Description and manual for the use of DRIVER - an interactive modelling aid

P R Furniss

A report of the Savanna Ecosystem Project
National Programme for Environmental Sciences

SOUTH AFRICAN NATIONAL SCIENTIFIC PROGRAMMES REPORT NO

17

SEPTEMBER 1977

Issued by the
Cooperative Scientific Programmes
Council for Scientific and Industrial Research
P O Box 395
PRETORIA 0001
from whom copies of reports in this series are available on request.

*Printed 1977 in the Republic of South Africa
by the Graphic Arts Division of the CSIR*

ISBN 0 7988 1197 8

Author's address -

P R Furniss
Department of Applied Mathematics
University of the Witwatersrand
1 Jan Smuts Avenue
JOHANNESBURG 2001
South Africa

PREFACE

The Savanna Ecosystem Project of the National Programme for Environmental Sciences is one of several national scientific programmes administered by the CSIR. The National Programme is a cooperative undertaking of scientists and scientific institutions in South Africa concerned with research related to environmental problems. It includes research designed to meet local needs as well as projects being undertaken in South Africa as contributions to the international programme of SCOPE (Scientific Committee on Problems of the Environment), the body set up in 1970 by ICSU (International Council of Scientific Unions) to act as a focus of non-governmental international scientific effort in the environmental field.

The Savanna Ecosystem Project being carried out at Nylsvley is a joint undertaking of more than fifty scientists from the Department of Agricultural Technical Services, the Transvaal Provincial Administration, the CSIR, the Transvaal Museum, and eight universities. As far as possible, participating laboratories finance their own research within the project. The shared facilities at the study area and the research of participating universities and museums are financed from a central fund administered by the National Committee for Environmental Sciences and contributed largely by the Department of Planning and the Environment.

The research programme of the Savanna Ecosystem Project has been divided into three phases - Phase I (mid 1974 to mid 1976) - a pilot study of the Nylsvley study area, in particular the description and quantification of structural features of the ecosystem, Phase II (mid 1976-1979) - studies in the key components and processes including the development of mathematical models, and Phase III (1979-1984) - extension to other sites and the study of management strategies for the optimal utilization of *Burkea* savanna ecosystems.

This report describes a FORTRAN language computer programme which was developed within the Project. The programme has two main aims. Firstly, it enables researchers with little or no computer background to exercise existing mathematical models interactively after virtually no instruction. Secondly, it enables modellers to implement their models on a computer without having to concern themselves with input-output routines. This report is intended as an instruction manual for both these user groups.

CURRENT TITLES IN THIS SERIES

1. A description of the Savanna Ecosystem Project, Nylsvley, South Africa. December 1975. 24 pp.

2. Sensitivity analysis of a simple linear model of a savanna ecosystem at Nylsvley. W M Getz and A M Starfield. December 1975. 18 pp.

3. Savanna Ecosystem Project - Progress report 1974/1975. S M Hirst. December 1975. 27 pp.

4. Solid wastes research in South Africa. R G Noble. June 1976. 13 pp.

5. Bibliography on marine pollution in South Africa. D A Darracott and C E Cloete. June 1976. 131 pp.

6. Recycling and disposal of plastics waste in South Africa. R H Nurse, N C Symington, G R de V Brooks and L J Heyl. June 1976. 35 pp.

7. South African Red Data Book - Aves. W R Siegfried, P G H Frost, J Cooper and A C Kemp. June 1976. 108 pp.

8. South African marine pollution survey report 1974-1975. C E Cloete and W D Oliff (editors). September 1976. 60 pp.

9. Modelling of the flow of stable air over a complex region. M T Scholtz and C J Brouckaert. September 1976. 42 pp.

10. Methods and machinery for pulverising solid wastes. M J Simpkins. October 1976. 29 pp.

11. South African Red Data Book - Small mammals. J A J Meester. November 1976. 59 pp.

12. Savanna Ecosystem Project - Progress report 1975/1976. B J Huntley. March 1977. 41 pp.

13. Disposal and recovery of waste paper in South Africa. G R de V Brooks. April 1977. 35 pp.

14. South African Red Data Book - Fishes. P H Skelton. July 1977. 39 pp.

15. A checklist of the birds of the Nylsvley Nature Reserve. W R Tarboton. September 1977. 14 pp.

16. Grondsoorte van die Nylsvley-natuurreservaat. H J von M Harmse. September 1977. 64 pp.

17. Description and manual for the use of DRIVER - an interactive modelling aid. P R Furniss. September 1977. 23 pp.

ABSTRACT

The modelling aid DRIVER is described. It permits the interactive manipulation of the parameters and variables of difference models which are implemented as FORTRAN subroutines. Relationships in the model can be expressed as arbitrary functions. A choice of output formats is available. The arbitrary functions and the output can also be changed interactively. Parameter, variable and function values can be stored on disk. The use of the commands for DRIVER and how to arrange a model for use with DRIVER is explained.

UITTREKSEL

'n Beskrywing van die modelleringshulpmiddel DRIVER word gegee. DRIVER voorsien die wisselwerkende manipulering van parameters en veranderlikes van wiskundige modelle, wat as FORTRAN subroetines geïmplementeer is. Verwantskappe in die model kan as arbitrêre funksies uitgedruk word. 'n Keuse van uitvoer formate is beskikbaar. Die arbitrêre funksies en die uitvoer kan ook gedurende 'n lopie verander word. Parameter, veranderlike en funksie waardes kan op skyf geberg word. Die gebruik van bevele vir DRIVER en die aanpassing van 'n model vir gebruik met DRIVER word beskryf.

TABLE OF CONTENTS Page

INTRODUCTION

The DRIVER programme described in this report has been written to aid in the interactive use and manipulation of ecological models on a computer. The description is divided into several sections and it is anticipated that readers with different interests will read different sets of sections. Readers who wish merely to know the broad capabilities of DRIVER, without using it themselves, need only look at "Purpose and capabilities of the DRIVER programme" and Appendices 2 and 3, though they may find "Meanings and use of DRIVER commands" useful. Those wishing to use DRIVER, with a model that is already programmed need to read all of these sections. The section "Setting up a model for use with DRIVER" contains information required for fitting a model to DRIVER.

Certain aspects of DRIVER are machine dependent, and supplements to this guide are available for the installations on which DRIVER operates. At the time of writing, these are the University of the Witwatersrand's IBM 370/158, using the WITS time-sharing system and the Agricultural Technical Services' Burroughs B7700. Listings of DRIVER, with more detailed explanations of the programme may be obtained from the South African Savanna Ecosystem Data Coordinator[1].

DRIVER may be implemented on any machine capable of executing a FORTRAN programme interactively. It is also necessary for the programme to read and write to disk files. Minimum core requirements are not known, as both the machines mentioned above are large. DRIVER has been written for use with a VDU terminal (a cathode ray tube screen with a keyboard), but can be used with a teletype. DRIVER can also produce output for a printer while running interactively.

PURPOSE AND CAPABILITIES OF THE *DRIVER* PROGRAMME

Models suitable for use with DRIVER

DRIVER is designed for use with medium and small models based on difference equations. In such models, time is considered in discrete steps or iterations. At any one time the status of the system is determined by the values of the *state variables*. These correspond to measureable properties of the modelled system. The essential features of the state variables are that they change with time and that their values at a particular time affect the future behaviour of the system, because their present values are the starting point for the determination of their own future values.

In addition to the state variables there will be other measureable properties of the system which change with time but which, in the model, are calculated afresh at each time interval, usually from the state variables. Although their values may then affect the state variables these values are not themselves carried on to the next time interval. These qualities will be termed merely *other variables* (for the time being).

[1] Botanical Research Institute, Private Bag X101, Pretoria 0001

The structure of the model, that is how these variables affect each other is defined in a computer programme, which is merely a set of rules for calculating the next set of values for all the variables, from the present values of the state variables and from other, constant values known as parameters.

