WORKING PAPER

POPULATION MODELS ANALYSIS PROGRAM
(POPMAN)

Anna Lewandowska

October 1986
WP-86-053

IIASA
International Institute
for Applied Systems Analysis

POPULATION MODELS ANALYSIS PROGRAM (POPMAN)

Anna Lewandowska

October 1986
WP-86-53

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

FOREWORD

The Population Program at IIASA deals with various aspects of population aging phenomena in developed countries. The demographic future of populations can be estimated using equations for population dynamics. The variety of assumptions about the demographic characteristics which are appropriate for any particular population can be realized in various forms of boundary conditions for partial differential equations. This paper by Anna Lewandowska describes a convenient interactive procedure which can be used on an IBM PC (or compatible) for calculating such demographic equations. The procedure is combined with a Lexis program developed in IIASA's Population Program that allows to see the results of calculations in the form of shaded contour maps on the color monitor of a PC. This tool is useful for analyzing the dynamic properties of the age-specific population structure.

Anatoli Yashin
Deputy Leader
Population Program

POPULATION MODELS ANALYSIS PROGRAM (POPMAN)

Anna Lewandowska

1. Introduction.

The POPMAN (acronym for **POP**ulation **M**odel **AN**alysis **P**rogram) was developed especially for fast analysis of the mathematical models of population dynamics, formulated in terms of partial differential equations of hyperbolic type. This program makes possible:

- analysis of the models formulated in terms of function of 2 variables with given structure and parameters,

$$y=f(x,t,a_1,\ldots,a_p)$$

- analysis of the model formulated as a hyperbolic partial differential equation

$$@u/@t = - @u/@x - f(x,t,a_1,\ldots,a_p)u(x,t)$$

with given initial and boundary conditions.

The program produces the output file which can be used by LEXIS program (Gambill and Vaupel, 1985) as the input one; this makes possible easy visualization of output data generated by POPMAN in terms of color contour maps.

The following assumptions were taken into account when designing the program structure:

1. The program should be very highly interactive, user-friendly and should be simple enough to be operated by a user not being a computer specialist.

2. All the formulas and data constituting the problem definition should be entered in a very simple form, using standard mathematical notation.

3. Results should be presented in graphical form.

4. The program should not require to return to the operating system for any purpose during its utilization.

The software presented in this paper is the result of cooperation with SDS program at IIASA. Some rather essential parts of it, like EDITOR, WINDOW LIBRARY, PL0 COMPILER are modified versions of the software developed within the MDA project for DIDAS system (Lewandowski, 1986). Without availability of these tools, development of the POPMAN program would require much more time and resources.

2. General structure of the POPMAN program.

All the assumptions specified above have led to the following structure of the program:

1. All formulas describing the problem are defined using a very small subset of Pascal language. This subset is small enough to be efficiently managed by non experienced user and broad enough to make defining of complicated problem possible. Therefore the COMPILER and the INTERPRETER of this subset constitute the most important parts of the system.

The problem description written in the subset of Pascal is translated to some internal form (known as P-code) which is similar to the sequence of commands of simple calculator, utilizing the reverse Polish notation (like, for example, the Hewlett-Packard calculator). This sequence of commands is interpreted by the INTERPRETER routine every time, when it is necessary to calculate the values of right hand side of the equation, boundary conditions, or initial conditions. The INTERPRETER is invoked by the differential equation solver routine, which transfers to the INTERPRETER the current values of state variables x and time t and expects from the INTERPRETER the information about right hand sides of the equation being solved.

2. The INTERACTIVE EDITOR makes possible easy defining the problem, i.e. entering the problem description formulated in the mentioned above subset of Pascal. Utilization of this EDITOR is rather simple - all the actions are menu-driven and are self explanatory.

3. The numerical solution of hyperbolic partial differential equation is performed by the SOLVER MODULE. This module utilizes the fourth-order Runge-Kutta integration method with automatic step-size adjustment, for solving the set of ordinary partial differential equations. This set of ordinary differential equations is the result of applying the method of lines for discretization the hyperbolic partial differential equation.

Sometimes it is enough to analyze simpler model formulated in terms of algebraic equations. In this case it is not necessary to invoke the SOLVER; simple tabulation of the function must be performed. The CALCULATE FUNCTION TABLE module tabulates the function (1) for given values of independent variables x and t. Similarly to previous case, this module utilizes compiled form of the function and directly invokes the INTERPRETER.

4. The LEXIS interface transfers all necessary data from POPMAN program to LEXIS program. The user should consult the LEXIS manual for details of operation. The POPMAN program generates standard LEXIS input file, which can be analyzed by LEXIS to produce a contour map. See Appendix for details related to organization of this file.

5. All parts of the POPMAN program are supervised by the USER INTERFACE MODULE, which invokes all parts of the system on user request, performs all necessary coordination of system components, ensures proper data transfer and supervises correctness of the operation. Similarly to the EDITOR, this part of the system is also menu-driven and all the actions are self explanatory. Especially, particular attention was oriented to ensure high level of error detection - for example it is not possible to invoke the compiler without a problem description having previously entered.

The overall structure of the system and the information flow between its components is presented on Fig.1.

