```
3450 NEXT Z
3460 T=T1+H
3470 Z=4
3480 FOR J=1 TO N0
3490 A(J)=B(J)+K(J,3)
3500 NEXT J
3510 GOSUB 4630 ! K(J,4)
3520 FOR J=1 TO N0
3530 A(J),B(J)=B(J)+(K(J,1)+2*K(
     J,2)+2*K(J,3)+K(J,4))/6 ! A
     (J)n+1
3540 NEXT J
3550 RETURN
3560 ! ----------STO PLOT
3575 PRINT# 1 ; T
3580 FOR J=1 TO P9
3590 A3=A(M(J,1)) ! VAR NO.
3600 ! CHECK GRAPH SCALES
3610 IF A3<M(J,2) THEN M(J,2)=A3
3620 IF A3>M(J,3) THEN M(J,3)=A3
3630 PRINT# 1 ; A3
3650 NEXT J
3660 RETURN
3670 ! ----------PRT OUTPUT
3690 DISP "T=";T
3700 FOR J=1 TO N2
3710 DISP "A(";A1(J" )=";A(A1(J
     ))
3720 NEXT J
3730 RETURN
3740 ! ----------
3760 FOR J=1 TO E9
3770 J1=E1(J,3)+E1(J,2)-1 ! 1ST
     VAL.
3780 J1=J1-1
3790 E2(J1+1)=E2(J1) ! MOVE QUEU
     E
3800 IF J1>E1(J,3) THEN 3780
3810 E2(J1)=A(E1(J,1))
3820 NEXT J
3830 RETURN
3840 ! ----------USE @
3850 ! Q=J,LAG TIME=L
3860 DEF FNQ(J)
3900 IF J<=E9 AND J>0 THEN 3920
3910 PRINT "QUEUE NO.";J;"DOES N
     OT EXIST-END" @ END
3920 IF L<=E3(J,2) THEN 3930
3925 PRINT "LAG TIME";L;"SPECIFI
     ED WITH QUEUE NO.";J;"IS TO
     O LONG" @ END
3930 IF L=0 THEN L=E3(J,2)
3940 R2=E1(J,3)+L/(E0$H)-(E+1)/E
     0 ! FRACTION NO.OF E2
3950 ! IF R1<=R2<=R3,INTERPOLATE
      BETWEEN R1&R3.IF R2<1ST RO
     W E2;EXTRAPOLATE FROM 1ST&2
     ND
3970 R1=MAX(INT(R2),E1(J,3))
3980 R3=MAX(INT(R2+1),E1(J,3)+1)
```

```
3990 V6=E2(R1) @ V7=E2(R3)
4000 FNQ=V6+(V7-V6)*(R2-R1)/(R3-
     R1)
4005 L=0
4010 FN END
4020 ! ----------USE TABLE
4040 DEF FND(T5) = ! TABLE T5
4042 IF T5<=D9 AND T5>0 THEN 405
     0
4045 DISP "TABLE NO.";T5;"DOES N
     OT EXIST-END" @ END
4050 X=T+D0 ! INTERP TIME
4060 ! 1ST&LAST START TIME
4070 R1=D2(D1(T5,2))
4080 R2=D2(D1(T5,2)+D1(T5,1)-1)
4090 ! MAX TIME INCR
4100 R3=D3(D1(T5,4)+D1(T5,3)-1,1
     )
4110 IF X<R1 OR X>R2+R3 THEN 433
     0 ! RETURN ZERO
4120 ! FIND TABLE START TIME S1<
     =X
4130 R1=D1(T5,2)-1 @ R2=D1(T5,2)
     +D1(T5,1)-1 ! 1ST&LST ROWS
     OF D2
4140 R1=R1+1
4150 IF R1<R2 THEN 4160
4155 IF D2(R1)>X THEN R1=R1-1
4157 GOTO 4170
4160 IF D2(R1)>X THEN R1=R1-1 EL
     SE 4140
4170 S1=D2(R1)
4180 IF X>S1+R3 THEN 4330 ! RETU
     RN ZERO
4190 ! FIND ROW OF D3 WHERE TIME
      JUST>X
4200 R1=D1(T5,4) @ R2=D1(T5,4)+D
     1(T5,3)-1 ! 1ST&LST ROWS OF
      D3
4210 R1=R1+1
4220 IF R1=R2 THEN 4240
4230 IF S1+D3(R1,1)<X THEN 4210
4240 ! INTERPOLATE TABLE VAL-TIM
     ES T6<=X<=T7
4260 T6=S1+D3(R1-1,1)
4270 T7=S1+D3(R1,1)
4280 ! TABLE VALS AT T6&T7
4290 V6=D3(R1-1,2)
4300 V7=D3(R1,2)
4310 FND=V6+(X-T6)*(V7-V6)/(T7-T
     6)
4320 GOTO 4340
4330 FND=0
4340 D0=0 @ FN END
4350 ! ----------PLOT OUTPUT
4370 ! C1-1=NO.PTS STO'D
4380 IF C1>2 THEN 4410
4390 PRINT "NO PLOT PTS STORED-C
     HECK PLOT  RANGE(T3,T4)"
4400 GOTO 4620 ! RTRN
```

```
4410 ASSIGN# 1 TO *  ! EMPTY BUFF
     ER
4415 FOR J=1 TO P9 ! EACH GRAPH
4416 IF M(J,2)<M(J,3) THEN 4420
4417 PRINT @ PRINT "VARIABLE A("
     ;M(J,1);")HAS A" @ PRINT "C
     ONSTANT VALUE=";M(J,2)
4418 GOTO 4580
4420 PRINT @ PRINT "
                A(";M(J,1);")"
4430 GCLEAR
4435 IF T3>T4 THEN 4438
4436 X0=T3 @ X9=T4 @ GOTO 4440
4438 X0=T4 @ X9=T3
4440 SCALE X0,X9,M(J,2),M(J,3)
4450 XAXIS M(J,2),X1
4460 YAXIS X0,M(J,4)
4461 ASSIGN# 1 TO "P-DATA" ! RES
     ET
4465 J1=0
4470 J1=J1+1 @ READ# 1 ; T
4475 J2=0
4480 J2=J2+1 @ READ# 1 ; A2
4485 IF J2<J THEN 4480
4490 IF J1=1 THEN MOVE T,A3 ELSE
      DRAW T,A3
4495 IF J1=C1-1 THEN 4520
4500 IF J2=P9 THEN 4470
4505 J2=J2+1 @ READ# 1 ; A3
4510 GOTO 4500
4520 COPY
4530 PRINT
4540 PRINT "FOR T&A AXES:"
4550 PRINT "MIN ";X0;TAB(19);M(J
     ,2)
4560 PRINT "MAX ";X9;TAB(19);M(J
     ,3)
4570 PRINT "UNIT";X1;TAB(19);M(J
     ,4)
4580 PRINT USING 4581
4581 IMAGE 32("-")
4590 NEXT J
4600 PRINT J1;"POINTS PLOTTED"
4610 ALPHA
4620 RETURN
4630 ! ------------MODEL
4640 REM MAIN SUBR TO CALC F(i),
     K(i,z)FOR ALL i AND z=1,2,3
     ...
7960 FOR J=1 TO N0
7970 K(J,Z)=H*F(J) ! Z=1,2,3OR4
7980 NEXT J
7990 RETURN
8000 ! ------------INITIAL
8800 ! DATA FOR 1)INIT VALS
8850 ! DATA FOR 4)PRT LIST
8870 ! DATA FOR 6)PLOT INPT
8900 ! DATA FOR 10)TABLES
8970 ! DATA FOR 11)QUEUES
8999 RETURN
9000 ! ------------FINAL
9010 ! SUBR TO PERFORM ANY USER-
     DEFINED OPERATIONS AFTER A
     RUN
9999 RETURN
```