To clarify this, and the description of DRIVER, we will consider an extremely simple model of the production and disappearance of ungulate dung from the Nylsvley *Burkea* system. This was constructed during the Savanna Ecosystem Project's First Modelling Workshop in January 1977, and a more detailed report on it is to be found in the report on the Workshop. The time unit of the model is one week. The only state variable is the quantity of dung present. Some of the "other" variables are the week's loss and production of dung and the total number of cattle present. The parameters include the dung production per unit of cattle biomass and per unit of wild ungulate biomass. The model's structure defines, for example, that the loss is subtracted and the production added to the quantity of dung present. A listing of the FORTRAN implementation of the model is shown in Appendix 1.

When a model is constructed of a real ecological system, there is typically uncertainty as to the structure, and the values of the parameters and variables. In such a case it is desirable to make frequent changes to these and observe the resulting behaviour of the model. Even when the structure and standard values are known with some confidence, it is of interest to investigate the model's response to alterations. In both cases the capability of working with the model in an interactive environment is very useful. DRIVER enables the parameter and variable values to be changed very easily and also permits a choice of several different forms of output. In can be used by someone without any knowledge of computer programming as such.

Concepts involved in DRIVER

The conceptual structure of DRIVER is shown in Figure 1. It will be seen that the DRIVER programme (in fact a set of subroutines) communicates between several entities - the user at the terminal, the model (programmed as a subprogramme of DRIVER), a set of information known as the reference information and copies of the reference information on disk. Most of the communication between the model, DRIVER and the terminal is via the reference information. As mentioned in the previous section, the models with which DRIVER can be used have a basic time step or iteration interval (one week in the example). The operation of the model is a sequence of such iterations, termed a *run*. The number of iterations in a run is user-defined, and referred to as the *runtime*. The runtime is in units of simulated time (ie iterations), which may be days, years, generations or any other time unit. Figure 2 shows typical output from a run of the example model.

Within any one run the *parameters* of the model will be fixed, though they may be varied by the user between runs. The *variables* of the model will vary within a run, typically being redefined each iteration. Some of the variables will be state variables of the system, which must be given a meaningful value at the beginning of a run. For "other" variables the previous value will not affect the present or future behaviour of the

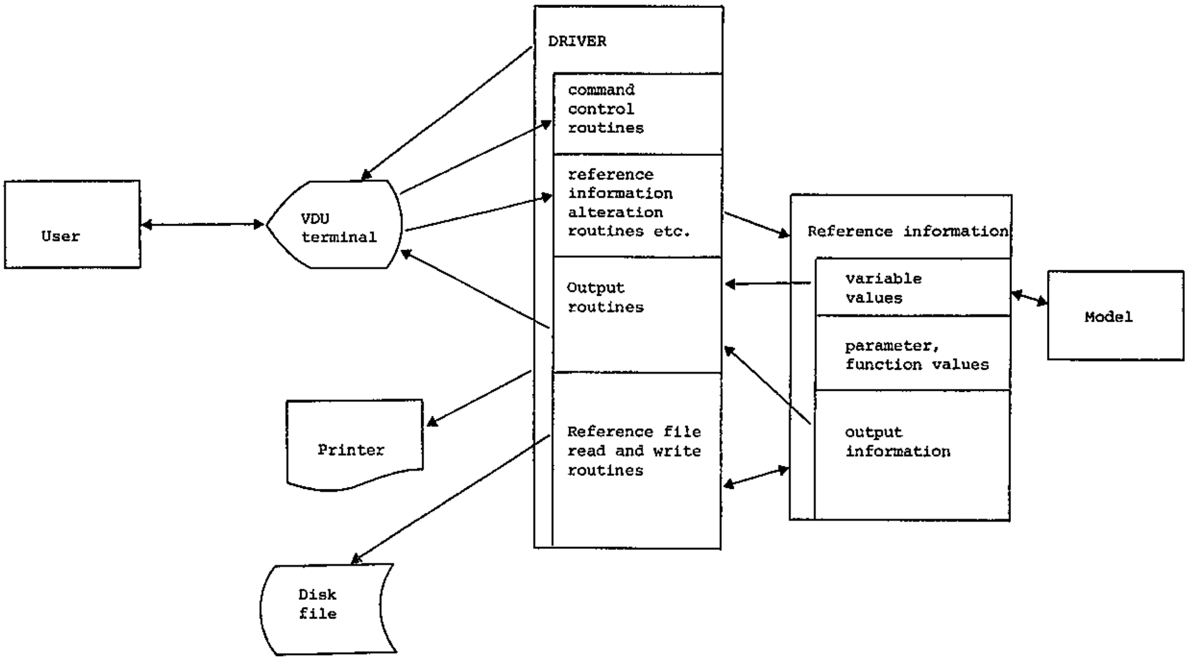
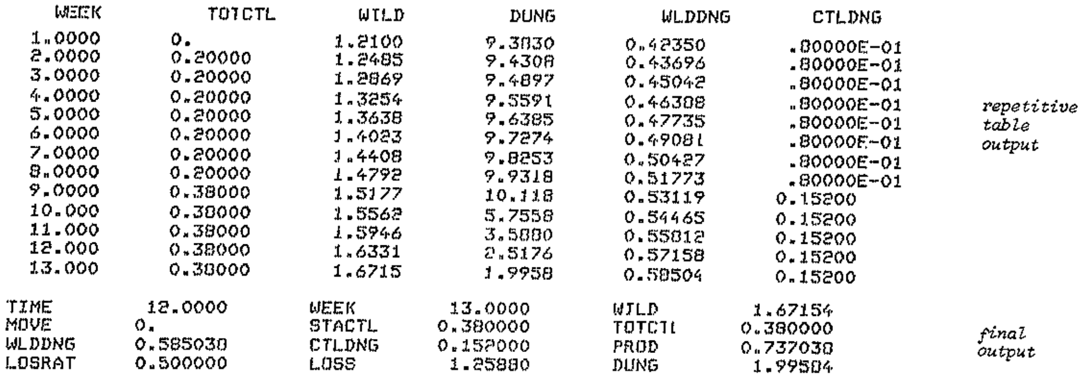


Figure 1. Conceptual structure of DRIVER programme.

model. DRIVER does not distinguish between state and "other" variables, but refers to them all as "variables". Thus "other" variables have an initial value, defined in DRIVER, but not used in the model. This description will no longer distinguish between state and other variables.