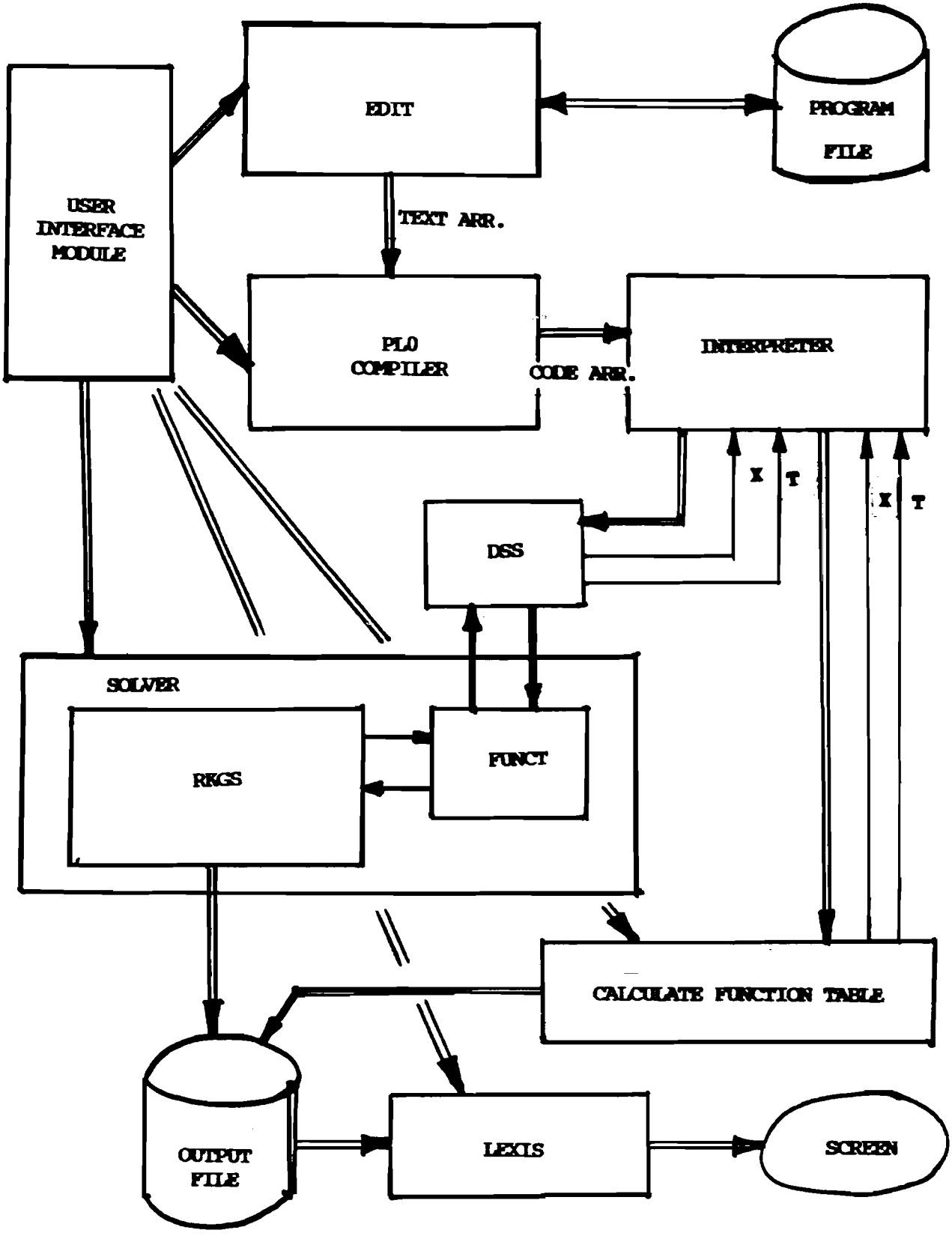

Fig. 1 General structure of POPMAN program.

3. Implementation of the POPMAN program

3.1. Numerical method of lines for solving hyperbolic partial differential equations.

There exist several methods for numerical solution of partial differential equations; rather detail and extensive discussion of these problems can be found, for example, in the book by Vemuri and Karplus (1981). From the practical experience follows, that the method which can be recommended for numerical simulation of systems described by partial differential equations, is the method known as **method of lines.**

The principle idea of this method is to replace the spatial derivatives @u/@x (and possibly derivatives of higher order) with finite difference approximations. This procedure leads to a system of ordinary differential equations which can be solved with respect to unknown function u(t) for given set of discretization points. The derivative @u/@x is approximated at a series of grid points $x_1, x_2, \ldots, x_{N-1}$ with

$$h = x_2 - x_1 = x_3 - x_2 = \ldots = x_{N-1} - x_{N-2}$$

where h is prescribed discretization step of the space interval [0,L]. The resulting equations must be integrated simultaneously since they are interconnected through the algebraic approximation for spatial derivatives. Thus the dependent variable vector $u_1(t), u_2(t), \ldots, u_N(t)$ can be computed by integration the derivative vector du_1/dt, du_2/dt, ..., du_N/dt. These N functions can be considered to move along lines parallel to line x=0; this is the reason, why this method is generally termed as the **numerical method of lines.**

Using standard grid methods for solving partial differential equations it is usually necessary to develop new software and new approach for any particular problem. It is rather difficult to apply these methods directly for development general and universal software packages, which could be used for highly automatic solution of any partial differential equation.

The method of lines is a remarkably flexible and comprehensive algorithm for partial differential equations, which is currently more popular and more broadly used than the classical grid methods. The number of very efficient software packages and simulation languages which utilize method of lines are currently available (for detail discussion see Vemuri and Karplus, 1981). The advantages of this method can be especially observed, when system under study is described by a set of complicated, usually nonlinear partial differential equations, frequently with parts described by ordinary differential equation.

One of the most essential problems arising when method of lines is applied, relates to the form of finite difference operator approximating the spatial derivatives. It was discovered, that it is especially important in the case of hyperbolic equations. Improper selection of this approximation can lead to serious distortion of the numerical solution, especially when the initial or boundary conditions contain sharp peaks or jumps. This problem was discussed in details, among others, by Chen (1980) and Carver (1980).

The most widely known methods used for approximation the spatial derivative are the following:

- first-order backward difference,
- central difference,
- upwind difference,
- biased upwind difference.

It was shown (Sincovec, 1975, Carver, 1980, Vemuri, 1981) that using the central differences approximation or upwind differences approximation for hyperbolic partial differential equations is not very effective. Better results were obtained by using combination of central and total upwind approximations, which is named biased upwind approximation. Numerical experiments have shown, that in most cases the best results can be obtained with four-points biased upwind approximation, in which the derivative @u/@x was replaced with the following finite difference formulas:

$$@u_1/@x = (-11\,u_1 + 18\,u_2 - 9\,u_3 + 2\,u_4)/(6h)$$

$$@u_2/@x = (-2\,u_1 - 3\,u_2 + 6\,u_3 - u_4)/(6h)$$

...

$$@u_i/@x = (u_{i-2} - 6\,u_{i-1} + 3\,u_i + 2\,u_{i+1})/(6h)$$

...

$$@u_N/@x = (-2\,u_{N-3} + 9\,u_{N-2} - 18\,u_{N-1} + 11\,u_N)/(6h)$$

The above numerical scheme was used in the POPMAN program. This scheme is implemented by the separate procedure (named **DSS**), which converts the state vector $\{u_k\}$ into the vector of derivatives $\{@u_k/@x\}$. This procedure can be easily replaced by other one, implementing different discretization scheme.

3.2. The SOLVER module for integration the ordinary differential equations.

The existing experience with solving ordinary differential equations indicates, that fourth-order **Runge-Kutta** method with automatic control of integration step is the most effective procedure. Therefore, this method was applied in the POPMAN program.

