw,w,n10,n,n,1,1)
c*
c*
           if(ipr.le.3.or.ipr.eq.11)call ccc(400)
        do 400 i=1,n
        dqh(i)=0.
        do 400 j=1,n
400     dqh(i)=dqh(i)-dq(i,j)*w(j)
        v=1.
        return
        end
```

```
c*          created 15-dec-80
c*  last    revised 24-dec-80
c*
            subroutine dir1(gnk,
       *    nksi,n,nfi,nw,mu,spaspb,np,ipr,
       *    isub,nnp,aifi,
       *    nfi0,n0,k31,amu,amu0)
c*
            parameter (npar=20,nmeas=1001,ncon=20)
            real ksinp1,l
            dimension w4(npar,npar),w5(npar,npar),ll(npar),mm(npar),
       *    w8(npar),ww(npar),wf(npar),wb(npar),
       *    aifi(npar,npar),w1(npar,npar),w2(npar,npar),dan(npar,npar),
       *    w3(npar,npar),
       *    ddfi(ncon,npar),r(npar,npar),s(npar,npar),l(npar,ncon),wc(npar)
            common/cia/epsi(npar),lw2(npar),w(npar)
       .    ,finp1(ncon),fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
       .    ,ksinp1(nmeas),pksi(nmeas),dksi(nmeas,npar)
       .    ,zq0(npar),q0(npar),qnp1(npar),pq(npar,npar)
       .    ,dq(npar,npar),dqh(npar)
c*          comput of (31')
            if(isub.eq.1) go to 1111
            do 131 i=1,n
            w8(i)=0.
            do 131 j=1,n
 131        w4(i,j)=0.
 1111       continue
            if(isub.eq.0) go to 133
            call matr(dq,npar,pq,nnp,w5,npar,n,n,n,2)
            do 121 i=1,n
            aa=0.
            do 120 j=1,n
 120        aa=aa+w5(i,j)*(q0(j)-qnp1(j))
 121        w8(i)=aa
            call matr(w5,npar,dq,npar,w4,npar,n,n,n,1)
 133        continue
```

```
c*
          if(k31.eq.1) go to 777
          if(ipr.le.3) write (6,*) '*******  1    '
          if(ipr.le.3) write (6,*) '*******31     '
          call matr(dfi0,ncon,dfi0,ncon,w1,npar,n,nfi0,n,2)
          if(ipr.le.2) call mxout('w1 1 ',w1,npar,n,n,0,60,132,1)
          do 10 i=1,n
          do 10 j=1,n
          aa=0.
          do 20 k=1,nksi
20        aa=aa+dksi(k,j)*dksi(k,i)*pksi(k)
10        dan(j,i)=aa+w4(j,i)+w1(j,i)*amu0
          if(ipr.le.2) call mxout('dan 1 ',dan,npar,n,n,0,60,132,1)
c***************************
          call array(2,n,n,npar,npar,w2,dan)
          call minv(w2,n,det,ll,mm)
          call array(1,n,n,npar,npar,w2,w1)
          if(ipr.le.2) call mxout('w1 i 1',w1,npar,n,n,0,60,132,1)
          if(ipr.le.2) write(6,*) 'det=',det
c***************************
          if(ipr.le.2) call mxout('w2 nfi',w2,npar,nfi0,nfi0,0,60,132,1)
          call array(2,nfi0,nfi0,npar,npar,w3,w2)
          call minv(w3,nfi0,det,ll,mm)
          call array(1,nfi0,nfi0,npar,npar,w3,w2)
          if(ipr.le.2) call mxout('w2infi',w2,npar,nfi0,nfi0,0,60,132,1)
          if(ipr.le.2) write(6,*) 'det=',det
          call matr(w2,npar,dfi0,ncon,ddfi,ncon,nfi0,nfi0,n,2)
          if(ipr.le.2) call mxout('ddfi  ',ddfi,ncon,n,nfi0,0,60,132,1)
c***************************
          call matr(ddfi,ncon,ddfi,ncon,w2,npar,n,nfi0,n,2)
          if(ipr.le.2) call mxout('w2 dd  ',w2,npar,n,n,0,60,132,1)
c*
c*
c*           comp of  (dan+dv*d'd)
c*
          do 2 i=1,n
          do 2 j=1,n
```