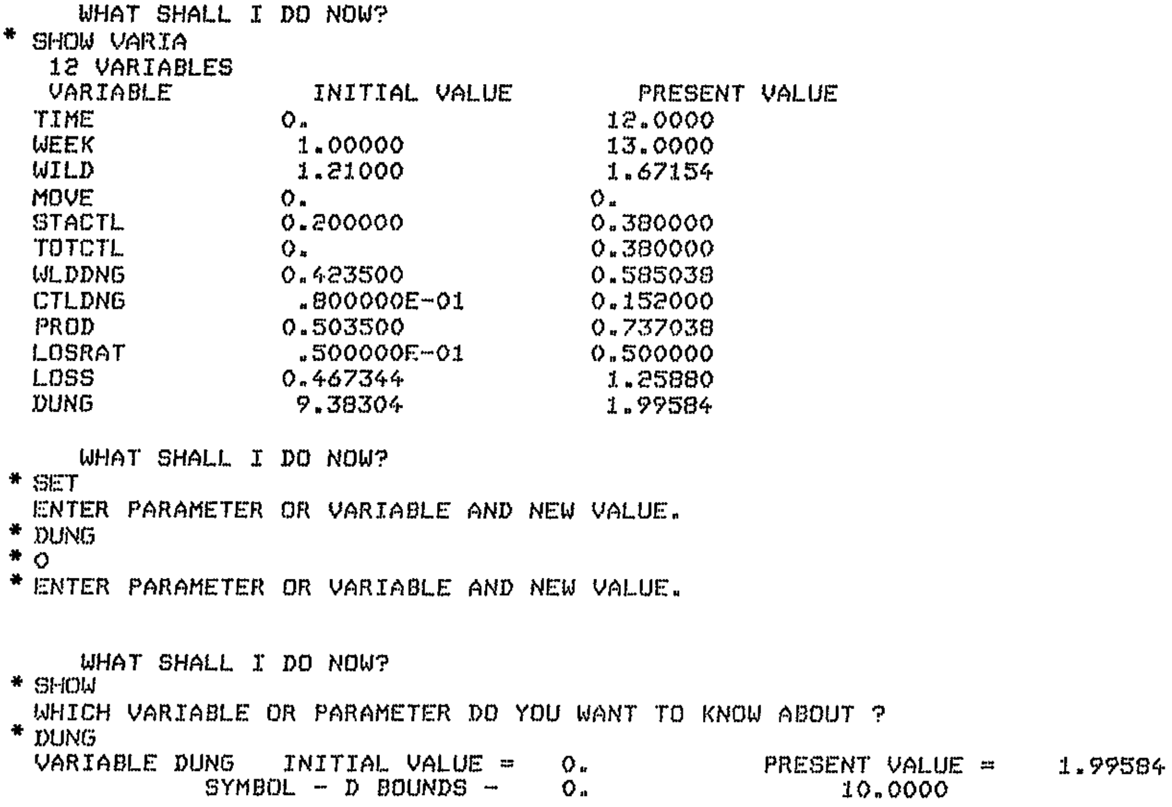
```
  WEEK        TOTCTL        WILD         DUNG        WLDDNG        CTLDNG
 1.0000     0.            1.2100       9.3830      0.42350      .80000E-01
 2.0000     0.20000       1.2485       9.4308      0.43696      .80000E-01
 3.0000     0.20000       1.2869       9.4897      0.45042      .80000E-01
 4.0000     0.20000       1.3254       9.5591      0.46308      .80000E-01
 5.0000     0.20000       1.3638       9.6385      0.47735      .80000E-01
 6.0000     0.20000       1.4023       9.7274      0.49081      .80000E-01
 7.0000     0.20000       1.4408       9.8253      0.50427      .80000E-01
 8.0000     0.20000       1.4792       9.9318      0.51773      .80000E-01
 9.0000     0.38000       1.5177       10.118      0.53119     0.15200
 10.000     0.38000       1.5562       5.7558      0.54465     0.15200
 11.000     0.38000       1.5946       3.5880      0.55812     0.15200
 12.000     0.38000       1.6331       2.5176      0.57158     0.15200
 13.000     0.38000       1.6715       1.9958      0.58504     0.15200

TIME        12.0000       WEEK       13.0000       WILD        1.67154
MOVE       0.             STACTL    0.380000       TOTCTL     0.380000
WLDDNG     0.585030       CTLDNG    0.152000       PROD       0.737038
LOSRAT     0.500000       LOSS       1.25880       DUNG        1.99504
```

repetitive table output

final output

Figure 2. Example run showing table output.

Time is regarded as a normal variable by DRIVER, though it is used in a distinctly different manner to the other variables in the model routine. Parameter and variable values can be changed with the command "set", and their values inspected by entering other commands, as shown in Figure 3. The parameters and variables are referred to individually by a name.



```
     WHAT SHALL I DO NOW?
* SHOW VARIA
   12 VARIABLES
   VARIABLE          INITIAL VALUE         PRESENT VALUE
  TIME            0.                     12.0000
  WEEK             1.00000               13.0000
  WILD             1.21000               1.67154
  MOVE            0.                    0.
  STACTL          0.200000               0.380000
  TOTCTL          0.                     0.380000
  WLDDNG          0.423500               0.585038
  CTLDNG           .800000E-01           0.152000
  PROD            0.503500               0.737038
  LOSRAT           .500000E-01           0.500000
  LOSS            0.467344               1.25880
  DUNG             9.38304               1.99584

     WHAT SHALL I DO NOW?
* SET
  ENTER PARAMETER OR VARIABLE AND NEW VALUE.
* DUNG
* 0
* ENTER PARAMETER OR VARIABLE AND NEW VALUE.


     WHAT SHALL I DO NOW?
* SHOW
  WHICH VARIABLE OR PARAMETER DO YOU WANT TO KNOW ABOUT ?
* DUNG
  VARIABLE DUNG   INITIAL VALUE =   0.          PRESENT VALUE =    1.99584
             SYMBOL - D BOUNDS -    0.                 10.0000
```

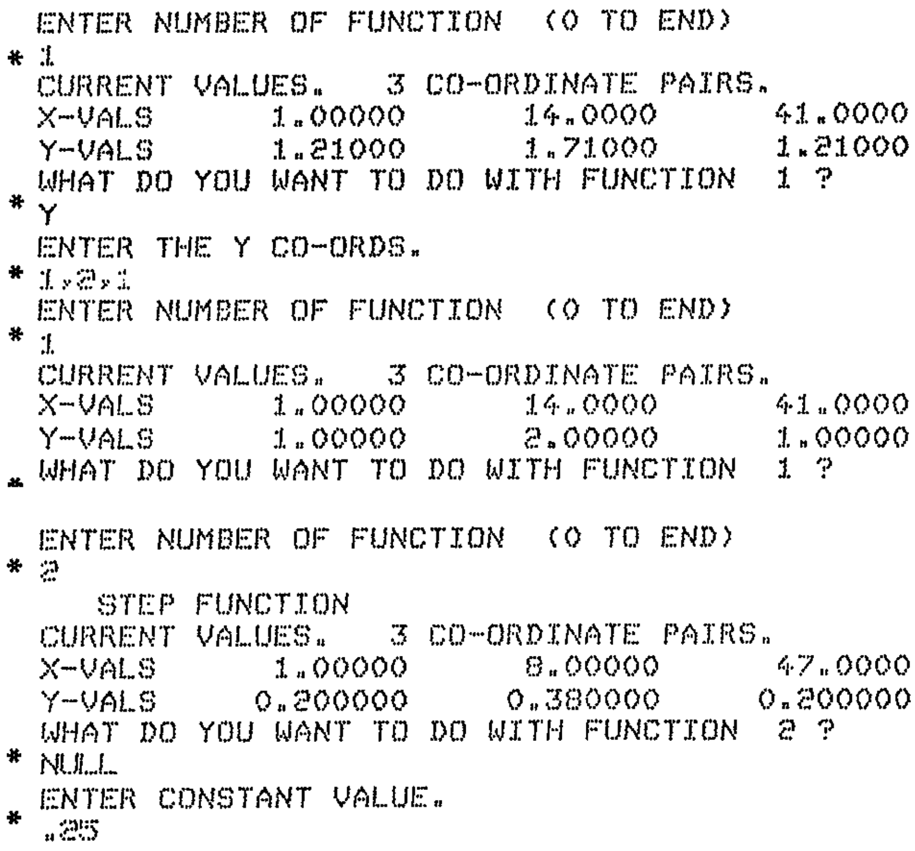
Figure 3. Examining and changing variable values (* are entries by user).

In addition to the structure and relations of the model defined rigidly in the programme, DRIVER allows the use of *arbitrary functions*. These are of the general form -

y = f(x)

where x and y are variables, or just internal values, of the model. The functions are defined by linear interpolation between a set of coordinates. Up to 6 coordinates may be used. The arbitrary functions are of great use when only the approximate form of a relationship is known or when the relationship is empirically derived from actual data. In such cases the modeller has the choices of using linear interpolation, as in the arbitrary functions, or finding a mathematical expression which is then fitted to the data. Both methods have advantages, interpolating allowing great freedom of alteration and often being clearer to the non-mathematical. The co-ordinates can be changed by the user of DRIVER. The arbitrary function routine (called "FF") can also be used to produce a step function instead of performing interpolation. A third possibility is to make the function return a constant value (regardless of the value of x).