APPENDIX D

LIST OF SYMBOLS USED IN THE SIMULATION PROGRAM

The simulation program listed in appendix C is written in the computer language BASIC. The program variables allowed in BASIC are any letter from A to Z and any letter immediately followed by a digit 0 through 9. The variables described below are those used (and reserved) for the simulation program. Any variable, not appearing on the list, may be used when the model is coded in BASIC. Names appearing in the discussion which are written in capitals, for example 6) PLOT DATA, refer to modules of the simulation program. The various modules are listed in appendix A and described in detail in appendix B.

SYMBOL	DESCRIPTION
A	An array containing the latest values of the variables used in the model (A(1),A(2), ------A(N1). Only these variables are recognised by the simulation program.
AO	An array of initial values of all the variables. During simulation the values of A change and AO is used when it is desired to re-run the model using the same initial starting values (RE-INITIALISE).
A1	An array listing which elements of A are to be printed as output during simulation.
A2,A3 A4,A5	Variables used in various places in the program as temporary storage.
B	An array used in MAIN, which stores the latest variable values from the last iteration while the selected algorithm

searches for the values at the end of the current iteration.

B — A counter used in MAIN for the number of iterations performed.

C1 — A counter used in MAIN for the number of points stored in STO PLOT.

C8 — The approximate number of points to be plotted on each graph (*PLOT TIMES*)

C9 — The maximum number of graphs allowed (PLOT DATA).

D — FND is the interpolated table value required by MODEL from USE TABLE.

DO — Specified by MODEL when a time other than the current simulation time T is to be used for interpolating a variable value from a particular table in USE TABLE. The time used for interpolation is the current simulation time T plus DO. If DO is not specified in MODEL then DO=0 is assumed by USE TABLE.

D1 — A four column matrix where the row number equals the table number. Column 1 stores *number of times the particular table is* used; column 2 the first row number of matrix D2 containing the starting values for the particular table; column 3 the number of data pairs in the table; column 4 the first row number of matrix D3 containing the data pairs for the particular table.

D2 — The starting time values for each table

(see TABLE DATA).

D3 — A two column matrix. Column 1 stores the time increment values, after the start times in D2, and column 2 stores the table values.

D9 — The number of tables used by MODEL.

E — A counter used in MAIN to indicate when the specified variable values are to be stored in STO Q.

EO — Queue values are to be stored every EO time steps (calculated in MASTER and used by MAIN).

E1 — A three column matrix where the row number equals the queue number. Column 1 stores the variable number that the queue represents; column 2 stores the number of values in the queue; column 3 stores the row number of the array E2 containing the latest value of the variable represented by the particular queue (see Q DATA, STO Q and USE Q).