```
2       w2(i,j)=w2(i,j)*1./(amu-amu0)
        w4(i,j)=dan(i,j)+w2(i,j)
        if(ipr.le.2) call mxcut('w4 11',w4,npar,n,n,0,60,132,1)
c*
c*                                       -1
c*         comp. of (dan+dv*d'd)
c*
        call array(2,n,n,npar,npar,w3,w4)
        call minv(w3,n,det,ll,mm)
        call array(1,n,n,npar,npar,w3,w4)
        if(ipr.le.2) write(6,*) 'det=',det
        if(ipr.le.2) call mxcut('w4i 11',w4,npar,n,n,0,60,132,1)
c*
c*
c*         comp. of d'*fi0
c*
        call matr(ddfi,ncon,fi0,ncon,w3,npar,n,nfi0,1,2)
c*
c*                                        -1
c*         comp. of  (dan+dv*d'd)   d'*fi0
c*
        call matr(w4,npar,w3,npar,ww,npar,n,n,1,1)
c*
c*
c*         comp. of  dfi0'*fi0
c*
        call matr(dfi0,ncon,fi0,ncon,wf,npar,n,nfi0,1,2)
        do 3 i=1,n
3       wf(i)=amu0*wf(i)
c*
c*
c*         comp. of dksi'*pksi*ksinp1+amu0*dfi0'*fi0+dq'*(q0-q)
c*
        if(ipr.le.2) write (6,*) 'fi0(i)=',(fi0(i),i=1,nfi0)
        do 4 i=1,n
        aa=0.
        do 5 j=1,nksi
```

```
5       aa=aa+dksi(j,i)*ksinp1(j)*pksi(j)
4       wf(i)=aa+wf(i)-w8(i)
c************************************
c*
c*            comp. of 3 summ
c*

        call matr(w4,npar,wf,npar,wc,npar,n,n,1,1)
        call matr(w2,npar,wc,npar,w3(1,1),npar,n,n,1,1)
        call matr(w1,npar,w3(1,1),npar,wb,npar,n,n,1,1)
c*
c*            comp. of 2 summ
c*
        call matr(w1,npar,wf,npar,w3(1,1),npar,n,n,1,1)
        call matr(ddfi,ncon,w3(1,1),npar,w3(1,2),npar,nfi0,n,1,1)
        call matr(dfi0,ncon,w3(1,2),npar,w3(1,3),npar,n,nfi0,1,2)
        call matr(w1,npar,w3(1,3),npar,w3(1,1),npar,n,n,1,1)
c*****************************************
c*
c*            comp. of [( )-( )+( )+( )]
c*
        do 6 i=1,n
6       w(i)=wc(i)-w3(i,1)+wb(i)+ww(i)
        if(ipr.le.2) write (6,*) 'w3(i,1)=',(w3(i,1),i=1,n)
        if(ipr.le.2) write (6,*) 'ww(i)=',(ww(i),i=1,n)
        if(ipr.le.2) write (6,*) 'wb(i)=',(wb(i),i=1,n)
        if(ipr.le.2) write (6,*) 'wc(i)=',(wc(i),i=1,n)
c*
c*            comp. of direction dqh
c*
        do 7 i=1,n
        aa=0.
        do 8 j=1,n
8       aa=aa-dq(i,j)*w(j)
7       dqh(i)=aa
        i=0
```

```
           return
c*
            end of comp (31')
c*
 777       continue
           if(ipr.le.3) write (6,*) '*******  11  '
           if(ipr.le.3) write (6,*) '*******31    '
c*
            comp. of (31'')
c*
               comp dfi0'*dfi0
c*
           call matr(dfi0,ncon,dfi0,ncon,w5,npar,n0,nfi,n0,2)
c*
c*
               comp. a
c*
           call matr(w5,npar,aifi,npar,w3,npar,n0,n0,n,1)
           call matr(aifi,npar,w3,npar,w1,npar,n,n0,n,2)
c*
               comp. [dksi'pksi dksi+...]
c*
           do 210 i=1,n
           do 210 j=1,n
           aa=0.
           do 220 k=1,nksi
 220       aa=aa+dksi(k,j)*dksi(k,i)*pksi(k)
 210       dan(j,i)=aa+w4(j,i)+w1(j,i)*amu0
c*
c*                                   -1
c*            comp of (dfi0'dfi0)
c*
           call array(2,n0,n0,npar,npar,w1,w5)
           call minv(w1,n0,det,11,mm)
```

```
          call array(1,n0,n0,npar,npar,w1,w2)
          if(ipr.le.2) write(6,*) 'det=',det
c*
c*                     -1
c*          comp of a
c*
          call array(2,n,n,npar,npar,w5,dan)
          call minv(w5,n,det,ll,mm)
          call array(1,n,n,npar,npar,w5,w1)
          if(ipr.le.2) write(6,*) 'det=',det
c*
c*                            -1      -1
c*          comp    (aifi*a *aifi')
c*
          call matr(w1,npar,aifi,npar,w4,npar,n,n,n0,3)
          call matr(aifi,npar,w4,npar,w3,npar,n0,n,n0,1)
          call array(2,n0,n0,npar,npar,w4,w3)
          call minv(w4,n,det,ll,mm)
          call array(1,n0,n0,npar,npar,w4,w3)
          if(ipr.le.2) write(6,*) 'det=',det
c*
c*          comp of  r
c*
          call matr(w3,npar,aifi,npar,w4,npar,n0,n0,n,1)
          call matr(aifi,npar,w4,npar,r,npar,n,n0,n,2)
c*
c*          comp of  s
c*
          call matr(w2,npar,w4,npar,w3,npar,n0,n0,n,1)
          call matr(w4,npar,w3,npar,s,npar,n,n0,n,2)
c*
c*          comp of  l
c*
          call matr(w4,npar,w2,npar,w3,npar,n,n0,n0,2)
          call matr(w3,npar,dfi0,ncon,l,npar,n,n0,nfi,3)
c*
```