The control mechanism responsible for selection the integration step applied in the implemented routine is one of the simplest known. Starting with given integration step, integration over the interval $[t_o,t_1]$ is performed twice - with step h and step h/2. Results of these calculations are compared. If the difference between obtained solutions at the end of integration interval is less than the specified tolerance (this tolerance is denoted in the program as **eps**), this process is interrupted and the obtained solution is considered as satisfactory. If this difference is greater than **eps**, integration step is divided by 2 and the whole process is repeated. This process is more time consuming than other methods for adapting integration step, based on the estimation of truncation error, but is much more robust. This property is rather important and decided about applying the mentioned above method in POPMAN.

However, due to the properties of method applied for adjustment the integration step, the following should be observed by the user:

1. Computation time can be different for various problems. In specific cases, computation time can exceed few times the average one.

2. Too large integration step or too high accuracy required (parameter **eps**) can result in rather essential increasing of computing time.

3. Too low value of **eps** can result in serious problems during integration equations - in such cases the program will be interrupted and the error message displayed to the user.

It follows from above, that it is reasonable to experiment with the program to find the best integration step for a given particular problem. This can result in rather essential saving of computing time during experimenting with the model.

The particular implementation of Runge-Kutta method used in POPMAN program is direct translation to Pascal the RKGS procedure from FORTRAN library SSP. More detailed information about this procedure can be found in SSP manual. The procedure communicates with the rest of the program through FUNCT procedure, which defines the right-hand side of equation being solved and OUTP procedure, which provides some control over the integration process.

4. Using the POPMAN program

4.1 Defining the model.

The model and all necessary information can be defined by the user using the **Micro-Pascal**, a very small subset of Pascal language. The Micro-Pascal is simple enough to be efficiently used by non specialist, but simultaneously powerful enough to define quite complex models.

The Micro Pascal contains the following keywords:

begin
end
if ... then
while ... do
var
const
procedure
call

Some variables are predefined as standard ones:

- the parameters defining the model under study:

t current value of variable t ("time")
x current value of variable x ("space variable")
fct current value of the function f(x,t,...) defining the model
bcn current value of boundary condition for given t

icn current value of initial condition for given x

- the parameters defining the integration process:

tinit starting value of variable t
tmax end value of variable t
step integration step
xlow starting value of variable x
xupp end value of variable x
dim number of discretization points of the interval [xupp, xlow]

The above variables cannot be defined by the user using the VAR declaration. Their values are predefined and can be changed by the user. The default values are following:

tinit = 0,
tmax = 100,
step = 0.05,
xlow = 0,
xupp = 100,
dim = 10.

The variable **status** has a special meaning. Its value informs the program about the current status of computing process. The value of this variable can be equal to one of the 4 predefined constants:

parameter program is in **parameter definition phase**
initial program computes the value of **icn - initial condition**
boundary program computes the value of **bcn - boundary condition**
function program computes the value of **function.**

This variable can be used to avoid unnecessary computation, what results in shorter execution time. To illustrate its usage, let us consider the following example:

Simple and not efficient program:

```
begin
   fct:=t*exp(x);
   icn:=sin(x);
   bcn:=cos(x);
end.
```

The same program in more efficient form:

```
begin
   if status=function then fct:=t*exp(x);
   if status=initial then icn:=sin(x);
   if status=boundary then bcn:=cos(x);
end.
```

The user can define his own variables using **var** declaration as in the following example:

```
var a,b,c;
```

There is no need to declare the type of defined variables; all variables are considered as **real**.

The user can define also real constants, for example:

```
const z=15;
```

The Micro Pascal program can contained **procedures** without parameters and call them using the **call** instruction. The procedures can be nested, and may contain local variables.

The same standard functions as in Pascal are available for the programmer:

sin(x)
cos(x)
exp(x)
ln(x)
log(x)
sqrt(x)
abs(x).

4.2. Starting the POPMAN program

To start the program it is necessary to enter diskette to one of the drives and enter the command **pop** (from the DOS level). As the result of this action, the title of the program is displayed on the screen (Fig.2). After pressing any key the user will see the Main Menu displayed on the screen (Fig.3).