E2 — An array used to store the variable values of each of the queues. Matrix E1 is used to store data and extract information from E2.

E3 — A two column matrix where the row number equals a queue number. The initial values for each queue are stored in column 1 and the lag times (behind the current simulation time T) are stored in column 2 (see E1 and E2).

E7	The summation of all the lag times used in proportioning the storage space for each queue in array E2 (Q DATA).
E8	The total number of spaces specified in Q DATA for the storage of all the queue values in E2.
E9	The number of queues used by the model (Q DATA).
F	An array used to store the current values of each differential equation in MODEL.
FO	A flag used for printing output (FO=1 or 0 means yes or no).
F1	A flag used for plotting output (F1=1 or 0 means yes or no).
F2	A flag used to indicate which solution algorithm has been selected. (F2= 1,2 or 3 for the Euler, Modified Euler and 4th-Order Runge-Kutta algorithms respectively).
F3	A flag used for the queues (F3=1 or 0 means queues are or are not being used).
F4	A flag set by DATA/PROMPT indicating, to the input modules 1) to 11), whether data is to be read from data statements in INITIAL (F4=1) or entered via the keyboard (F4=2).
H	The time increment between simulation iterations.
I$	A string array used for non-numeric input (generally used for yes or no).

J,J1 -2	Counters used in various places in the program (generally used as row counters for arrays and matrices).
K	For each differential equation there are either 1,2 or 4 K-values associated with the solution algorithm. The K values are calculated at the end of MODEL and equal the differential values, F, multiplied by the time-step-size H.
L	Used as a flag and as a means to change the lag time for a particular queue. If L=0 or L is not specified in MODEL then the lag time originally specified in matrix E3 will be used. L specified in MODEL prior to a particular queue being called will result in a queue value being interpolated at a lag time L.
M	A four column matrix defined in PLOT DATA and used by PLOT OUTPUT for plotting graphs. The row number equals the graph number and columns 1 to 4 respectively are used to store the variable number of the graph, the minimum and maximum values of the variable axes and the spacing between tic marks along the variable axis.
NO	The number of differential equations (MASTER).
N1	The number of variables used in the differential equations and in the model (greater than or equal to NO).
N2	The number of variables to be printed after each SO iteration time steps (PRT LIST).

N3	The number of variables whose initial and current values must be changed in 3) CHANGES).
P	A counter used in MAIN for determining when data must be stored in STO PLOT for later plotting.
PO	Data for plotting must be stored every PO iteration time steps.
Q	FNQ is the interpolated/extrapolated queue value required by MODEL from USE Q.
R1,R2,R3	Counters for row numbers used at various places in the program for row numbers of arrays and matrices.
S1	Used for temporary storage of values in USE TABLE.
SO,S	MAIN calls PRT OUTPUT every SO simulation time iterations to print results during the simulation. S is the counter.
TO,T9	The starting and completion times for the simulation (SIMUL TIMES).
T1	Time at the end of the last iteration (MAIN).
T2,T8	The times specified in PRT TIMES for printed output to start and end.
T3,T4	The times specified in PLOT TIMES for storage and plotting of data to start and end.

T5	The table number specified by MODEL when calling USE TABLE.
T6,T7	Used as temporary storage in the interpolation of values from tables and queues in USE TABLE and USE Q.
T8,T9	See T2 and T0 above.
V6, V7	*Used as temporary storage for the* interpolation of values from tables or queues in USE TABLE and USE Q.
X	Interpolation time for calculating table values in USE TABLE.
X0,X9	The minimum and maximum values of the time axes of the graphs plotted by PLOT OUTPUT.
X1	The spacing of tic marks along the time axis of graphs (PLOT DATA).
Z	A counter used by each solution algorith for the number of times MODEL is called to calculate the K-values during each iteration. Z=1 for EULER, Z=1, 2 for MOD.EULER and Z=1, 2, 3 and 4 for 4R-KUTTA modules.
Z1	A flag set by MASTER and used by INPUT. Z1=0 tells INPUT not to give the user the option to change any input until all the initial data have been entered. Z1=1 allows input to display the various options for changing and correcting input.
Z2	Do-loop counter in MASTER defining the input module option to be called by INPUT.

APPENDIX E

SIMULATION PROGRAM USER INSTRUCTIONS

The mathematical model of a system can be simulated once it has been coded in BASIC language and the input data is available. Guidelines are given in chapter 5 on the development and use of simulation models. Repeated simulations are normally required before a model is available for experimentation. This section describes in detail how to use the simulation program to simulate a given model.

The flow-chart shown in fig.E1 describes the procedure followed when using the simulation program. The flow-chart is briefly explained and certain procedures and aspects are discussed in greater detail below.