```
c*          comp a+v*s
c*
        do 202 i=1,n
        do 202 j=1,n
        s(i,j)=s(i,j)*1./(amu-amu0)
202     w2(i,j)=dan(i,j)+s(i,j)
c*
c*                          -1
c*          comp of (a+v*s)
c*
        call array(2,n,n,npar,npar,w3,w2)
        call minv(w3,n,det,l1,mm)
        call array(1,n,n,npar,npar,w3,w2)
        if(ipr.le.2) write(6,*) 'det=',det
c*
c*            dfi0'*finp1
c*
        call matr(dfi0,ncon,finp1,ncon,wc,npar,n0,nfi,1,2)
        call matr(aifi,npar,wc,npar,wb,npar,n,n0,1,2)
        do 204 i=1,n
        aa=0.
        do 205 j=1,nksi
205     aa=aa+dksi(j,i)*ksinp1(j)*pksi(j)
204     wb(i)=aa+wb(i)*amu0-w8(i)
c*
c*             l*finp1
c*
        call matr(l,npar,finp1,ncon,wf,npar,n,nfi,1,1)
c*
c*                    -1
c*             (a+v*s)*l*finp1
c*
        call matr(w2,npar,wf,npar,wc,npar,n,n,1,1)
c*
c*             -1         -1
c*            a *s*(     ) [      ]
```

```
c*
           call matr(w2,npar,wb,npar,wf,npar,n,n,1,1)
           call matr(s,npar,wf,npar,ww,npar,n,n,1,1)
           call matr(w1,npar,ww,npar,w3(1,1),npar,n,n,1,1)
c*
c*              -1    -1
c*             a   *r*a [           ]
c*
           call matr(w1,npar,wb,npar,wf,npar,n,n,1,1)
           call matr(r,npar,wf,npar,ww,npar,n,n,1,1)
           call matr(w1,npar,ww,npar,w3(1,2),npar,n,n,1,1)
c*
c*
c*
           do 214 i=1,n
214        w(i)=w3(i,1)-w3(i,2)+wf(i)+wc(i)
           if(ipr.le.2) write (6,*) 'w3(i,1)=',(w3(i,1),i=1,n)
           if(ipr.le.2) write (6,*) 'w3(i,2)=',(w3(i,2),i=1,n)
           if(ipr.le.2) write (6,*) 'wf(i)=',(wf(i),i=1,n)
           if(ipr.le.2) write (6,*) 'wc(i)=',(wc(i),i=1,n)
           do 207 i=1,n
           aa=0.
           do 208 j=1,n
208        aa=aa-dq(i,j)*w(j)
207        dqh(i)=aa
           i=1
           return
           end
```

```
      subroutine funct(q,ksi,pksi,fi,ff,amu,nfi,nksi,n,np)
      dimension q(*),ksi(*),fi(*),pksi(*)
      real ksi
      call resid(q,nksi,n,ksi)
      ff=0
      a=0.
      do 12 j=1,nksi
      if(ksi(j).gt.1.e19)goto 2000
      if(1.e38-a.le.ksi(j)*ksi(j))goto 2000
12    a=a+ksi(j)*pksi(j)*ksi(j)
      goto 2005
2000  a=1.e38
2005  ff=a
      if(nfi.eq.0) return
      call constr(q,n,fi,nfi)
      do 20 i=1,nfi
      ff=ff+amu*fi(i)**2
20    continue
      return
      end
```

```
      subroutine functa(n,q,q0,ff,pq,nnp)
      dimension q(*),q0(*),pq(nnp,*)
      fff=0.
      do 40 i=1,n
      aa=0.
      do 50 j=1,n
50    aa=aa+pq(i,j)*(q0(j)-q(j))
40    fff=fff+aa*(q0(i)-q(i))
      ff=ff+fff
      return
      end
```

```
c          subroutine loc
c
c          purpose
c             compute a vector subscript for an element in a matrix of
c             specified storage mode
c
c          usage
c             call loc (i,j,ir,n,m,ms)
c
c          description of parameters
c             i    - row number of element
c             j    - column number  of element
c             ir   - resultant vector subscript
c             n    - number of rows in matrix
c             m    - number of columns in matrix
c             ms   - one digit number for storage mode of matrix
c                    0 - general
c                    1 - symmetric
c                    2 - diagonal
c
c          remarks
c             none
c
c          subroutines and function subprograms required
c             none
c
c          method
c             ms=0   subscript is computed for a matrix with n*m elements
c                    in storage (general matrix)
```