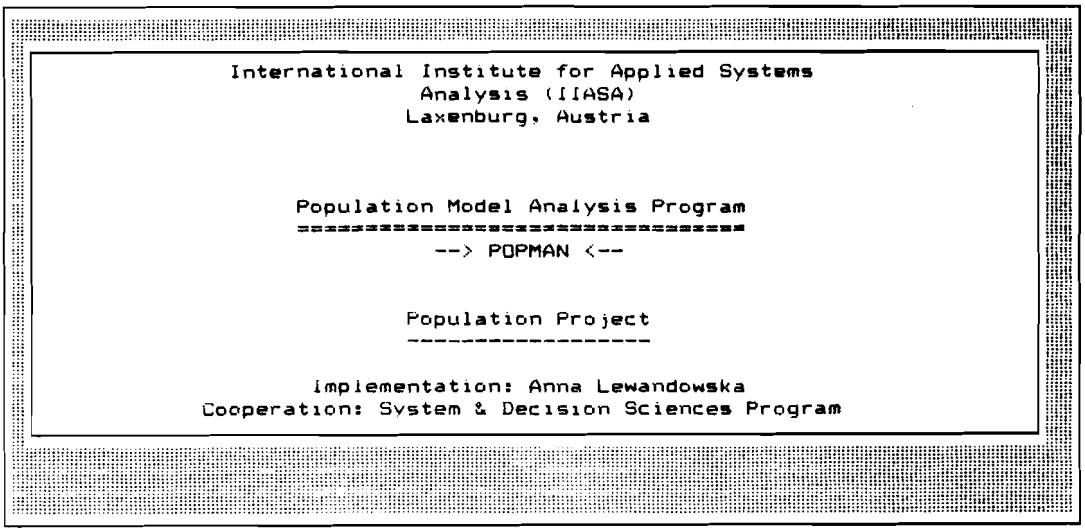

Fig. 2 The title screen of the POPMAN program

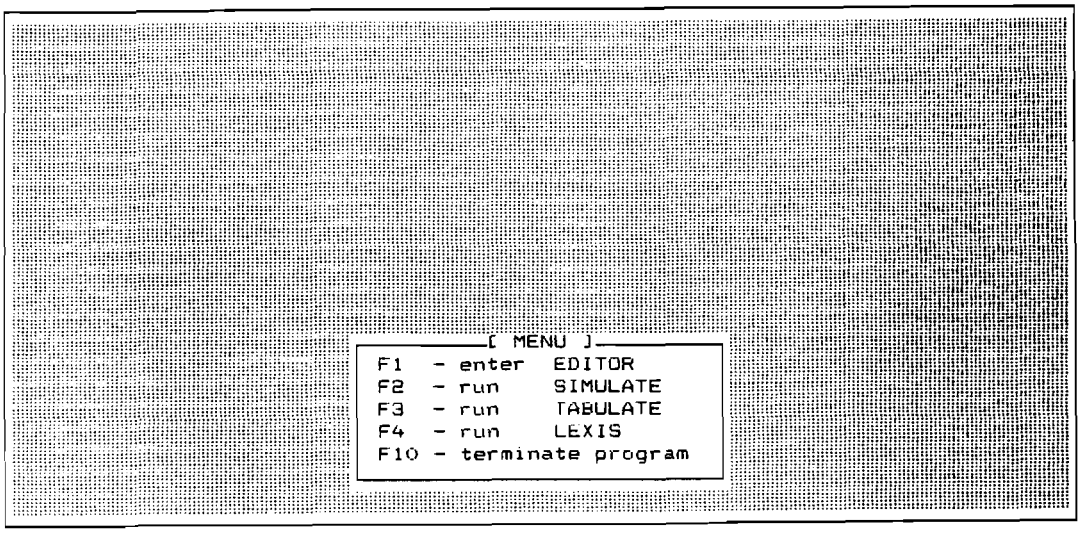

Fig. 3 The Main Menu of the POPMAN program

In the **Main Menu** the following options can be selected by the user in order to perform certain action

- entering EDITOR to define the problem,
- running SIMULATE, to solve the differential equation,
- running TABULATE, to calculate function values,
- running LEXIS program to display results in the form of map,
- termination the program and return to DOS.

An attempt to run SIMULATE or TABULATE without previously defined program causes display of warning and entering EDIT.

4.3. Editing the program

The interactive editor **EDIT** being one of the modules of POPMAN program makes possible interactive entering and updating the model definition. The EDIT module can be in one of the following states:

- Program Entering (Editing) when new program can be entered or existing one edited,
- Line Entering State, when single lines can be inserted in program text,
- Line Appending State, when new lines can be appended to the program,
- Line Editing State, when single line can be modified.

Pressing the F1 Function Key from Main Menu level the program enters the Editor Menu (Fig.4). Pressing the F2 Function Key program text can be loaded from disk. The user should respond with file name (Fig.5). If the file specified by the user does not exist, the editor switches automatically to the Program Entering State. When loading process is completed,

the editor enters the Program Editing Mode and the program text is displayed on the screen (Fig.6).

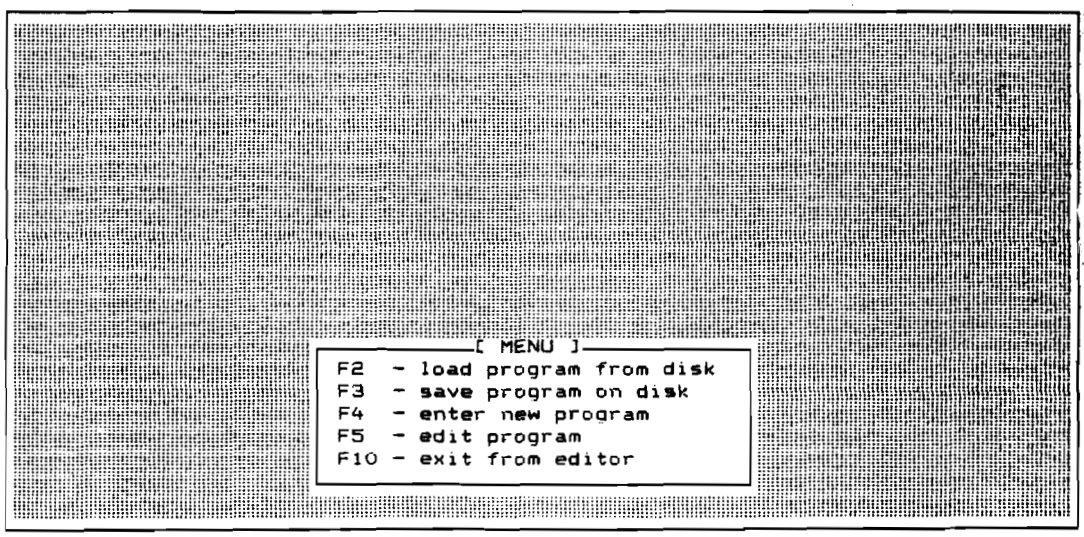

Fig. 4 The Editor Menu

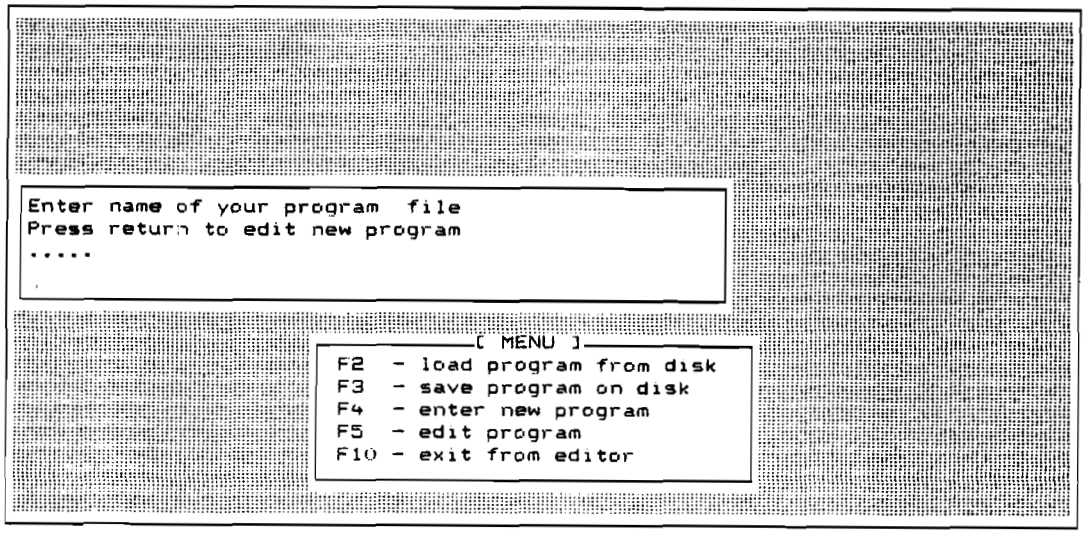

Fig. 5 Loading the program file

The user can select the active line by moving the line cursor. This can be achieved with the help of cursor keys (arrow-up, arrow-down). The active line can be edited or deleted. The Line Editing mode can be entered by pressing the F7 Function Key. As the response to this action, the line

editing window appears on the screen (Fig. 7). This window contains the line which should be edited. The small cursor can be moved with the help of cursor keys (arrow-right, arrow-left). Pressing other alphanumeric key will cause insertion of the corresponding character under cursor; afterwards the cursor moves one position to the right. If the cursor is located on the last character of the edited line, new characters will be appended to the line. Single characters can be deleted using the Del key. When editing is completed, it is possible to exit this mode by pressing the F10 Function Key.