E1 FLOW-CHART DESCRIPTION (FIG E1)

The less obvious flow-chart symbols have the following meaning: a rectangle depicts some action on the part of the user; the small circles are connectors to different parts of the flow-chart; the large parallelogram depicts user input or simulation output; the cigar-shaped symbol indicates a point in the flow-chart where the mode changes. The procedure is briefly as follows:

1) The Hewlett Packard HP85 is switched on, setting the HP85 to programming mode. The tape cartridge is then inserted. Either a file containing the simulation program and an existing model can be loaded or a blank copy of the simulation program containing no model can be loaded. If a blank copy is loaded then the programming statements of the model must be entered in the module called MODEL and any pre- and post-simulation

FIG E1 How to Use the Simulation Program (letters above flow-chart symbols are references)

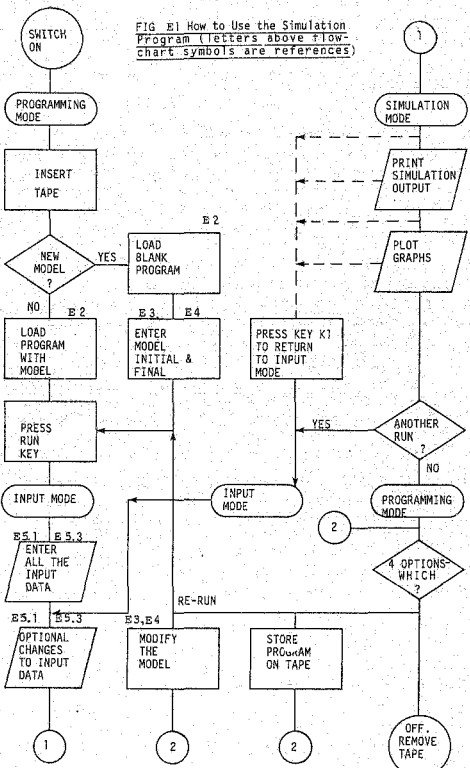
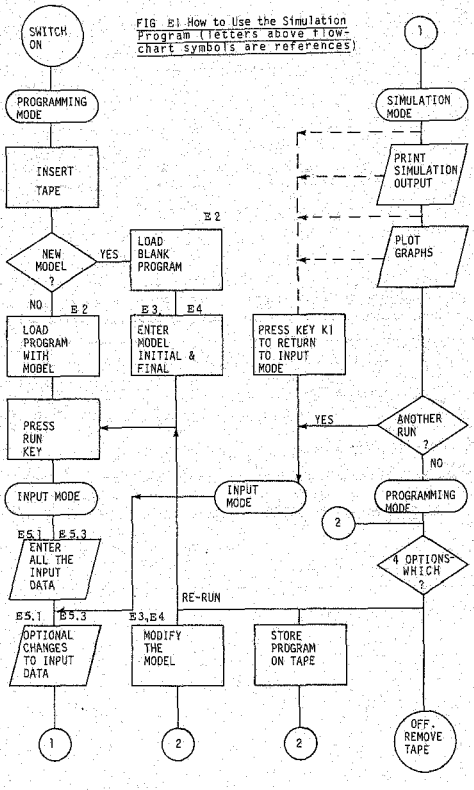


FIG B1 How to Use the Simulation Program (letters above flow-chart symbols are references)



instructions can be entered in the modules INITIAL and FINAL.

2) The RUN-key is pressed and the mode changes from programming mode to input mode in which no programming changes can be made. All the input data required for simulation and output are entered by answering questions appearing on the screen. When all the input data have been entered, a list of optional changes is displayed so that errors or changes can be corrected.

3) Once all the changes have been made the last option to simulate is selected changing the mode to simulation mode. The simulation will be carried out with the results being printed and/or the graphs plotted. At any time during simulation mode, the simulation can be aborted by pressing key k1 whereupon control will be transfered to input mode where changes can be made and another simulation done.

4) If the simulation is allowed to proceed to completion then the option exists of getting back into input mode (by replying 'yes' to another run) or into programming mode (by replying 'no'). Once back in programming mode changes can be made to the program, MODEL, INITIAL or FINAL but all variables become undefined and the RUN key must again be pressed to get into input mode and all the input data must again be entered.

Above some of the symbols in the flow chart is a reference to sections below where the steps are discussed in greater detail. The examples in chapters 6,7 and 8 illustrate the procedures discussed here and also contain further explanation. The programming language used throughout is BASIC and all statements and commands used in the examples are covered in the manual (HP-85 Owners Manual and Programming Guide (July 1980)).

E2 LOADING AND STORING PROGRAMS

The blank simulation program, containing no model, is loaded into program memory with the command

LOAD "SIMULN"

"SIMULN" is secured against overwriting. Once the model and pre- and post-simulation instructions have been programmed, the combined program can be stored on tape for future use. The program is given a name, "UNISEL" for example, up to six letters long, and stored with the command

STORE "UNISEL"

The same program can simply be loaded in future with the command

LOAD "UNISEL"

E3 ENTER THE MODULES MODEL, INITIAL & FINAL DURING PROGAMMING MODE

The three modules are dealt with separately below.

E3.1 Entering the Model into the Module MODEL.

1) After the model has been coded in BASIC it is entered, line by line, into program memory using any line numbers between 4640 and 7960.

2) Where there is a possibility that numbers (parameters and variables) used in the program may be changed between simulations or displayed or plotted during simulation, then they should be allocated a position in array A. Only the elements of array A, that is A(1), A(2), A(3) ---, are recognised by the simulation program.

3) Standard BASIC variables, for example Z3, A9, I, can be used in the model so long as that variable does not appear on the list of program symbols listed in appendix D. The initial value of these symbols must be defined in the

model itself or in INITIAL and cannot be displayed as output by the simulation program.