In the example, the effect of time of year on the biomass/unit area of wild ungulates and on the dung loss rate are arbitrary functions. Both are usually used as step functions. Inspecting the previously assigned values of the arbitrary functions, and changing them is achieved with a group of commands. With these it is possible to change the number of line-segments in the function, or to convert it from an interpolating to a step function, or *vice versa*. The particular values of f(x) for an entered x can also be found. Figure 4 shows a segment of dialogue performing some of these changes. It will be seen that the functions are referred to by number.



```
  ENTER NUMBER OF FUNCTION  (O TO END)
* 1
  CURRENT VALUES.   3 CO-ORDINATE PAIRS.
  X-VALS        1.00000        14.0000        41.0000
  Y-VALS        1.21000        1.71000        1.21000
  WHAT DO YOU WANT TO DO WITH FUNCTION  1 ?
* Y
  ENTER THE Y CO-ORDS.
* 1,2,1
  ENTER NUMBER OF FUNCTION  (O TO END)
* 1
  CURRENT VALUES.   3 CO-ORDINATE PAIRS.
  X-VALS        1.00000        14.0000        41.0000
  Y-VALS        1.00000        2.00000        1.00000
* WHAT DO YOU WANT TO DO WITH FUNCTION  1 ?

  ENTER NUMBER OF FUNCTION  (O TO END)
* 2
     STEP FUNCTION
  CURRENT VALUES.   3 CO-ORDINATE PAIRS.
  X-VALS        1.00000        8.00000        47.0000
  Y-VALS       0.200000       0.380000       0.200000
  WHAT DO YOU WANT TO DO WITH FUNCTION  2 ?
* NULL
  ENTER CONSTANT VALUE.
*  .25
```

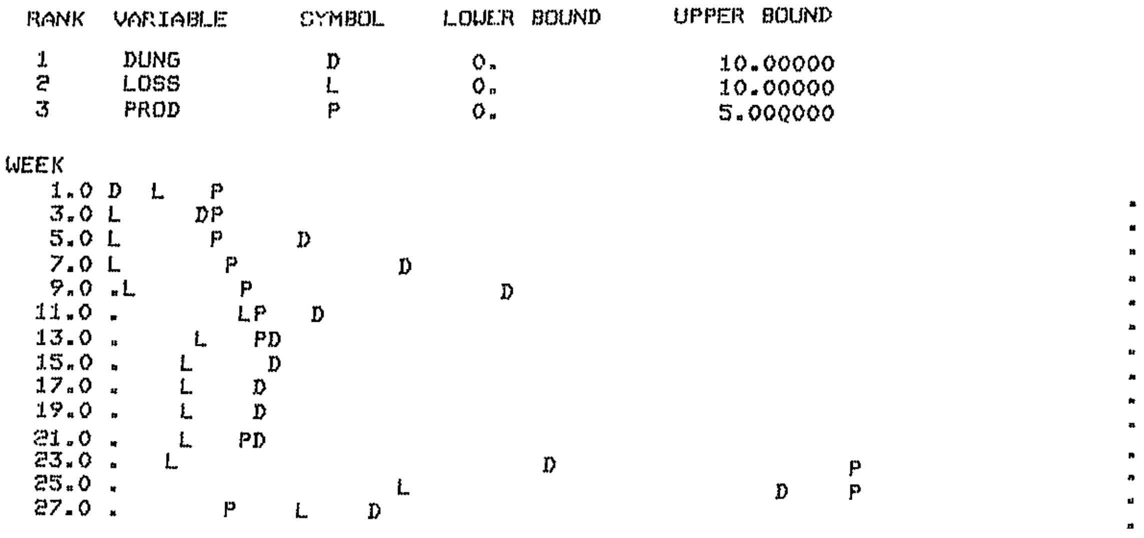
Figure 4. Inspecting and changing arbitrary functions.

The *reference information* includes the values of all the parameters, the initial and present values of all the variables and the values associated with the arbitrary functions. It also includes the names by which the user refers to the parameters and variables and the information telling DRIVER what output to produce when the model is running.

Output from model

When the model is running it changes the values of the variables. The changes in the variables are the behaviour of the model that it is desired to know. DRIVER has facilities allowing a choice of output formats to show these changes. A major distinction between these outputs is between *repetitive* output, occuring during the run, and *final* output, occurring when the run is concluded.

For repetitive output, there are four choices - "none", "table", "graph" and "complete". "None" will mean there is no repetitive output, (final output only being used). "Table" output is simply a table of the values of a selection of the variables, as shown in Figure 2. Only one line of values is written at each one iteration. "Graph" output writes a graph on the terminal, one line per iteration, (thus time runs *down* the page) with the values of a selection of the variables indicated by the positions of user-chosen symbols as shown in Figure 5. The scale of the graph is under user-control and is independent for each of the variables. The value of the first variable in the table output is written at the left of the line. If this is the time it provides a scale for the graph. "Complete" output is described below. Final output consists of a list of the names and values of none, some or all of the variables. The choice of which variables are in the table, graph and final output is under user-control.



```
 RANK  VARIABLE    SYMBOL    LOWER BOUND    UPPER BOUND

  1     DUNG         D         0.              10.00000
  2     LOSS         L         0.              10.00000
  3     PROD         P         0.              5.000000

WEEK
   1.0 D  L   P
   3.0 L     DP                                                          .
   5.0 L      P     D                                                    .
   7.0 L       P          D                                              .
   9.0 .L       P                D                                       .
  11.0 .       LP   D                                                    .
  13.0 .     L  PD                                                       .
  15.0 .    L     D                                                      .
  17.0 .    L    D                                                       .
  19.0 .    L    D                                                       .
  21.0 .    L   PD                                                       .
  23.0 .   L                       D                   P                 .
  25.0 .                  L                        D   P                 .
  27.0 .       P   L    D                                                .
                                                                         .
```

Figure 5. Example graph output.

The maximum number of variables in the table output depends on the terminal's line length - an 80 character terminal allows for 6 variable values. Up to 8 variables may be plotted with the graph output. There is no limit to the number of variables in the final output - the values of all the variables may be written (on a number of lines).

DRIVER has been written for use with a VDU (Visual Display Unit, that is a keyboard linked to a screen on which the user's input and the computer's output appear). The VDU does not have graphics (line drawing) capability and the information displayed on the screen is not permanent. It is desirable, however, to have some form of permanent output from the simulations. DRIVER provides this by writing to a file which can subsequently be printed. The output on this file is basically similar to that on the terminal. Advantage is taken of the printer's greater line width to output more variables in table output and to print the values of the second to fifth variables in the table output list to the right of the graph output. "Complete" output consists of the names and values of all the variables on the printer and standard table output on the terminal.

Complete output is particularly useful when it is desired to trace the operation of the model step by step (usually when a "bug" is suspected). Repetitive output, on both terminal and printer, need not be produced each iteration of the model, but any suitable interval (eg every fourth iteration) may be chosen. This is especially useful for models with a short iteration time (chosen to minimise errors caused by approximating a continuous process with discrete time units). Figure 5 shows graph output produced on every second iteration.

Use of DRIVER

One of the additional benefits of DRIVER is that it makes the writing of the model programme easier, since no detailed input or output instructions are needed. Once the model is written as a subroutine it can be linked to DRIVER. Before this combination can be used it is necessary to define values for all the reference information. An important feature of DRIVER is that copies of the reference information can be preserved (on "disk") from one session to another. This makes it unncessary to re-enter all the parameters and initial values each time - the model can be run using the values left in a file after a previous session. At any time when using DRIVER the reference information can be written out to a file, or a previous set of information retrieved. An auxiliary programme, CHANGE, sets up a first file of reference information and is also used to make the modifications necessary when the model structure is altered, (as by addition of new parameters, variables, or functions). The capability of keeping the reference information is particularly useful with models in which desirable parameter, variable and function values are found by trial and error. A suitable approach in such a case would be to set up a first reference file with guessed values, using CHANGE. DRIVER, with the model, could be then used to alter the values until the model behaved acceptably. The reference information would then be kept on disk, and these values used as the starting point for subsequent work (such as determining the effects of perturbations). The use of CHANGE is described in "Setting up a model for use with DRIVER".

When actually using DRIVER with a model, the usual sequence is - DRIVER asks a question, (known as a prompt) which the user answers with one of a number of command words. DRIVER then either performs the action commanded or asks a further question. The commands and their uses are described in "Meaning and use of DRIVER commands". An example of a user-DRIVER dialogue is shown in Appendix 2. The corresponding printer output is in Appendix 3.

Some features of DRIVER include

- The use of abbreviations for command words. This is an aid in striking the balance between meaningful commands, for the new user, and brief commands for the more experienced.

- Entry of the word "help" to any prompt which expects a command, will produce a list of the possible answers.

- For nearly all prompts the entry of a blank line will cause a return to the previous prompt, as shown by the dashed lines in Figure 6. A number of prompts expect the response to be a variable or parameter name, possibly followed by further information. All such prompts are repeated until a blank name is entered.