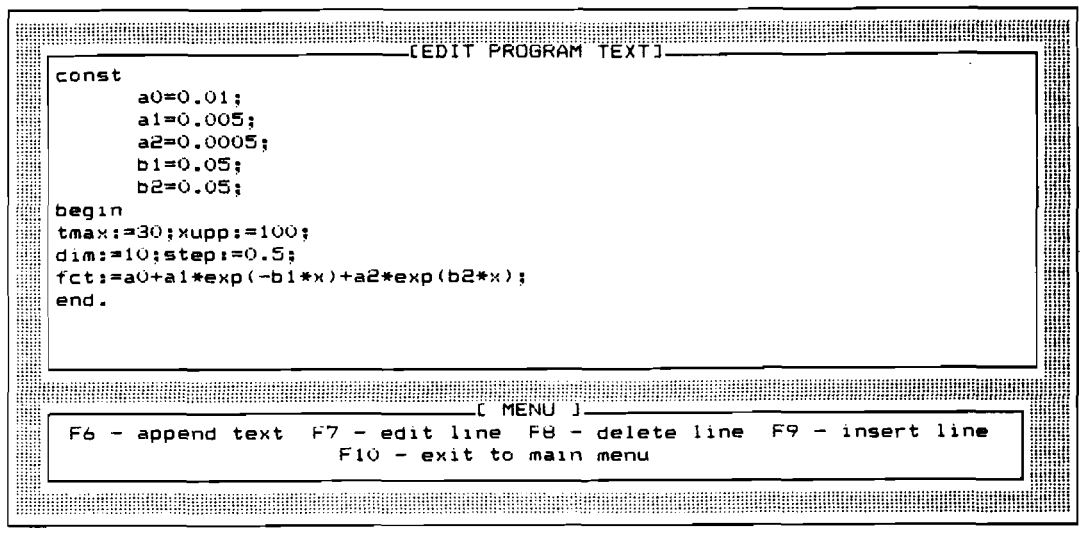

Fig. 6 Entering and editing the program text

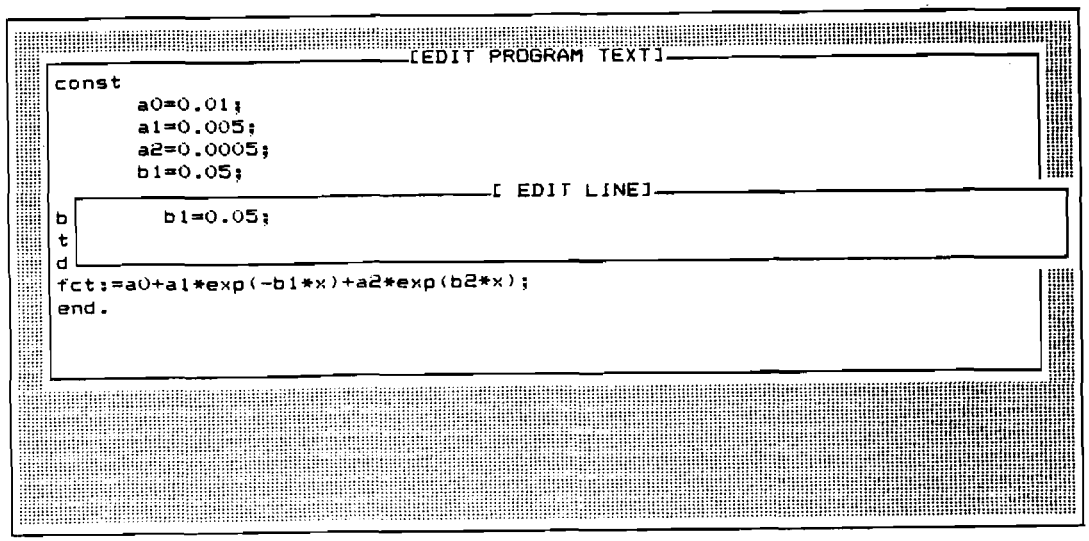

Fig. 7 Editing the single program line

New line can be inserted by pressing the F9 Function key. New line is inserted **before** the active line (marked by the line cursor). When the line is completed, the RETURN key terminates this process. The line cursor is located on the new line. Applying this procedure it is possible to enter only one line. In order to enter the next one, this procedure must be repeated.

New lines can be appended to the program text by pressing the F6 Function Key. The line cursor is located on the first line following the last line of the program. Any number of lines can be entered in this mode. Every line must be finished by pressing the RETURN key. In order to interrupt this process, empty line must be entered.

From the Editor Menu level the user can enter directly the Program Entering State by pressing the F4 Function Key. The empty windows appears on the screen. The user can enter the program text in the same way, like in Appending Mode. Empty line terminates this process. It is necessary to remember, that before leaving the POP program (either by returning to DOS or by invoking LEXIS) it is necessary to save the program on disk. Otherwise the program will be lost.

Utilization of the EDIT program is rather straightforward and does not need more detail explanation. Short training is necessary to investigate all possible options.

4.4. Running the program.

The Micro Pascal program can be executed from the Main Menu level (Fig. 3). This can be achieved by invoking the SIMULATE or TABULATE options. The first one initiates the simulation program (solving the hyperbolic PDE), the second one - tabulation of the function. As the first action, both options cause compilation of the previously loaded or entered program. This program is translated into some internal form which is used for further computations.

Frequently an error occurs in the source program. This situation is detected and proper information displayed to the user (see Fig. 8). Error message together with content of this line where error occurred is displayed in the Error Window. It is necessary to point out, that in several situations error occurs **before** the line being displayed with error message. Pressing any key it is possible to return to the editor. The line where error occurred (or sometimes next line) will be highlited.

If the Micro Pascal program is not complete (too less **end** or too many **begin**) the POP program loops with information "Program Incomplete". It is necessary to use the Ctrl-C to interrupt. This problem occurs due to limitations of Turbo-Pascal compiler. It is also necessary to point out that the Micro Pascal program must be finished by **end and period (end.)**.