```
c             ms=1   subscript is computed for a matrix with n*(n+1)/2 in
c                    storage (upper triangle of symmetric matrix). if
c                    element is in lower triangular portion, subscript is
c                    corresponding element in upper triangle.
c             ms=2   subscript is computed for a matrix with n elements
c                    in storage (diagonal elements of diagonal matrix).
c                    if element is not on diagonal (and therefore not in
c                    storage), ir is set to zero.
c
c     ...............................................................
c
      subroutine loc(i,j,ir,n,m,ms)
c
      ix=i
      jx=j
      if(ms-1) 10,20,30
   10 irx=n*(jx-1)+ix
      go to 36
   20 if(ix-jx) 22,24,24
   22 irx=ix+(jx*jx-jx)/2
      go to 36
   24 irx=jx+(ix*ix-ix)/2
      go to 36
   30 irx=0
      if(ix-jx) 36,32,36
   32 irx=ix
   36 ir=irx
      return
      end
```

```
      subroutine matr(a,na,b,nb,c,nc,n,m,k,npr)
      dimension a(na,m),b(nb,k),c(nc,k)
      do 1 i=1,n
      do 1 j=1,k
    1 c(i,j)=0.
      go to (2,3,4,5) npr
    2 continue
      do 21 j=1,k
      do 21 i=1,n
      do 21 k1=1,m
   21 c(i,j)=c(i,j)+a(i,k1)*b(k1,j)
      return
    3 continue
      do 31 j=1,k
      do 31 i=1,n
      do 31 k1=1,m
   31 c(i,j)=c(i,j)+a(k1,i)*b(k1,j)
      return
    4 continue
      do 41 j=1,k
      do 41 i=1,n
      do 41 k1=1,m
   41 c(i,j)=c(i,j)+a(i,k1)*b(j,k1)
      return
    5 continue
      do 51 j=1,k
      do 51 i=1,n
      do 51 k1=1,m
   51 c(i,j)=c(i,j)+a(k1,i)*b(j,k1)
      return
      end
```

```
c
c     ..................................................................
c
c        subroutine minv
c
c        purpose
c           invert a matrix
c
c        usage
c           call minv(a,n,d,l,m)
c
c        description of parameters
c           a - input matrix, destroyed in computation and replaced by
c               resultant inverse.
c           n - order of matrix a
c           d - resultant determinant
c           l - work vector of length n
c           m - work vector of length n
c
c        remarks
c           matrix a must be a general matrix
c
c        subroutines and function subprograms required
c           none
c
c        method
c           the standard gauss-jordan method is used. the determinant
c           is also calculated. a determinant of zero indicates that
c           the matrix is singular.
c
c     ..................................................................
c
      subroutine minv(a,n,d,l,m)
      dimension a(*),l(*),m(*)
c
c        ...............................................................
c
c        if a double precision version of this routine is desired, the
```

```
c          c in column 1 should be removed from the double precision
c          statement which follows.
c
c       double precision a,d,biga,hold
c
c          the c must also be removed from double precision statements
c          appearing in other routines used in conjunction with this
c          routine.
c
c          the double precision version of this subroutine must also
c          contain double precision fortran functions.  abs in statement
c          10 must be changed to dabs.
c
c          ..............................................................
c
c          search for largest element
c
        d=1.0
        nk=-n
        do 80 k=1,n
        nk=nk+n
        l(k)=k
        m(k)=k
        kk=nk+k
        biga=a(kk)
        do 20 j=k,n
        iz=n*(j-1)
        do 20 i=k,n
        ij=iz+i
     10 if( abs(biga)- abs(a(ij))) 15,20,20
     15 biga=a(ij)
        l(k)=i
        m(k)=j
     20 continue
          if(biga.eq.0.)
          call err('minv',20,25,'biga',0,0,biga)
```

```
c
c         interchange rows
c
21        j=l(k)
       if(j-k) 35,35,25
    25 ki=k-n
       do 30 i=1,n
       ki=ki+n
       hold=-a(ki)
       ji=ki-k+j
       a(ki)=a(ji)
    30 a(ji) =hold
c
c         interchange columns
c
    35 i=m(k)
       if(i-k) 45,45,38
    38 jp=n*(i-1)
       do 40 j=1,n
       jk=nk+j
       ji=jp+j
       hold=-a(jk)
       a(jk)=a(ji)
    40 a(ji) =hold
c
c         divide column by minus pivot (value of pivot element is
c         contained in biga)
c
    45 if(biga) 48,46,48
    46 d=0.0
       return
    48 do 55 i=1,n
       if(i-k) 50,55,50
    50 ik=nk+i
       a(ik)=a(ik)/(-biga)
    55 continue
```