4) Each of the differential equations F(1), F(2), ---- can be a function of time T, other differential equations F(i), program variables A(j) or standard BASIC variables used in the model. For example F(3) meaning dA(3)/dt may be entered on line 4690 and look like this

```
4690 F(3) = Z6*(F(9)/T+A(7)+A(D8)).                    (E-1)
```

5) The order in which the differential equations appear in the model and the sequence in which variable values are calculated may be important. When MODEL is called calculation proceeds from top to bottom and if, for example, in equation (E-1) above, F(9) and A(7) are calculated on line numbers after 4690 then the values of F(9) and A(7) from the last iteration will be used in the calculation of F(3). If the calculation for F(9) and A(7) appears before F(3) in the model then the current iteration values will be used in the calculation of F(3).

6) The current simulation time is given by the variable T which is defined by the simulation program and used anywhere in the model.

7) The table value at the current simulation time can be used by simply including FND(table number) in the calculation. For example,

```
4720 A(65) = FND(2) * B1
```

uses the value of table 2(at time T) to calculate A(65). The example in chapter 8 uses 4 tables. The interpolation time can be changed by defining an add-on time DO in the model somewhere before the FND statement. For example

```
4720  DO = - 2@ A(65) = FND(2) * B1
```

would result in a value being interpolated from table 2 at a time two units (e.g. 2 hours) or whatever earlier than the current simulation time T. DO is set to zero each time before

MODEL is called therefore it must be defined each time in the model if a time other than T is to be used. DO *is not* illustrated in the examples in chapters 6 and 7 as table values are only required at T.

8) The value of a variable at some time earlier in the simulation can be interpolated from the values stored in a queue by simply including FNQ(queue number) in a calculation. For example, the calculation

```
5330 A(82) = FNQ(1)/A(66)
```

uses the value interpolated from queue number 1. The example in chapter 7 uses one queue (see section 7.4.2.1). The variable whose values are stored in the queue(s) and the lag time(s) is specified during input. The model can override the lag time specified by defining a new lag time L somewhere before the FNQ statement, e.g.

```
5330 L = 3 @ A(82) = FNQ(1)/A(66)
```

would result in a queue value being interpolated from queue number 1 using a lag time of three units (e.g. hours. If insufficient values are stored in the queue and interpolation is not possible because L is too large then execution is terminated with an error message and the computer reverts to programming mode. The original lag time specified is used whenever L is not defined by the model in an iteration. L is not used in any of the examples in chapters 6, 7 and 8.

9) It is often useful to include an integrator in the model to calculate cumulative values such as flow volumes, salt masses etc. An example of how this is done is discussed in section 7.4.3.

E.3.2 Entering Pre-Simulation Instructions and Input Data into the Module INITIAL

INITIAL is called by the simulation program immediately

after all input data has been entered and just before simulation begins. The execution of INITIAL is done while in simulation mode.

1) Any BASIC program instructions can be entered between lines 8010 and 8800.

2) The following are typical uses for INITIAL: print the model name; describe the important parameters used and give their values; print important initial values; set I/O and initialise any standard BASIC variables used in the model; give instructions to the user; request extra input; set a timer (see HP 85 manual); dimension extra arrays or matrices required by the model; enter a sub-routine in INTIAL which is required by the model.

3) The same rules as listed in E3.1 above apply for program variables A(1), A(2) ---- and standard BASIC variables used in INTIAL.

4) Perhaps the most important use of INITIAL is the use of reserved lines where input data can be entered as data statements. Data statements are a permanent record of the input data which, when entered, can be stored with the program on tape and is immediately available whenever the program is loaded at some later date (see E2 for load and store). Whenever a new model is being developed and an error is encountered by the computer during simulation, it will report the error and return the user to programming mode. All input data is lost and must be re-entered into computer memory whenever programming mode is entered. This is quickly done by reading the input data from the data statement instead of entering long lists of numbers each time the computer aborts a run. The disadvantage of using data statements is that extra memory is required which may be at a premium with a large model. Certain input data can be entered in data statements between the lines shown(the maximum values in brackets can be changed - see E4):

8800 - 8850 The initial values of all the program variables A(1), A(2), ---- (105 maximum)

8850 - 8870 a) The number of variables to be printed as output during simulation (50 maximum)

b) The variable numbers in the desired order.

8870 - 8900 a) The number of graphs to be plotted (20 maximum)

b) Data-File storage space for all the graph co-ordinates (maximum of 2000 graph points allowed in the present file).

c) Unit spacing marks along the time axis.

d) Enter the following 4 values for each graph in the order in which they must be plotted.

(i) Variable number
(ii) Minimum value on the variable axis
(iii) Maximum value on the variable axis
(iv) Unit spacing marks on the variable axis.

8900 - 8970 a) The number of tables used (10 maximum)

b) Enter the following 2 values for each table in turn.

i) number of times table can be used;
ii) number of data pairs in each table.

c) Enter the start times for each table in turn (maximum of 100 total for all graphs).

d) For each table in turn, enter each data pair in the table (maximum 100 pairs total for all graphs) consisting of the following two values:

i) time increment after the start time. In each table these values must start at 0 and each one must be greater than the previous one otherwise an error message results.

ii) The table (or graph) value corresponding to the table time increment.