- A run of the model can be either an initial run or a "carry on" run for which the variables (including the time) are not reinitialised.

- The command "loop" performs a sequence of runs using a regular series of values for one of the parameters or variables.

- The command "make initial" makes the present values of the variables the initial values for all future runs. With a model that will come to an equilibrium, it would be possible, starting from arbitrary state variable values, to run the model until it was sufficiently near equilibrium and then make that equilibrium state the initial state for subsequent runs. The "carry on" command could be used repeatedly while bringing the model to equilibrium.

- The repetitive output interval and the symbols and scales for the graph output can be changed from the main prompt and from the graph output prompt.

- It is possible to switch from changing graph output to changing table output and *vice versa*. This does not affect which output type is to be used. This is useful as certain elements of the table output are used in the graph output on the printer.

- The output intervals for terminal and printer are independent. If either is set to zero, there is no repetitive or final output on that device. This can be used to suppress printer output when, for example, reasonable parameter values are being searched for and the model output is not of lasting interest, or conversely, to suppress terminal output (and thus speed up "real" execution time) when printer output is all that is desired.

- Whenever a parameter or variable value, or an arbitrary function is changed by the user, DRIVER writes the new value to the printer. The printer output thus contains a complete record of the values before the output from a run. These printings are unaffected by the output intervals.

- Additional information may be written to the printer file. The command "print" writes the values of all parameters, variables and functions. The command "title" allows the entry of any number of lines of annotation from the terminal.

- The command "get" causes a set of reference information to be read in from a disk file. The label line (see next feature) is echoed to the terminal and the printer file. An automatic read is carried out at the beginning of a session.

- The command "put" causes the reference information to be written to disk. The user supplies a label line, also written to the disk, which can be used to draw attention to special features of that particular set of reference information. The user also defines the name by which the file is to be known. The name, label and values of parameters, variables and functions are written on the printer file.

MEANINGS AND USE OF *DRIVER* COMMANDS

Introduction

There are six "prompts" (that is questions displayed on the terminal by the computer), which require "commands" as the user's responses. The commands are usually words, but a few commands are single characters. In addition to the prompts requiring commands there are a number of prompts requiring parameter or variable names or numbers as the response. This section describes the use of all the commands.

If "help" is entered as a prompt, DRIVER lists all the appropriate commands and re-prompts. Entry of an invalid command also results in DRIVER re-prompting.

Any command consisting of one word can be abbreviated to its first two letters. The two word commands can be abbreviated to the first two letters of the first word, and the first letter of the second word. There must be a space between the words of a two word command.

All of the prompts which require a variable or parameter name are "repeating prompts". After entry of the name and any associated information DRIVER repeats the prompt, waiting for another name. This continues until the user enters a blank line instead of a name. DRIVER then returns to the previous prompt (ie the command prompt that DRIVER came from to the repeating prompt).

Most of the other prompts are self-explanatory, and further details are only given below when necessary. Where two commands are listed, separated by a comma, the commands are synonyms.

Items in angled brackets, thus <item>, are descriptions of the item that should be entered, and are not entered as written.

Unless otherwise stated, after DRIVER has obeyed a command, it returns to the prompt it came from. The relationships between the prompts are shown in Figure 6. Transfer caused by entry of a command follows the solid arrows and transfer caused by entry of a blank line is to either the main prompt, directly, or to a prompt at the intermediate level (broken arrows).

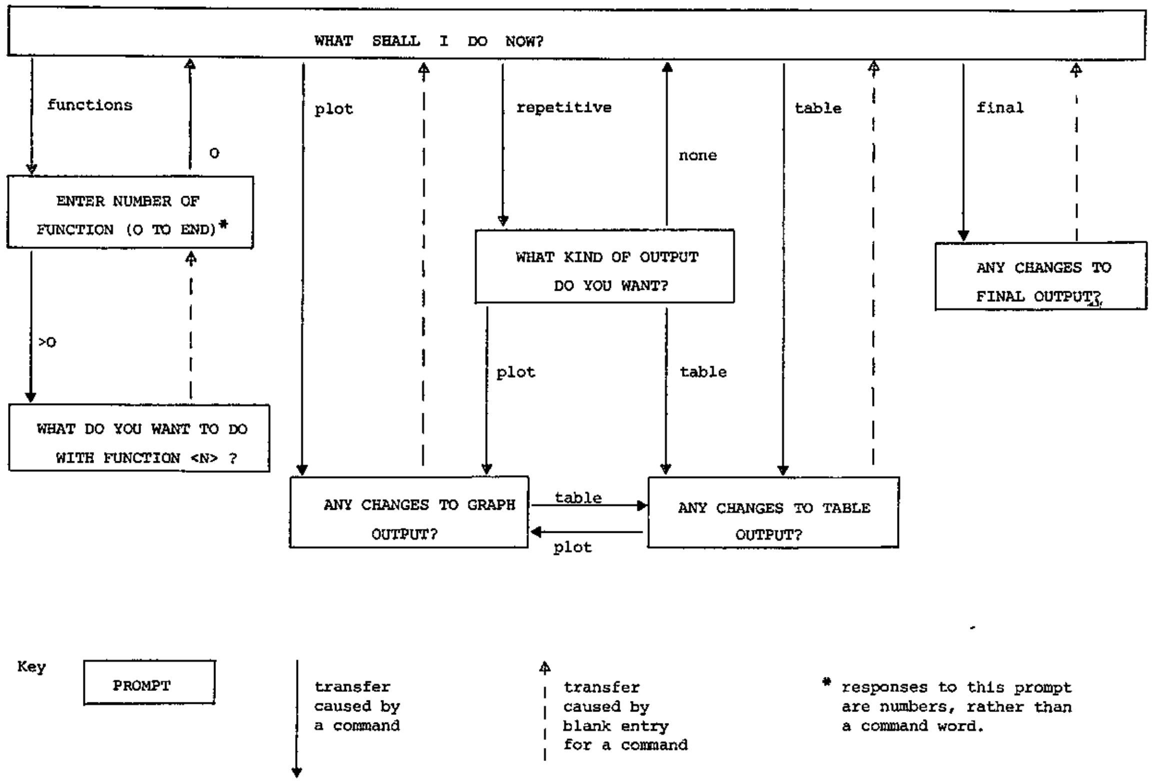

WHAT SHALL I DO NOW?
functions
0
ENTER NUMBER OF FUNCTION (0 TO END)*
>0
WHAT DO YOU WANT TO DO WITH FUNCTION <N> ?
plot
repetitive
none
WHAT KIND OF OUTPUT DO YOU WANT?
plot
table
table
final
ANY CHANGES TO FINAL OUTPUT?
ANY CHANGES TO GRAPH OUTPUT?
table
plot
ANY CHANGES TO TABLE OUTPUT?
Key
PROMPT
transfer caused by a command
transfer caused by blank entry for a command
* responses to this prompt are numbers, rather than a command word.

Figure 6. Relationships between the major prompts.

Main prompt

Main prompt : "WHAT SHALL I DO NOW?"