```
c
c          reduce matrix
c
        do 65 i=1,n
        ik=nk+i
        hold=a(ik)
        ij=i-n
        do 65 j=1,n
        ij=ij+n
        if(i-k) 60,65,60
     60 if(j-k) 62,65,62
     62 kj=ij-i+k
        a(ij)=hold*a(kj)+a(ij)
     65 continue
c
c          divide row by pivot
c
        kj=k-n
        do 75 j=1,n
        kj=kj+n
        if(j-k) 70,75,70
     70 a(kj)=a(kj)/biga
     75 continue
c
c          product of pivots
c
        d=d*biga
c
c          replace pivot by reciprocal
c
        a(kk)=1.0/biga
     80 continue
c
c          final row and column interchange
c
        k=n
```

```
100 k=(k-1)
    if(k) 150,150,105
105 i=l(k)
    if(i-k) 120,120,108
108 jq=n*(k-1)
    jr=n*(i-1)
    do 110 j=1,n
    jk=jq+j
    hold=a(jk)
    ji=jr+j
    a(jk)=-a(ji)
110 a(ji) =hold
120 j=m(k)
    if(j-k) 100,100,125
125 ki=k-n
    do 130 i=1,n
    ki=ki+n
    hold=a(ki)
    ji=ki-k+j
    a(ki)=-a(ji)
130 a(ji) =hold
    go to 100
150 return
    end
```

```
c
c      ................................................................
c
c         subroutine mxout
c
c         purpose
c            produces an output listing of any sized array on
c            logical unit 6
c
c         usage
c            call mxout(icode,a,n,m,nprint,ms,lins,ipos,isp)
c
c         description of parameters
c            icode- input code number to be printed on each output page
c            a-name of output matrix
c            n-number of rows in a
c            m-number of columns in a
c            nprint-number of rows to print
c            ms-storage mode of a where ms=
c                   0-general
c                   1-symmetric
c                   2-diagonal
c            lins-number of print lines on the page (usually 60)
c            ipos-number of print positions across the page (usually 132)
c            isp-line spacing code, 1 for single space, 2 for double
c                space
c
c         remarks
c            none
c
c         subroutines and function subprograms required
c            loc
c
c         method
c            this subroutine creates a standard output listing of any
c            sized array with any storage mode. each page is headed with
c            the code number,dimensions and storage mode of the array.
c            each column and row is also headed with its respective
```

```
c              number
c
c       ..................................................................
c
        subroutine mxout(icode,a,n,m,nprint,ms,lins,ipos,isp)
        character*6 icode
        dimension a(*),b(8)
      1 format(1h ,/5x,  7hmatrix ,a6,5x,i4,5h rows,5x,i3,4h col,
       13x,9hstor mod ,i1,3x,/)
      2 format(12x,8hcolumn  ,7(3x,i3,10x))
      3 format(1h )
      4 format(1h ,7x,4hrow ,i3,7e16.6)
      5 format(1h0,7x,4hrow ,i3,7e16.6)
c
        j=1
c
c                 write heading
c
        nend=ipos/16-1
        lend=(lins/isp)-2
        ipage=1
     10 lstrt=1
ccc       call ccc(10)
     20 if(ipage.eq.1) write(6,1)icode,n,m,ms
ccc       call ccc(20)
        jnt=j+nend-1
        ipage=ipage+1
ccc       call ccc(31)
     31 if(jnt-m)33,33,32
     32 jnt=m
ccc       call ccc(32)
     33 continue
ccc       call ccc(33)
        write(6,2)(jcur,jcur=j,jnt)
        if(isp-1) 35,35,40
     35 write(6,3)
```

```
ccc      call ccc(35)
     40 ltend=lstrt+lend-1
ccc      call ccc(40)
ccc      call ccc(80)
        do 30 l=lstrt,ltend
c
c          form output row line
c
ccc      call ccc(55)
        do 55 k=1,nend
        kk=k
        jt = j+k-1
        call loc(l,jt,ijnt,n,m,ms)
        b(k)=0.0
        if(ijnt)50,50,45
     45 b(k)=a(ijnt)
ccc      call ccc(45)
     50 continue
ccc      call ccc(50)
c
c          check if last column.  if yes go to 60
c
        if(jt-m) 55,60,60
     55 continue
c
c          end of line, now write
c
     60 if(isp-1)65,65,70
     65 write(6,4)l,(b(jw),jw=1,kk)
ccc      call ccc(65)
        go to 75
     70 write(6,5)l,(b(jw),jw=1,kk)
ccc      call ccc(70)
c
c          if end of rows,go check columns
c
     75 if(nprint-1)85,85,80
```

```
   80 continue
ccc     call ccc(80)
c
c          end of page, now check for more output
c
        lstrt=lstrt+lend
        go to 20
c
c          end of columns, then return
c
   85 if(jt-m)90,95,95
   90 j=jt+1
ccc     call ccc(90)
        go to 10
   95 continue
ccc     call ccc(95)
        end
```