During the run of SIMULATE, the plot of consecutive solutions (as the function of variable x) is displayed, as well as all current parameters of the problem (Fig.8). This gives a possibility to observe a progress of computations. In the case of bad behavior of this process (badly selected integration step or parameter definition) it is possible to interrupt, hitting any function key or Esc key. It should be noticed, that the effect of this operation is not immediate - the program checks the keyboard status every few seconds.

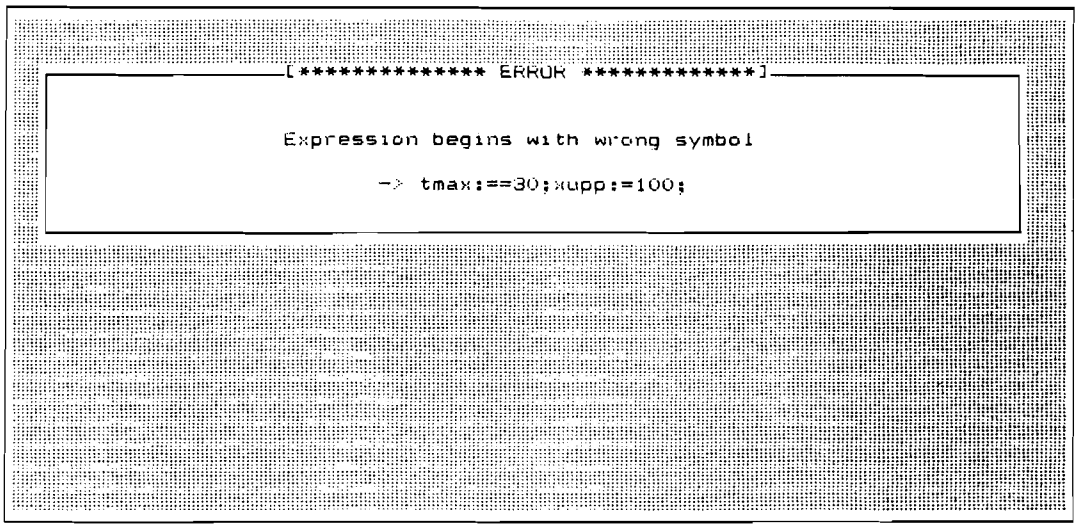

Fig. 8 Error message screen

The numerical calculations (execution of Micro Pascal program) is supervised by Turbo Pascal run time system. Therefore any numerical error (like logarithm or square root of negative value, too high or too low value of function argument, dividing by zero etc.) causes termination of the program and return to DOS with the information:

RUN-time error "number", PC="address"

(PC means Program Counter). In order to find the statement of the program where the error occurred, the following actions must be undertaken:

- the Turbo Pascal compiler must be invoked from the DOS level,
- options should be invoked from Turbo Pascal menu level (by pressing the key "o"),
- the "f" suboption must be invoked; as the response Turbo Pascal will ask about the file name. The user should respond with "pop". After that program will ask about PC address. The value displayed together with error message should be entered.

During the SIMULATE or TABULATE phases all the results are transferred to the output file named **LEX**, which structure is the same, like standard LEXIS input file. The data label begins from 1001; all consecutive labels are incremented with step 1. The number of X data is equal to

(tmax-tmin)/step.

Number of Y data entries is equal to **dim** and they are labeled starting from 1, with increment 1.

It should be noticed, that in a case of abnormal termination of the execution of SIMULATE module, the LEX file has wrong structure and is not complete. Therefore the LEXIS program should be not invoked in such cases.

The LEXIS program can be invoked from the Main Menu level. The user should consult the LEXIS manual for details relating to running this program. After completion of LEXIS it is possible to return to POP program, according to the information displayed on the screen.

4.5. Examples.

In this section of the paper we will present some sample programs which were used for testing the software. They are presented here in order to demonstrate how to solve some particular problems, how to code the problem in Micro Pascal and what solutions can be obtained. We will not discuss here the detailed meaning of these examples since this paper is oriented to description of the software, rather than to detailed analysis of the particular population models.

Example I. There is given the 5 parameter mortality model

$$\mu(0,x)=a_0 + a_1 \exp(-b_1 x) + a_2 \exp(b_2 x),$$

as the function of one variable x (age).

The POPMAN program was used to tabulate the function $\mu(0,x)$, for x from the interval [0, 100]. The text of Micro Pascal program is shown on Fig.6. On Fig.9 there is presented the sample screen produced by invoking the TABULATE module and containing the plot of this function.

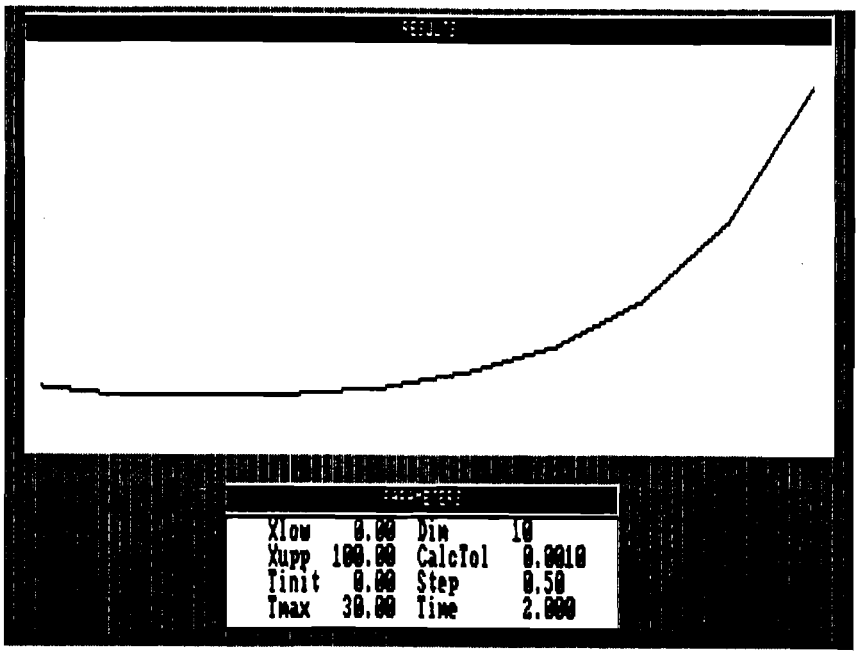

Fig. 9 The sample output screen

Example II. The other version (time dependent) of the mortality model is given

$$\mu(t,x) = \mu(0,x)\ f_1(t,x),$$

where

$$f_1 = 1 - \alpha t / \sqrt{(x + 1)},$$

for t from the interval [0, 100] and x from the interval [0, 100].The function u(0,x) is the same as in the Example I; x denotes here the age and t - the time. The program in Micro Pascal calculating values of the above function is as follows

```
const
      alfa=0.01;
      a0  =0.01;
      a1  =0.005;
      a2  =0.0005;
      b1  =0.05;
      b2  =0.05;
var   mi0, f1;
begin
      step:=0.5; tmax:=30;
      mi0 :=a0+a1*exp(-b1*x)+a2*exp(b2*x);
      f1  :=1-alfa*t/sqrt(x+1);
      fct :=mi0*f1;
end.
```

The values of the parameters **dim** and **xupp** are not explicitly defined in the program, therefore default values are taken for calculations (10 and 0 respectively). The resulting contour plots obtained from LEXIS program is presented on Fig.10.