8970 - 9999 a) Number of queues used (20 maximum).

b) Enter the following 2 values for each queue in turn:

i) variable number

ii) lag time (or travel time).

c) Total spaces available for all queues (100 maximum)

d) Enter an initial value for each queue in turn.

During input mode the user can choose to enter data by reading it from the above predefined data statements or by typing it in through the keyboard or both. If the option to read from data statements is selected then a pointer is automatically set to the first line number of the relevant group above. If no data is present in that group then the incorrect data will be read from the first group below which does have data (if no groups below have data an error results). Section E3.3 below gives useful hints and chapter 7 gives examples of data in data statements.

E3.3 Entering Post-Simulation Instruction into the Module FINAL

This module is called by the simulation program immediately after the simulation is complete, and any graphs have been plotted and copied. The execution of the instructions in FINAL are carried out during the simulation mode.

Any BASIC program instructions can be entered between lines 9010 and 9999

The following are typical uses for FINAL: describe and print the value of important output; state the time-step size; start and end times of the simulation; algorithm used; further operating instructions; stop the inbuilt clock and state the time taken to complete the simulations; enter a subroutine used by the model.

The same rules as listed in E2.1 above apply for program and standard BASIC variables

E4. MAKING CHANGES TO THE SPACE REQUIREMENTS OF ARRAYS, MATRICES AND PLOTTING - DATA - FILE DURING PROGRAMMING MODE

When the program is run the following heading and notes are displayed:

```
CONTINUOUS SIMULATION OF A
SYSTEM DESCRIBED BY ORDINARY
DIFFERENTIAL EQUATIONS WITH
INITIAL BOUNDARY CONDITIONS.

M.C.HOLTON-WITS UNIVERSITY.
**********************************
NOTE
1)Press key k1 WHILE EXECUTING
  to get back into input mode
2)Space is provided inF&K for up
  to 20 differential equations
3)Space is provided in A,A0&B
  for up to 105 variables in the
  prgm
4)Space is provided in A1 for
  printing up to 50 variables at
  each step
5)File space provided for<=A000
  graph points.M has space for
  <=20graphs.
6)Space is provided in D1 for
  <=10 tables and in D2&D3 for a
  total of<=100start times and
  <=100data points
7)Space is provided in E1%E2for
  <=20queues and a total of 100
  values
**********************************
```

The notes, 2) to 7), give the maximum number of equations, variables, graphs etc. which are allowed with the current dimensions of arrays and matrics. It may be necessary in future to increase the number of queues for example or to reduce the storage requirements of all the arrays and matrices to make space for a larger model. The names of the relevant arrays and matrices are also given to facilitate dimension changes. During input mode, reference is often made to these notes and should any dimensions be changed, then the notes must be updated as well. Lines 110 to 150 of the simulation program can be listed with the command

```
LIST 110,150
```

or

```
PLIST 110,150
```

for a printed copy of the following:

```
118 REM DIM/DEF ARRAYS&VARBLS
115 OPTION BASE 1
120 REAL A(105),B(105),K(20,4),F
    (20)
130 SHORT A0(105),D2(100),D3(100
    ,2),E2(100),E3(20,2),M(20,4)
140 INTEGER A1(50),C,C1,C9,D1(10
    ,4),D9,E,E0,E1(20,3),E9,F0,F
    1,F2,F3,J,J1,J2,N0,N1,N2,N3
150 INTEGER P0,P,P9,S,S0,T5,V
```

The available memory space left (in bytes) can be found by executing the command

```
LIST 9999
```

This can only be done in programming mode and is best done after a simulation when all variables have been defined and available memory is a minimum. Pressing the PAUSE key during input or simulation mode will change the mode to programming mode. If either of the above two commands are used or programming changes are made, then the RUN key must be pressed to get back into input mode and all input must be re-entered.

The data file for storing plotting data can be increased to hold any number of points by simply creating a bigger data file on tape called "P-DATA". The present file has space for

2000 graph co-ordinates, that is, space for 20 graphs each with 100 points, for example, or space for 10 graphs with 200 co-ordinates each.

E5. ENTERING INPUT DATA AND MAKING CHANGES DURING INPUT MODE

E5.1 Input Data Required for the First Simulation Run

Certain input, 1) to 11) below is required for the first simulation. Some of the data listed is optional and will depend on the requirements of the model. The rest of the data is compulsory. During input of 1) to 11) below, those preceeded by an asterisk (*), will give the user the choice of reading the data from data statements (see E3.2 4)) or typing it in through the keyboard, i.e.

```
1)USE PREDEFINED DATA
2)KEYBOARD INPUT
(1or2)?
```

A record of all data entered can be printed on paper otherwise it is much quicker to use the display on the screen.

Useful hints concerning the input data which should result in more efficient and faster simulations are given in E5.3. The following list of input data is in the same order as it is requested during a first run simulation and should facilitate data preparation:

1) Number of differential equations in the model (20 max.)

2) Number of program variables in the model (105 max.)

*3) The initial values of all program variables

*4) The list of variables, if any, which must be printed during simulation (50 max.)

5) The time period and fequency if results are to be printed.