```
            subroutine ort(dq0,dqCt,aqi,aet1,detC,dqort,ii,n,nk,anorm)
c*
            parameter (npar=20)
            real dq0(npar,npar),dq0t(npar,npar),dqi(npar)
            ,dqort(npar),w1(npar),w2(npar)
            call matr(dq0,npar,dqi,npar,w1,npar,ii,n,1,2)
            call matr(dq0t,npar,w1,npar,w2,npar,ii,ii,1,1)
            call matr(dq0,npar,w2,npar,w1,npar,n,ii,1,1)
            a=0.
            do 10 i=1,n
            dqort(i)=dqi(i)-w1(i)
   10       a=a+dqort(i)**2
            anorm=sqrt(a)
            det1=det0/anorm
            return
            end
```

```
      subroutine sergo(nk,n,eps,det,nfi,nksi,hpar,ipr,amu)
      parameter (npar=20,nmeas=1001,ncon=20)
      real ksinp1
      dimension dq0(npar,npar),dq0t(npar,npar),dqort0(npar),l(npar),
     * dqmax(npar),a(npar),b(npar),dq1(npar,npar),dq1t(npar,npar),
     * lll(npar),
     * dqort1(npar),ll(npar),mmm(npar),w(npar,npar),nn(npar)
      common/cid/epsi(npar),lw2(npar),zw(npar),finp1(ncon)
     . ,fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
     . ,ksinp1(nmeas),pksi(nmeas),dksi(nmeas,npar)
     . ,q0(npar),q00(npar),qnp1(npar),pq(npar,npar)
     . ,dq(npar,npar),dqh(npar)
      icount=0
      np=2
      kk=0
      do 2 i=1,n
2     l(i)=0
1     continue
      knn0=0
      kk=kk+1
      if(kk.gt.1)mm=nn(max)
      if(max.eq.nn0) knn0=1
      do 11 i=1,n
      lll(i)=nn(i)
      b(i)=0.
11    continue
13    continue
      kpr=0
      do 3 i=1,n
3     nn(i)=0
      ii=0
      do 10 i=1,n
      if(l(i).eq.1) go to 10
      if(i.eq.nk) go to 40
      if(kpr.eq.1) go to 40
      if(icount.eq.1) go to 20
      if(lw2(i).eq.0) go to 20
40    ii=ii+1
```

```
30      do 30 j=1,n
        dq0(j,ii)=dq(j,i)
        nn(i)=ii
        go to 50
20      kpr=1
        nn0=i
50      continue
10      continue
        if(ipr.le.3) write(6,*) 'ii=',ii,'kpr=',kpr,'icount=',icount
        if(ipr.le.3) write(6,*) 'nk=',nk
        if(ipr.le.3) write(6,*) 'lw2(i)=',(lw2(i),i=1,n)
        if(ipr.le.3) write(6,*) 'nn0=',nn0
        if(ipr.le.3) write(6,*) 'nn(i)=',(nn(i),i=1,n)
        if(ii.ge.1.and.kpr.ne.0.or.icount.eq.1) go to 14
        icount=1
        go to 13
14      continue
        if(knn0.eq.1)mm=lll(nn0)
        if(kk.gt.1) go to 12
        call matr(dq0,npar,dq0,npar,dq0t,npar,ii,n,ii,2)
        call array(2,ii,ii,npar,npar,w,dq0t)
        call minv(w,ii,detdet,ll,mmm)
        call array(1,ii,ii,npar,npar,w,dq0t)
        if(ipr.le.3) write(6,*) 'detdet=',detdet
        go to 17
12      continue
        if(mm.eq.ii+1) go to 500
        do 300 i=1,ii+1
        if(i.ne.mm) go to 301
        do 302 iv=1,ii+1
302     dqmax(iv)=dq0t(iv,mm)
301     continue
        if(i.le.mm) go to 300
        do 400 j=1,ii+1
        dq0t(j,i-1)=dq0t(j,i)
400     continue
```

```
300   continue
      do 306 i=1,ii+1
306   dq0t(i,ii+1)=dqmax(i)
      do 310 i=1,ii+1
      if(i.ne.mm) go to 311
      do 312 iv=1,ii+1
312   dqmax(iv)=dq0t(mm,iv)
311   continue
      if(i.le.mm) go to 310
      do 340 j=1,ii+1
      dq0t(i-1,j)=dq0t(i,j)
340   continue
310   continue
      do 350 i=1,ii+1
350   dq0t(ii+1,i)=dqmax(i)
500   continue
      do 600 j=1,ii
      do 600 i=1,ii
      aa=dq0t(j,ii+1)*dq0t(i,ii+1)/dq0t(ii+1,ii+1)
600   dq0t(j,i)=dq0t(j,i)-aa
17    call ort(dq0,dq0t,dq(1,nn0),det0,det,dqort0,ii,n,nk,anorm)
      if(ipr.le.4) write(6,*) 'anorm=',anorm
      if(ipr.le.3) write(6,1000) (dqort0(i),dq(i,nn0),i=1,n)
1000  format(5x,' dqort0(i) ',2x,' dq(i,nn0) ',/,(5x,g11.5,2x,g11.5))
      if(ipr.le.3) write(6,*) 'dq0(i,j)='
      do 1100 i=1,n
1100  if(ipr.le.3) write(6,1001) (dq0(i,j),j=1,ii)
1001  format(x,6g12.5)
      if(ipr.le.3) write(6,*) 'dq0t(i,j)='
      do 1110 i=1,ii
1110  if(ipr.le.3) write(6,1001) (dq0t(i,j),j=1,ii)
      if(ipr.le.3) write(6,*) 'det=',det,'det0=',det0
      if(ipr.le.3) write(6,*) '**********************************'
      if(abs(det0).lt.eps) go to 100
      lw2(nn0)=lw2(nn0)+1
      np=1
```