Response	Use/meaning
go	Set all variables to their initial values and run the model.
carry on	Run the model, without resetting variables.
show	To inspect the values associated with parameters and variables. Repeating prompt : "WHICH VARIABLE OR PARAMETER DO YOU WANT TO KNOW ABOUT?". For parameters DRIVER shows the value, for variables the present and initial values and the graph symbol and bounds.
show parameters	Display the names and values of all the parameters.
show variables	Display the names, initial and present values of all the variables.
set	To set the value of any parameter or the initial value of any variable. Repeating prompt : "ENTER PARAMETER OR VARIABLE NAME AND NEW VALUE".

set present
To set present value of any variable. Repeating prompt : "ENTER VARIABLE AND NEW PRESENT VALUE".

make initial
Set the initial values of all variables to their present values.

functions
To inspect or change the arbitrary functions. See section below on Functions.

loop
To perform a number of model runs, changing the value of one of the parameters or the initial value of a variable each time. Effectively performs a FORTRAN "DO-loop", using a parameter or variable as the index. Prompt : "ENTER PARAMETER OR VARIABLE TO BE CHANGED, THE FIRST VALUE, THE FINAL VALUE AND THE INCREMENT". After performing the runs the original value of the index is restored.

repetitive
To change the repetitive output. See section below on Repetitive output.

final
To change the final output. Switch on the final output, if switched off. See section below on Final output.

intervals
To change the output intervals. Prompt : "ENTER OUTPUT INTERVALS FOR TERMINAL AND PRINTER".

bounds
To change the graph bounds (scale) for any variable, whether or not it is in the graph output list. Repeating prompt : "ENTER VARIABLE, NEW LOWER BOUND, NEW UPPER BOUND".

symbols
To change the graph symbol for any variable. Repeating prompt : "ENTER VARIABLE AND NEW SYMBOL".

title
To write annotation to the printer. Any number of lines of information may be entered, which are copied to the printer. Prompt : "ENTER TITLING INFORMATION. COLUMN 1 WILL BE USED FOR CARRIAGE CONTROL. TO END TITLE, ENTER \ IN COLUMN 1".

print
Write, on the printer, the names and values of all parameters, variables and functions.

put
Write out the present reference information as a new reference file. Prompt : "ENTER A LABEL LINE". The line of information entered is copied to the reference file. Prompt : "WHAT NAME IS THIS TO BE SAVED UNDER?". The file name entered is used, overwriting a previous reference file if the name is that of an old file. DRIVER writes, on the printer "THIS SAVED AS FILE <filename>, the label line and the same output as is generated by the command "print".

get	Read in a reference file. DRIVER writes the name and the label on the terminal and the printer.
end, stop	Has the same effect as put, but execution stops afterwards.
kill	Cease execution.
none graph, plot table complete	All produce exactly the same result as they do when given as responses to the repetitive output prompt "WHAT KIND OF OUTPUT DO YOU WANT?". See section below on Repetitive output.
show repetititive	Produces same result as entry "show" to prompt "WHAT KIND OF OUTPUT DO YOU WANT?". See section below on Repetitive output.

Functions

Number prompt : "ENTER NUMBER OF FUNCTION (0 TO END)". If response is 0 (zero), DRIVER will return to the main prompt. Otherwise the present values of the chosen function are displayed, followed by the prompt : "WHAT DO YOU WANT TO DO WITH FUNCTION <number>?".

Response	Use/meaning
<blank>	No change.
all	To change all the values, including the number of coordinate pairs. Prompt : "ENTER NUMBER OF CO-ORDINATE PAIRS (IF A STEP FUNCTION -VE). If the function is to be an interpolating function, the new number of coordinate pairs is entered. If it is to be a step function, minus that number. DRIVER then proceeds as if the command "coordinates" had been used.
coordinates, xy	To change the x and y coordinates, without changing the number of coordinate pairs. Prompt : "ENTER X COORDINATES". They should be entered, all on one line. Prompt : "ENTER Y COORDINATES". Also all on one line.
x	To change only the x coordinates. Prompt : "ENTER THE X COORDINATES".
y	To change only the y coordinates. Prompt : "ENTER THE Y COORDINATES".
null	To make the function return the same value, regardless of the value of "x". Prompt : "ENTER CONSTANT VALUE".

test	To determine what value will be returned by the FF function for a particular value of "x". Prompt : "ENTER TEST VALUE". DRIVER displays : FF (<test value>, <number>) = <returned "y" value>
step	Make the function a step function, without altering the coordinates.
interpolating	Make the function an interpolating function, without altering the coordinates.

After any of these commands are completed DRIVER returns to the number prompt, with the exception that after "test" DRIVER returns to "WHAT DO YOU WANT TO DO WITH FUNCTION <number>?".

Repetitive output

Prompt : "WHAT KIND OF OUTPUT DO YOU WANT?".

Response	Use/meaning
none	Set no repetitive output, return to main prompt.
graph, plot	Set graph repetitive output, go to graph prompt (see below).
table	Set table repetitive output, go to table prompt (see below).
complete, all	Set complete output (all variables on printer, as table on terminal). Go to table prompt.
show	Show which output type is set. If graph or table is set, DRIVER performs the display that would occur if "show" was entered to the graph or table prompt. After displaying the output set, DRIVER proceeds to the prompt that it would go to if that output type had just been set.

Graph prompt : "ANY CHANGES TO GRAPH OUTPUT?"

Response	Use/meaning
<blank>	No further changes, return to main prompt.
show	Display names of variables which will be plotted, with associated symbols and graph bounds, and the output intervals.
+, add	See section below on Changing output lists.
-, subtract	
empty	
swop	

bounds symbols intervals	Identical to same commands when used as responses to main prompt.
table	Go to table prompt, without changing output type selected.

Table prompt : "ANY CHANGES TO TABLE OUTPUT?"

Response	Use/meaning
<blank>	No further changes, return to main prompt.
show	Display names of variables in the table output list and the output intervals.
+, add -, subtract empty swop	See section below on Changing output lists.
intervals	As for main prompt.
graph, plot	Go to graph prompt, without changing output type selected.

Final output

Prompt : "ANY CHANGES TO FINAL OUTPUT?".

Response	Use/meaning
<blank>	No further changes, return to main prompt.
show	Display names of variables in final output list.
+, add -, subtract empty swop	See section below on Changing output lists.
all	Replace present list by list of all the variables.
off	Turn final output off, but without changing the list. Return to main prompt. Final output is automatically turned on by the response "final" to the main prompt.

Changing output lists

Graph, table and final output all work with lists of variables, and the same commands are used for modifying all three of these lists. Several of the commands invoke repeating prompts.

Response	Use/meaning
+, add	To add variables at the end of the list. Repeating prompt : "WHICH VARIABLE?".
-, subtract	To remove variables from the list. Repeating prompt : "WHICH VARIABLE".
empty	Remove all variables from the list.
swop	To exchange positions of two variables in the list. Repeating prompt : "WHICH TWO VARIABLES DO YOU WANT TO SWOP". The first must already be in the list.

SETTING UP A MODEL FOR USE WITH *DRIVER*

Introduction

This guide deals with two aspects of setting up a model for use with DRIVER - writing the programme and creating the first copy of the reference file. Writing the programme will always involve some system dependent features, and this guide only deals with the standard elements. Creating the reference file, using the CHANGE programme, is more or less standard.

Writing the programme

The model must be programmed as a FORTRAN subroutine, with the name MODEL, and no F-parameters[1]. An example is given in Appendix 1. The D-parameters and D-variables are passed to the subroutine in COMMON in a particular, user-defined sequence. Details are system-dependent. On entry to the subroutine, from DRIVER, all the D-parameters and D-variables will have defined values. However, some of the "other" D-variables may have inappropriate values, since they may be such things as the ratio of some of the state variables, functions of time and so forth.

There are various elements of the model subroutine that DRIVER requires. These are shown in Figure 7. The first required executable statement is -

```
      CALL OUT1 (0)
```

[1] To avoid confusion, parameters and variables of the model (ie DRIVER terminology) will be distinguished from the FORTRAN terminology by the terms D-parameters and D-variables for the DRIVER ones, and F-parameters and F-variables for the FORTRAN ones.

The subroutine OUT1 handles repetitive output and has one integer F-parameter. If this is zero, as here, then the appropriate output is performed, regardless of the output interval. The use with a non-zero parameter is explained below. (If an output interval is zero (see "Purpose and capabilities of the DRIVER programme") there is no output on that device). Thus at the beginning of a run there is a line of the chosen repetitive output. Since some of the "other" variables may have erroneous values it may be desirable to calculate them, from the state variables, before the call to OUT1.

```
      SUBROUTINE MODEL

      COMMON <declaration for parameters, including RUNTIM>

      COMMON <declaration for variables, including TIME>

           (Output variables may be calculated here
           from the state variables)


      CALL OUT1 (0)

      LIMTIM = RUNTIM

      DO 100 ITIME = 1,LIMTIM

      TIME = INT (TIME + 1)

           (Main part of model)


      CALL OUT1 (<integer expression>)

  100 CONTINUE

      CALL OUT2

      RETURN

      END
```

Figure 7. Elements required in the MODEL subroutine.