*6) If graphs are to be plotted then the following information is requested:

a) The number of graphs to be plotted (20 max.)

b) Data-File storage space for all the graph co-ordinates (maximum of 2000 graph points allowed in the present file).

c) Unit spacing marks along the time axis.

d) Enter the following 4 values for each graph in the order in which they must be plotted.

i) variable number
ii) minimum value on the variable axis
iii) maximum value on the variable axis
iv) unit spacing marks on the variable axis.

7) The start and finish time and the approximate number of points to be plotted on each graph.

8) The start and finish time and the step-size to be used for executing the simulation.

9) The algorithm to be used for numerically solving the differential equations. Either the Euler, Modified Euler or the Fourth-Order Runge-Kutta method can be used.

*10) If tables are to be used to represent the time relationship of certain program variables then the following data are required.

a) The number of tables used (10 max.)

b) Enter the following 2 values for each table in turn:

i) number of times table can be used.

ii) number of data pairs in each table.

c) Enter the start times for each table in turn (maximum of 100 total for all graphs).

d) For each table in turn, enter each data pair in the table (maximum 100 pairs total for all graphs) consisting of the following two values:

i) time increment after the start time. In each table these values must start at 0 and each one must be greater than the previous one otherwise an error message results.

ii) The table (or graph) value corresponding to the table time increment.

*11) If the model requires the use of queues to store past values of a variable during simulation then the following data is required:

a) Number of queues used (20 max.)

b) Enter the following 2 values for each queue in turn:

i) variable number

ii) lag time (or travel time)

c) Total spaces available for all queues (100 max.)

d) Enter an initial value for each queue in turn.

After the above data has been entered, the following list of options is presented so that any mistakes made above can be corrected or any last-minute changes can be made:

```
1)CHANGE ALL INIT. VALUES
2)RE-USE INIT.VARIABLES&QUEUES
3)CHANGE SOME VARIABLE VALUES
4)CHANGE PRINT LIST
5)CHANGE PRINT TIMES
6)CHANGE PLOTTING DATA
7)CHANGE TIMES FOR PLOTTING
8)CHANGE EXECUTION TIMES
9)CHANGE ALGORITHM
10)CHANGE FUNCTION TABLES
11)CHANGE QUEUE PARAMETERS
12)EXECUTE PROGRAM
(LAST OPTION 12 )
ENTER NEW OPTION 1TO 12?
```

At this stage, only option 2) is not applicable. Option 12) is selected after any changes have been completed simulation begins (simulation mode).

E 5.2 Input Data After the First Simulation Run

Input mode is entered after a completed simulation or after the escape key k1 has been pressed during simulation. The same list of 12 options, as shown above, is again presented and the user can make any of the changes shown and do another simulation by selecting the last option, number 12.

When input mode is entered from simulation mode, the current variable values from the simulation are retained. If, for example, option 12 was selected, without any changes, then the identical simulation would be carried out but the salt concentrations would continue to build up and dam volumes change etc., as it would have if the previous simulation had been twice as long. It should be noted that simply pressing 12 will use the same start and finish time for execution, printing results and printing graphs. If the identical simulation is to be carried out to obtain the same output as before then option 2 will reinitialise all current variable values and values stored in queues to those originally entered. Option 3 is used to change the current value and the value in the original list of variables.

The only input data that changes between simulations is the current value of the program variables and the values stored in queues. All other data, initial values, print list and times, plotting data and times, execution times, algorithm function tables, queue specifications (not values) only change when the appropriate option 1 to 11 is selected. This allows great flexibility during simulation. For example, simulation can be done in stages using larger time steps or a faster less accurate algorithm at different stages as the model approaches steady state conditions.

E5.3 Data Input Hints for Faster, More Efficient Simulation

1) Option 5 The printer is very slow and if frequent printing of a large print list is required then graphical presentation may be better. Alternatively, the values could be displayed on the screen ("Print or display results?", is asked just before simulation).

2) The simulation can be temporarily halted by pressing the PAUSE key. The screen display can then be copied onto paper by pressing the COPY key. The current value of any variable used in the program or model can be found during the interruption by typing the variable name, for example A(98) or Z3 followed by pressing the END OF LINE key. Similarly, the value of any variable can be changed during the interruption by typing A(98) = 1,036, for example and pressing the END OF LINE key. The simulation is continued by pressing the CONT key.

3) Options 6 and 7 The plotting data for all graphs is stored on tape file during simulation. At the end of the simulation the tape is repeatedly rewound and searched so that the graphs can be plotted one at a time.

The screen has a resolution of 256 dots along the time axis and there is no benefit in specifying a number greater than 256 for the number of points to be plotted on each graph. The less points required, the faster the simulation and plotting and the less the tape wear will be. The number of points should depend on the nature of the graph. Smooth graphs with gradual changes would only require approximately 25 points whereas an intricate graph with large fluctuation would require many more points to avoid missing detail. If the time span is too large and the graph too compressed then the simulation should be carried out in a number of shorter time stages with a plot at the end of each. The maximum number of points available for plotting is

$$\frac{\text{The time span for plotting}}{\text{Iteration time-step H}} + 1.$$

Choose any factor of this number for the number of points to be plotted.

The minimum and maximum values on the variable axis are automatically decreased or increased if a point to be plotted is beyond the range specified. Often the range of values to be plotted is not known and this feature of the program can be used to ensure that the maximum point plotted is at the top of the graph or the minimum value is at the bottom or both. To ensure that both values are at the extremity of the graph, for example, set the minimum value equal to 100000 and the maximum value to zero. In this way, the graphs can also be used to find the maximum and minimum value of a variable during the plotting period of the simulation.