```
      call step(hpar,dqort0,anorm,n,np,nn0,nksi,ipr,amu,nfi)
      det=det0
      return
c***********************************
100   continue
      anormm=anorm
      detm=det0
      max=nn0
      do 120 i=1,n
120   dqmax(i)=dqort0(i)
      nni=nn0
c*************************************
      nnip1=nn0+1
      do 60 i=nnip1,n
      if(l(i).eq.1.or.i.eq.nk.or.i.gt.n) go to 60
      if(lw2(i).ne.0.and.icount.eq.0) go to 60
70    continue
      do 80 j=1,n
80    a(j)=dq(j,nn0)-dq(j,i)
      b(nn(i))=1.
      if(nn(nni).le.0.and.ipr.ne.9)
      call err('sergo',80,81,'nn',nni,0,nn(nni))
      if(nn(nni).gt.0)b(nn(nni))=0.
      nni=i
      do 31 iv=1,n
      do 31 iw=1,ii
31    dq1(iv,iw)=dq0(iv,iw)
      do 32 iv=1,n
32    dq1(iv,nn(nni))=dq(iv,nn0)
      call detreq(dq0t,dq1t,dq0,a,b,ii,n)
      if(ipr.le.3) write(6,*) 'dq1t(i,j)='
      do 1111 ih=1,ii
1111  if(ipr.le.3) write(6,1001) (dq1t(ih,j),j=1,ii)
      if(ipr.le.3) write(6,*) 'a(i)='
      if(ipr.le.3) write(6,1001) (a(ih),ih=1,n)
      if(ipr.le.3) write(6,*) 'b(i)='
```

```
      if(ipr.le.3) write(6,1001) (b(ih),ih=1,n)
      call ort(dq1,dq1t,dq(1,nni),det1,det,dqort1,ii,n,nk,anorm)
      if(ipr.le.3) write(6,*) 'nni=',nni
      if(ipr.le.3) write(6,*) 'det1=',det1,'anorm=',anorm
      if(abs(det1).gt.eps) go to 140
      if(abs(det1).lt.abs(detm)) go to 60
      anormm=anorm
      detm=det1
      max=nni
      do 155 in=1,n
155   dqmax(in)=dqort1(in)
      go to 160
140   lw2(nni)=lw2(nni)+1
      call step(hpar,dqort1,anorm,n,np,nni,nksi,ipr,amu,nfi)
      det=det1
      return
160   continue
60    continue
200   continue
      call step(hpar,dqmax,anormm,n,np,max,nksi,ipr,amu,nfi)
      det=detm
      l(max)=1
      lw2(max)=lw2(max)+1
      if(ipr.le.3) write(6,*) 'max=',max
      if(ipr.le.3) write(6,*) 'l(i)=',(l(i),i=1,n)
      if(kk.lt.n-2) go to 1
      if(ipr.le.3) write(6,*) '****det=',det
      det=1
      return
      end
```

```
c*      last revised 23-dec-1980
        subroutine step(hpar,dqort,
     *   anorm,n,np,nn0,nksi,ipr,amu,nfi)
c*
        parameter (npar=20,nmeas=1001,ncon=20)
        real ksinp1
        real dqort(npar)
     .  ,qn(npar),ksin(nmeas),fin(ncon)
        common/cid/epsi(npar),lw2(npar),w(npar),finp1(ncon)
     .  ,fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
     .  ,ksinp1(nmeas),pksi(nmeas),dksi(nmeas,npar)
     .  ,q0(npar),q00(npar),qnp1(npar),pq(npar,npar)
     .  ,dq(npar,npar),dqh(npar)
        beta=.8
        amin=1.e30
        do 102 i=1,n
        if(abs(dqort(i)).lt.1.e-20) go to 102
        f=abs(hpar*dqh(i))/abs(dqort(i))
        hp=amax1(f,beta*epsi(i))
        if(hp.ge.amin) go to 102
        amin=hp
102     continue
        hp=amin
        aa=0.
        do 103 i=1,n
```