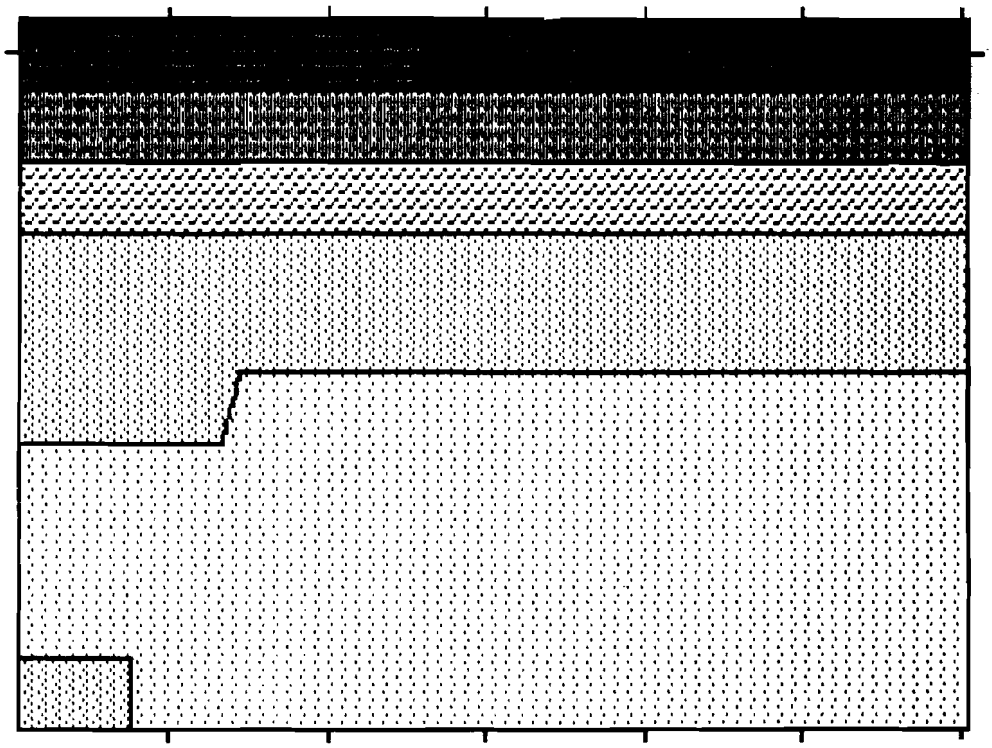

Fig. 10 The contour map produced by LEXIS program for Example IIa

If the function $f_1(t,x)$ from the Example II is replaced by the following one

$$f_2(t,x)=\exp(-b_2 t),$$

then the resulting surface plot will be as that one presented on Fig.11.

Fig. 11 The contour map produced by LEXIS program for Example IIb

Example III. There is given the hyperbolic partial differential equation describing the population density for given age x and time t:

$$@u/@t = - @u/@x - \mu(x,t)\ u(x,t)$$

with boundary condition

$$u(0,t) = \exp(b_2 t)$$

and with initial condition

$$u(x,0) = \exp\left(-\int_0^x \mu(0,z)dz\right)$$

The above equation must be solved for two cases:

$$(a) \quad \mu(t,x) = \mu(0,x)\ f_1(t,x)$$

$$(b) \quad \mu(t,x) = \mu(0,x)\ f_2(t,x)$$

where the function $\mu(0,x)$ is the same like in Example I, the functions $f_1(t,x)$ and $f_2(t,x)$ are the same, like defined in Example II.

The Micro Pascal program solving the above defined equation for the case (a) is as follows

```
const
      alfa=0.01;
      a0=0.01;
      a1=0.005;
      a2=0.0005;
      b1=0.05;
      b2=0.05;
var   mi0,f,e1;
begin
      if status=parameter then
            begin
              step:=0.5; dim:=10;
              tmax:=30; xupp:=100;
            end;
      if status=boundary then
         bcn:=exp(b2*t);
      if status=initial then
            begin
              e1:=a0*x+a1/b1*(1-exp(-b1*x))+a2/b2*(exp(b2*x)-1);
              icn:=exp(-e1);
            end;
      if status=function then
            begin
              mi0:=a0+a1*exp(-b1*x)+a2*exp(b2*x);
              f:=1-alfa*t/sqrt(x+1);
              fct:=mi0*f;
            end;
end.
```

Please note the program lines printed in boldface, responsible for assignment the values to the predefined variables **bcn**, **icn** and **fct** responsible for boundary values, initial values and function appearing in the right hand side of the equation. The usage of the **status** variable is also presented in the above program.

The program solving the second variant of the problem differs from the above only in one line (the 4.th from the end), defining the function f(x,t). This line should be replaced by the following one

```
f:=exp(-b2*t);
```

The results of simulation obtained from LEXIS program as the contour maps are presented for the case (a) on Fig.12 and for the case (b) on Fig.13.