The next three required statements concern the way in which simulation time is handled in DRIVER models. As explained in "Purpose and capabilities of the DRIVER programme", the length of a model run is defined by the "runtime". This is passed into the model routine as one of the D-parameters (usually called RUNTIM). This is a real F-variable, as are all the D-parameters and D-variables. Similarly, one of the D-variables is the time (shown as TIME in Figure 7, but can have any legal name). TIME is effectively a state variable, in that it has a meaningful initial value. It is incremented by one each iteration. Since RUNTIM and TIME are real they cannot be used in a DO-loop and the three lines -

```
     LIMTIM = RUNTIM

     DO 100 ITIME = 1, LIMTIM

     TIME = INT (TIME + 1)
```

form the head of the iteration loop. TIME is set to INT(TIME + 1) so as to retain a whole number value, and avoid any rounding error. It is not set to ITIME because it may start at any value, as in a "carry on" run. ITIME is used only as the DO-loop index.

After these three lines the main part of the model follows. In this all the D-variable values should be redefined. Arbitrary functions may be used in the form of the function call -

```
     FF (<real expression>, <integer expression>)
```

The <real expression> is the current value of "x", the independent variable. It is important to be clear what it is, and what units it and the returned "y" values are in. The <integer expression> (usually an integer constant), indicates which of the arbitrary functions is to be used. Again the programmer must keep clear which number is used when. Taking the model as a whole there should be no gaps in the sequence of function numbers (otherwise a dummy, unused, function must be included when using CHANGE).

After the main part of the model, which consists of the calculations for only one iteration, the model routine ends with the five compulsory lines -

```
         CALL OUT1 (<integer expression>)

     100 CONTINUE

         CALL OUT2

         RETURN

         END
```

This second call to OUT1 involves the use of the output "interval" feature. In its simplest form, output is only written on a device (terminal or printer) when the D-variable TIME is an exact multiple of the output interval associated with that device. In this case TIME should be placed first in the COMMON declaration of the D-variables, and the <integer expression> will be the constant 1. However in some models it is desirable to alter which D-variable is used in the interval feature. An example might be a model with weekly iterations, run for some years. If the output interval is not a factor of 52 then output for different years will not be comparable. It will be found to be convenient to have a second time D-variable which cycles from 1 to 52 over the year. The output interval can then be made to depend on this - the <integer expression> then indicating the position of the variable in the COMMON declaration of the D-variables. A further variation is to have a D-parameter which is used in the <integer expression> to indicate which D-variable is to be used for the interval calculations. Since these calculations perform integer arithmetic to determine if the values are exact multiples it is important that the D-variables used only take whole number values. The statement numbered 100 is the end of the iteration loop of the model. Subroutine OUT2 handles the final output.

Creating and modifying the reference file

The programme CHANGE is used to set up and alter reference files. Although the reference files can be altered by DRIVER, DRIVER cannot change the names of the D-parameters and D-variables, nor the number of D-parameters, D-variables or arbitrary functions.

CHANGE works with the lists of D-parameters, D-variables and arbitrary functions, dealing with each in a similar manner. When the reference file is completed, the order of D-parameters and D-variables must correspond exactly with this sequence in the COMMON declarations in the model subroutine. This is necessary because DRIVER considers the D-parameter and D-variables as a single vector, each element corresponding to an element in the appropriate vector of names. In the model routine the D-parameter and D-variable will be mostly simple F-variables, but may include arrays. The correspondence between these in the model and in DRIVER is achieved by the sequence in the COMMON declaration in the model. If multidimensional arrays are included in the model it is essential to understand the sequence in which such arrays are stored by the compiler. The names that the user gives to the D-parameter and D-variable need not be the name as the FORTRAN names in the model. D-parameters and D-variables that are elements of arrays in the model are not distinguished in DRIVER. It may be found helpful to give them names which include their index (eg AGE 1, AGE 2, etc). The position of the arbitrary functions in their list must also correspond to the numbers used in the calls to FF.

Since the method of use when altering an old file is part of the use in creating a file from scratch, the use in altering will be described first. With the parameters the names and values of all the defined D-parameters are displayed on the terminal. The user is then asked to enter the index (position in the list) of the next D-parameter. If the user enters a number corresponding to a defined parameter, he is asked to enter the name and value of it. These will overwrite the previous name and value. If a blank is entered for the name, the parameter will be deleted, (though the parameter list is not yet altered otherwise). If the user enters a negative number as the index, all the parameters with indexes from the absolute value of the entry and higher are moved up one and the new entry inserted. If the index entry is zero, and alterations have been made since displaying the parameter list, CHANGE removes deleted items, and then displays the parameter list as it now is. If zero is entered as index as the first entry after the list is displayed, the parameters are considered completed and CHANGE moves on to the variables, which are treated in the same way as the parameters, except the graph bounds and symbol are required, in addition to the initial value. On completing the variables, the functions are dealt with. The functions are not all listed, but the user is informed how many are defined. If the index entered corresponds to a defined function, its values are displayed and a response asked for. The responses are similar to the commands used in the function altering part of DRIVER, but are numeric. The meanings can be found by entering a negative number, which will cause a key to be displayed.

When creating a new reference file there will initially be no lists of parameters, variables, etc defined. Consequently CHANGE automatically increments the index and asks for the parameter or variable name. This will continue until a blank name is entered. Thereafter CHANGE reverts to the procedure described above, allowing the list to be corrected. A similar process is followed with the functions, except that the change from automatic indexing to the correction process is indicated by the entry for the number of coordinate pairs of -1.

When CHANGE starts the user is asked if he is creating a new file or updating an old one. This refers to whether he wishes to start from scratch or not, and bears no relation to whether the result will eventually overwrite the old file. Reading in an old file and writing the file out at the end of the run are carried out in exactly the same way as in DRIVER, and are partly system-dependent. Although CHANGE sets values for the output of DRIVER these will nearly always be inappropriate, and should be changed from DRIVER.

APPENDIX 1. LISTING OF EXAMPLE MODEL (*DUNG 1*)

```
              $SET SEPARATE
                    SUBROUTINE MODEL
              C-          THE PARAMETERS AND VARIABLES ARE DECLARED IN COMMON
                    REAL LOSS,LOSRAT
D-parameters        COMMON /PARAM/ RUNTIM,WKSIN,DENSMT,WILRAT,CATRAT,WEEKIN(10),WHICH
D-variables         COMMON /VARIA/ TIME,WEEK,WILD,SMITS,OTHCAT,CATTLE,WILDUN,CATUNG,
                   *PROD,LOSRAT,LOSS,DUNG
              C-           TIME INCREASES CONTINUOUSLY,WEEK CYCLES,MODULO 52
                    WEEK=MOD(INT(TIME),52)+1
repetitive    C-           THIS CALL PROVIDES FOR OUTPUT OF THE STARTING STATE
output               CALL OUT1(0)
              C-            THE NEXT 4 LINES ARE THE START OF THE MAIN LOOP
                     LIMTIM=RUNTIM
                     DO 100 ITIME=1,LIMTIM
                     TIME=INT(TIME+1)
                     WEEK=MOD(INT(TIME),52)+1
              C-          THE BIOMASS DENSITY OF WILD UNGULATES IS AN ARBITRARY
use of an     C-           FUNCTION OF THE TIME OF YEAR.
arbitrary            WILD=FF(WEEK,1)
function      C-            FROM HERE TO STATEMENT NUMBER 3 WE ARE DETERMINING
              C-            WHETHER THE ITINERANT (SMIT'S) CATTLE ARE PRESENT THIS WEEK
                     NWKS=WKSIN
                     IF(NWKS.EQ.0) GO TO 4
                     DO 1 I=1,NWKS
                     IF(WEEK.EQ.WEEKIN(I)) GO TO 3
                   1 CONTINUE
                   4 SMITS=0
                     GO TO 2
              C-           IF THE CATTLE ARE HERE,THE VARIABLE SMITS TAKES THE VALUE OF
              C-          THE PARAMETER DENSMT
                   3 SMITS=DENSMT
              C-           THE DENSITY OF THE 'OTHER' (RESIDENT) CATTLE IS AN ARBITRARY
              C-            FUNCTION OF THE TIME OF YEAR
                   2 OTHCAT=FF(WEEK,2)
                     CATTLE=OTHCAT+SMITS
              C-            THERE ARE DIFFERENT PRODUCTION RATES (MASS/BIOMASS) FOR
              C-            CATTLE AND BUCK
                     WILDUN=WILD*WILRAT
                     CATUNG=CATTLE*CATRAT
                     PROD=WILDUN+CATUNG
              C-           THE LOSS RATE (PROPORTION GOING) IS FUNCTION OF TIME OF YEAR
                     LOSS=DUNG*LOSRAT
                     DUNG=DUNG+PROD-LOSS
repetitive    C-             THIS IS THE CALL TO THE REPETITIVE OUTPUT ROUTINE.
output               CALL OUT1(INT(WHICH))
                 100 CONTINUE
final                CALL OUT2
output               RETURN
                     END
#
```