```
103   aa=aa+dqort(i)*dqh(i)
      if(aa.lt.0.) hp=-hp
      si=1.
      if(hp.lt.0.) si=-1.
      do 110 i=1,n
110   dq(i,nn0)=dqort(i)/anorm*si
      do 112 i=1,n
112   qn(i)=qnp1(i)+dqort(i)*hp
      if(ipr.gt.3) go to 111
      write(6,*) 'hp =',hp
      write(6,*) 'qn(i) =',(qn(i),i=1,n)
      write(6,*) 'nfi=',nfi,'      amu=',amu
111   continue
      call funct(qn,ksin,pksi,fin,fpar,amu,nfi,nksi,n,np)
      do 114 i=1,nksi
114   dksi(i,nn0)=(ksin(i)-ksinp1(i))/anorm/abs(hp)
      if(nfi.eq.0) return
      do 124 i=1,nfi
124   dfi(i,nn0)=(fin(i)-finp1(i))/anorm/abs(hp)
      return
      end
```

```
      subroutine subpq(n,nksi,ipr,nnp)
      parameter (npar=20,nmeas=1001,ncon=20)
      real ksinp1
      dimension w4(npar,npar),lw1(npar),lw2(npar)
      common/cid/epsi(npar),izlw2(npar),w(npar),finp1(ncon)
     .,fi0(ncon),dfi(ncon,npar),dfi0(ncon,npar)
     .,ksinp1(nmeas),pksi(nmeas),dksi(nmeas,npar)
     .,q0(npar),q00(npar),qnp1(npar),pq(npar,npar)
     .,dq(npar,npar),dqh(npar)
      call array(2,n,n,npar,npar,w4,dq)
      call minv(w4,n,detsub,lw1,lw2)
      call array(1,n,n,npar,npar,w4,dq)
      if(ipr.le.4) print*,'detsub',detsub
      do 10 i=1,n
      do 10 j=1,n
      aa=0.
      do 20 k=1,nksi
   20 aa=aa+dksi(k,j)*dksi(k,i)*pksi(k)
   10 w4(i,j)=aa
      call matr(dq,npar,w4,npar,dksi,nmeas,n,n,n,2)
      call matr(dksi,nmeas,dq,npar,w4,npar,n,n,n,1)
      do 30 i=1,n
      do 30 j=1,n
   30 pq(i,j)=pq(i,j)+w4(i,j)
      return
      end
```

REFERENCES

Benjamin, B. and J.H. Pollard (1980) The Analysis of Mortality and Other Actuarial Statistics. London: Heinemann.

Bogue, D. (1968) Principles of Demography. New York: John Wiley and Sons.

Brass, W. (1971) On the scale of mortality. Pages 86-110 in Biological Aspects of Demography, edited by W. Brass. London: Taylor and Francis.

Brooks, C., D. Sams and P. Williams (1980) A Time Series of Smooth Approximations for Age, Sex and Marital Status Specific Death Rates in Australia, 1950/51 to 1975/76 with Projections to the Year 2000. Research Memorandum. Melbourne, Austria: Impact Project Research Centre.

Brown, H.P. and A.R. Hall (1978) Australian Demographic Data Bank, Volume I: Recorded Vital Statistical 1921-1976. Canberra: Research School of Social Sciences, Australian National University.

Coale, A.J. and P. Demeny (1966) Regional Model Life Tables and Stable Populations. Princeton, New Jersey: Princeton University Press.

Coale, A.J. and D.R. McNeil (1972) The distribution by age of the frequency of first marriage in a female cohort. Journal of the American Statistical Association 67:743-749.

Coale, A.J. and J. Trussell (1974) Model fertility schedules: variations in the age structure of childbearing in human populations. Population Index 40(2):185-206.

Heligman, L. and J.H. Pollard (1979) The Age Pattern of Mortality. Research Paper No. 185. Sydney: School of Economic and Financial Studies, Macquarie University. Subsequently published in Journal of the Institute of Actuaries 107:49-80.

Henry, L. (1961) Some data on natural fertility. Eugenics Quarterly 8(2):81-91.

Hoem, J.M., D. Madsen, J.L. Nelsen, E.M. Ohlsen, H.O. Hansen and B. Rennermahn (1981) Experiments in modelling recent Danish fertility curves. Demography 18(2):231-244.

Powell, A.A. (1977) The Impact Project: An Overview. First progress report of the Impact Project, volume 1. Canberra: Australian Government Printing Service.

Rogers, A. and L.J. Castro (1981) Model Migration Schedules. RR-81-30. Laxenburg, Austria: International Institute for Applied Systems Analysis.

Rogers, A., R. Raquillet and L.J. Castro (1978) Model migration schedules and their applications. Environment and Planning A 10(5):475-502.

Sams, D. (1981) The Double Exponential Function. Research Memorandum. Melbourne, Australia: Impact Project Research Centre.

United Nations (1967) Methods of Estimating Basic Demographic Measures from Incomplete Data. New York: United Nations.

Williams, P. (1981) Marriage and Divorce in Australia: A Time Series of Fitted Distributions, 1921/22 to 1975/76. Working Paper B-20. Melbourne, Australia: Impact Project Research Centre.