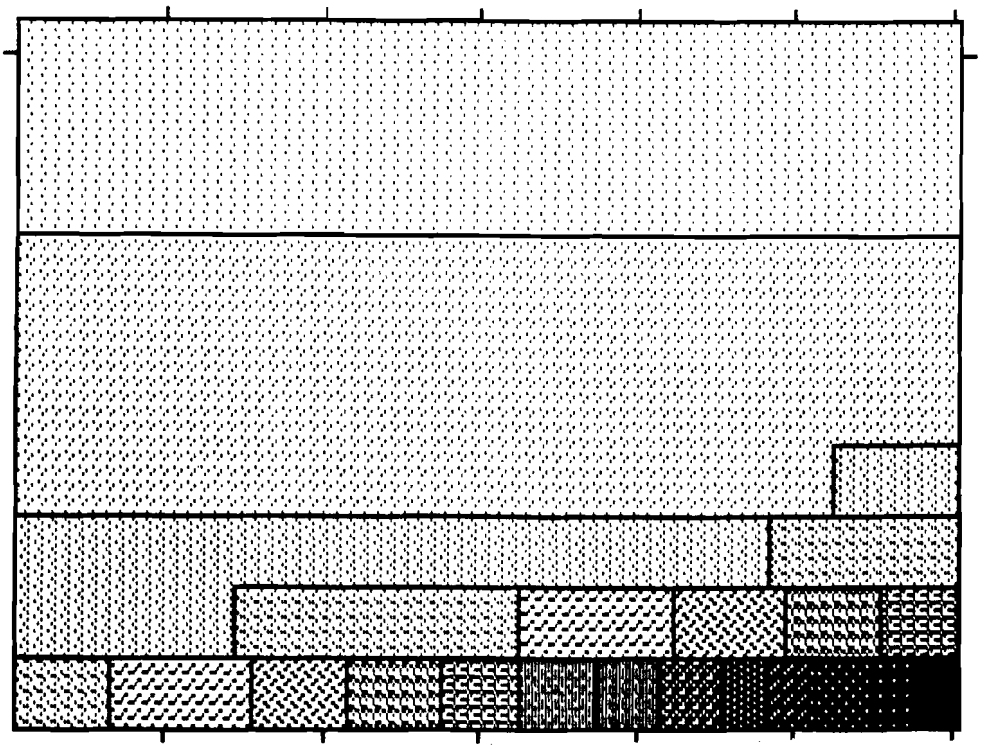

Fig. 12 The contour map produced by LEXIS program for Example IIIa

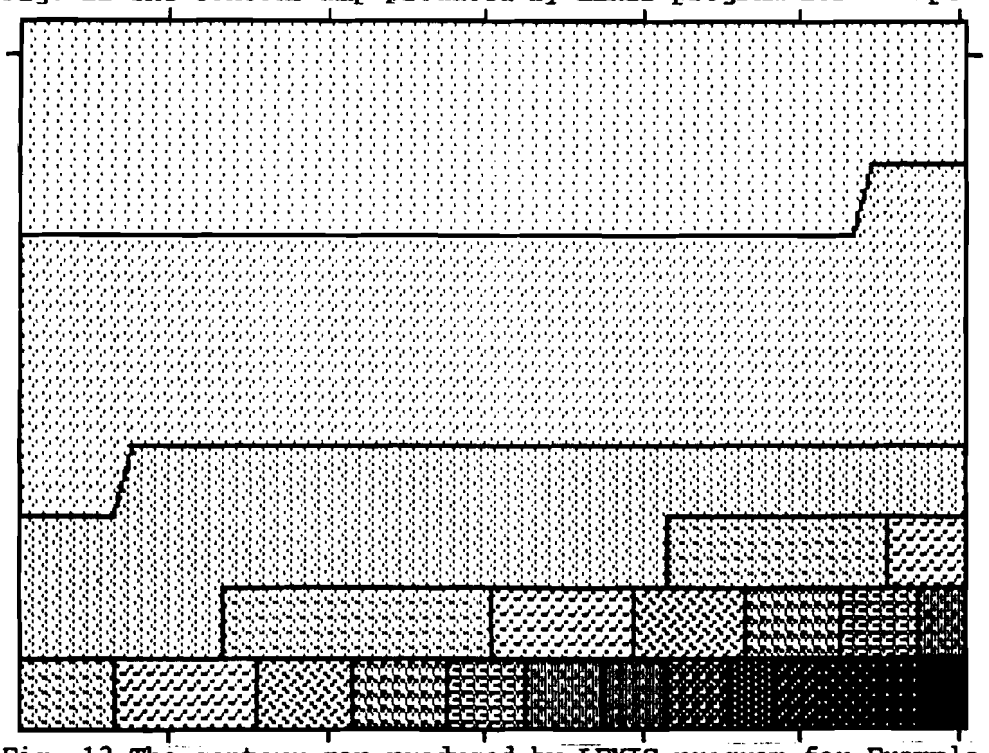

Fig. 13 The contour map produced by LEXIS program for Example IIIb

5. Possible extension

The program described in this paper should be treated as a very first step in development software tools for simulation analysis of population models. The following are the possible modifications and extensions to the first version of the software:

- The POP program should be extended in such a way, that run time errors could be monitored by its own error routines. In such cases numerical errors will not cause return to DOS. The generation of LEX file should be modified, as well as interface to other programs (like STATGRAPHIC) should be build in. Definition of the problem in the form of tables or information extracted from data bases should be made possible. More advanced graphic presentation of the results (3-D plots) could improve essentially analysis of the results.

- Class of the problems solved should be broadened. It is possible to extend this approach for solving vector PDE, for solving mixed problems described by interconnected partial and ordinary differential equations or differential equations of parabolic type.

- Sensitivity analysis module could be added to the program. This would be possible due to the availability of Micro Pascal compiler which can provide analytical differentiation of the mathematical expressions. This would be a first step for implementing parameter estimation and model identification procedures.

- Application of population modeling software for population policy analysis, through integration the POP program with the DIDAS system developed at SDS, could be possible.

6. References.

Arthur W.B. and Vaupel J.W. (1983). Some General Relationships in Population Dynamics. WP-83-89, Working Paper, International Institute for Applied System Analysis, Laxenburg, Austria.

Carver M.B. and Schiesser W.E. (1980). Biased Upwind DIfference Approximations for first-order Hyperbolic Partial Differential Equations. DSS/2 Manual. Lehigh University.

Chen K.L. and Schiesser W.E. (1980). Upwind Approximations in the Numerical Method of Lines Integration of Hyperbolic Partial Differential Equations. DSS/2 Manual. Lehigh University.

Gambill B.A. and Vaupel J.W. (1985). The LEXIS Program for Creating Shaded Contour Maps of Demographic Surfaces. WP-85-94, Working Paper, International Institute for Applied System Analysis, Laxenburg, Austria.

Lewandowski A. (1986). Problem Interface for Nonlinear DIDAS. International Institute for Applied System Analysis, Laxenburg, Austria, to appear.

Sincovec R.F. and Madsen N.K. (1975). Software for Nonlinear Partial Differential Equations. ACM Transactions on Mathematical Software. Vol.1. No.3.

SSP Manual. System/360 Scientific Subroutine Package, Version III, Programmers's Manual.

Vemuri V. and Karplus W.J. (1981). Digital Computer Treatment of Partial Differential Equations. Prentice-Hall.

Wirth N. (1976). Algorithms + Data Structures = Programs. Prentice-Hall.