4) Option 8 The accuracy of the simulation results depends on the iteration time step. Generally speaking, smaller time-steps are required when the model is in a state of rapid change than when the model is near or at equilibrium. The unsteady salt and flow model in chapter 7 is never really in equilibrium as salt concentrations, flows and volumes are constantly changing. The objective is to choose the largest iteration time step which will ensure the desired accuracy. This is probably best accomplished by reducing the time step until the change in results becomes insignificant. Further discussion on this aspect is covered in chapter 3.

5) Option 9 The Fourth-Order Runge-Kutta algorithm is many times more accurate and efficient than the Modified Euler algorithm which, in turn, is more accurate than the Euler algorithm. Simulations with Runge-Kutta take about twice as long as those with Modified Euler and four times

as long as those with the Euler method. A large time step in a simulation may mean, for example, that a nearly full dam that is filling at the start of an iteration will be filled far beyond its capacity at the end of the iteration. In such a case, the time step will have to be decreased for reasons of validity of the model and not for reasons of accuracy. The more powerful methods used on the program, the Runge-Kutta and Modified Euler methods, may be unsuitable at times because smaller iteration time steps are required for reasons other than accuracy and would result in excessive simulation times otherwise. The user would have to experiment to obtain the optimum time-step-algorithm combination for the desired accuracy which would minimise the simulation time.

6) Option 11 The latest variable values are stored in all the queues used, at the same time. Given the total number of spaces for the queues, the simulation program therefore calculates the number of spaces allocated to each queue in proportion to the lag time of each queue, with a minimum of two values in any one queue. The allocations are displayed for approval and the total queue space can be either increased or decreased and a re-allocation made. The maximum number of queue values that can be used for interpolation is the greater of 2 or

$$\frac{\text{lag (or travel) time}}{\text{iteration time step H}} + 1.$$

Any queue size in excess of this number represents an unnecessary increase in simulation time. Any value less than this means that values earlier in the simulation will be interpolated from values entered in the queues less frequently than every iteration. Therefore one selects the minimum number of queue values which will ensure that the discrete points stored in the queues adequately represent the functions occuring during the lag time.

REFERENCES

Binnie & Partners, 1981 Desalination Methods & Costs; Chamber of Mines Research Project GT2W12. Mine Service Water Quality Research Report No. BPI.

Grosman, D.D., Optimum Allocation of Mine Service Water Subject to Quality Constraints; Chamber of Mines Research Programme, University of the Witwatersrand, 1981.

Gerald, C.F., Applied Numerical Analysis;2nd ed. Addison Wesley, 1980.

Herold, C.E., Mineral Pollution of Water Resources of the Pretoria-Witwatersrand-Vereeniging Complex; Seminar on Urban Stormwater Management, December 1981, Pretoria.

IBM Application Program, General Purpose Simulation System /360-Users Manual; 5th edition, 1970.

IBM Application Program. System/360 Continuous System Modelling Program- Users Manual; 5th edition, 1972

James, W. Developing and Using Computer Simulation Models of Hydrological Systems; Vol. 1 Computational Hydraulics Inc. 1978.

Kelly, M., Structured Programming for Civil Engineers. Symposium - Computers in Civil Engineering, September 1980, Johannesburg.

Kopal, Z., Numerical Analysis Wiley, New York 1955.

Kreyszig, E., Advanced Engineering Mathematics;3rd. ed. John Wiley, 1972.

Loucks, P.D., et. al. Water Resource Systems Planning and Analysis; Prentice Hall, 1981.

Meta Systems, Inc. Systems Analysis in Water Resources Planning; A Water Information Center Publication, 1975.

Mitchell & Gauthier, Assoc., Inc., Advanced Continuous Simulation Language - User Guide/Reference Manual; 2nd ed. 1975.

Mize, J., and Cox, J., Essentials of Simulation; Prentice-Hall, 1968.

Olgata, K., System Dynamics; Prentice-Hall, 1978

Stark, R.M., and Nicholls, R.L., Mathematical Foundations for Design: Civil Engineering Systems; McGraw-Hill, 1972

Stephenson, D., Causes and Effects of Poor Quality Water, Chamber of Mines Research Programme, University of the Witwatersrand, 1981 a.

Stephenson, D., Water Purification Methods and Costs, Chamber of Mines Research Programme, University of the Witwatersrand, September 1981 b.

Stephenson, D., Holton, M.C., and Protheroe, B.E., Water Quality Observations in Doornfontein Gold Mine; Chamber of Mines Research Programme, University of the Witwatersrand, 1981.

Stephenson, D., The White Waters of the Witwatersrand; Inaugural lecture, Johannesburg, University of the Witwatersrand 1980.

Stewart, Sviridov & Oliver, 1981, Settling and Filtration Methods and Costs; Chamber of Mines Research Project GT2E12. Mine Service Water Quality Research Report No.SS01

Stummel, F., and Hainer, K., Introduction to Numerical Analysis;Scottish Academic Press Ltd., 1980.

Yourden, E., Techniques of Program Structure and Design; Prentice-Hall, 1975.