APPENDIX 2. EXAMPLE DIALOGUE

```
     WHAT SHALL I DO NOW?
 *INT
  ENTER OUTPUT INTERVALS FOR TERMINAL AND PRINTER (SAME LINE)
   IF NO OUTPUT WANTED ON A DEVICE,ENTER 0 FOR THAT INTERVAL
 *1,4

     WHAT SHALL I DO NOW?
 *GO


  RANK     VARIABLE    SYMBOL   LOWER BOUND     UPPER BOUND

    1     DUNG          D         0.               10.00000
    2     LOSS          L         0.               10.00000
    3     PROD          P         0.               5.000000


  WEEK
     1.0 D  L   P                                                             .
     2.0 L  D  P                                                              .
     3.0 L     DP                                                             .
     4.0 L      P  D                                                          .
     5.0 L      P     D                                                       .
     6.0 L       P       D                                                    .
     7.0 L       P           D                                                .
     8.0 .L      P              D                                             .
     9.0 .L       P                 D                                         .
    10.0 .        P   L    D                                                  .
    11.0 .        LP   D                                                      .
    12.0 .      L  P D                                                        .
    13.0 .     L   PD                                                         .
    14.0 .    L     D                                                         .
    15.0 .    L     D                                                         .
    16.0 .    L    PD                                                         .
    17.0 .    L    D                                                          .
    18.0 .    L    D                                                          .
    19.0 .    L    D                                                          .
    20.0 .    L    D                                                          .
    21.0 .    L   PD                                                          .
    22.0 .    L   D                                                           .
    23.0 .   L                         D                    P                 .
    24.0 .              L                         D         P                 .
    25.0 .                   L                         D    P                 .
    26.0 .        P             L   D                                         .
    27.0 .       P    L    D                                                  .
    28.0 .       PL   D                                                       .
    29.0 .     L P D                                                          .
    30.0 .    L  PD                                                           .
    31.0 .   L                        D                   P                   .
    32.0 .      P      L   D                                                  .
    33.0 .      P L   D                                                       .
    34.0 .     LP  D                                                          .
    35.0 L      P     D                                                       .
    36.0 L      P        D                                                    .
    37.0 L      P           D                                                 .
    38.0 L     P              D                                               .
    39.0 .L                                          D   P                    .
    40.0 . L   P                                     D                        .
    41.0 . L   P                                      D                       .
    42.0 . L   P                                       D                      .
    43.0 . L   P                                        D                     .
    44.0 . L   P                                         D                    .
    45.0 . L   P                                         D                    .
    46.0 . L   P                                          D                   .
    47.0 . L   P                                           D                  .
    48.0 . L   P                                           D                  .
    49.0 . L   P                                            D                 .
    50.0 . L   P                                            D                 .
    51.0 . L   P                                             D                .
    52.0 . L   P                                             D                .
     1.0 . L   P                                              D               .
TIME        52.0000       WEEK        1.00000       WILD       1.00000
MOVE       0.             STACTL     0.250000       TOTCTL    0.250000
WLDDNG     0.350000       CTLDNG     0.100000       PROD      0.450000
LOSRAT      .500000E-01   LOSS       0.384118       DUNG       7.74823
```

APPENDIX 2. (CONTINUED)

```
      WHAT SHALL I DO NOW?
*  NONE
   NO REPETITIVE OUTPUT.

      WHAT SHALL I DO NOW?
*  FINAL
   ANY CHANGES TO FINAL OUTPUT ?
*  EMPTY                                                     removing all variables
   FINAL OUTPUT LIST EMPTIED.                                from the final output
   ANY CHANGES TO FINAL OUTPUT ?
*  +
   WHICH VARIABLE?                                           adding dung to the (empty)
*  DUNG                                                      output list
   WHICH VARIABLE?
*
   ANY CHANGES TO FINAL OUTPUT ?                             blank line ends repeating prompt
*

      WHAT SHALL I DO NOW?
*  LOOP
   ENTER THE PARAMETER OR VARIABLE TO BE CHANGED,THE FIRST VALUE,THE LAST VALUE
   AND THE INCREMENT.
*  CTLRAT
*  0
*  1
*  .25

        CTLRAT =  0.
     DUNG          4.81907

        CTLRAT =  0.25000
     DUNG          6.64980

        CTLRAT =  0.50000
     DUNG          8.48053

        CTLRAT =  0.75000
     DUNG          10.3113

        CTLRAT =   1.0000
     DUNG          12.1420
                                                             parameter used as index of
        CTLRAT =     0.400000                                loop restored to its old value


    WHAT SHALL I DO NOW?
 TITLE
 ENTER TITLING INFORMATION. COLUMN 1 WILL BE USED FOR CARRIAGE CONTROL.
 TO END TITLE ENTER & IN COLUMN 1.
   TEXT ENTERED IN THIS MANNER WIILL APPEAR ON THE PRINTER OUTPUT
 &
   1 LINES WRITTEN.

    WHAT SHALL I DO NOW?
```

APPENDIX 3. EXAMPLE OUTPUT CORRESPONDING TO DIALOGUE

```
                                               INITIAL RUN

RANK     VARIABLE    SYMBOL   LOWER BOUND      UPPER BOUND

  1       DUNG         D            0.000000         10.00000
  2       LOSS         L            0.000000         10.00000
  3       PROD         P            0.000000         5.000000

 WEEK                                                                                TOTCTL      WILD        DUNG        WLDDNG
     1.000 D  L   P                                                              .        0.000        1.21       0.000        .423
     5.000 L      P     D                                                        .         .250        1.31        1.93        .458
     9.000 .L       P                 D                                          .         .250        1.62        3.89        .565
     13.00 .     L  PD                                                           .         .250        1.92        1.66        .673
     17.00 .    L    D                                                           .         .250        1.89        1.55        .661
     21.00 .    L   PD                                                           .         .250        1.74        1.44        .609
     25.00 .                   L                         D    P                  .         7.75        1.59        6.59        .557
     29.00 .     L P D                                                           .         .250        1.44        1.57        .506
     33.00 .      P L   D                                                        .         .250        1.30        1.90        .454
     37.00 L      P           D                                                  .         .250        1.15        2.75        .402
     41.00 . L   P                                       D                       .         .250        1.00        6.68        .350
     45.00 . L   P                                          D                    .         .250        1.00        7.11        .350
     49.00 . L   P                                             D                 .         .250        1.00        7.46        .350
     1.000 . L   P                                               D               .         .250        1.00        7.75        .350
       FINAL OUTPUT

TIME            52.0000    WEEK           1.00000   WILD             1.00000   MOVE             0.000000   STACTL           .250000
TOTCTL          .250000    WLDDNG         .350000   CTLDNG      1.000000E-01   PROD              .450000   LOSRAT      .500000E-01
LOSS            .384117    DUNG           7.74823

  CTLRAT   =      0.000000                        INITIAL RUN

DUNG            4.81905

  CTLRAT   =       .250000                        INITIAL RUN

DUNG            6.64979

  CTLRAT   =       .500000                        INITIAL RUN

DUNG            8.48051

  CTLRAT   =       .750000                        INITIAL RUN

DUNG            10.3112

  CTLRAT   =       1.00000                        INITIAL RUN

DUNG            12.1420

  CTLRAT   =      .40000
TEXT ENTERED IN THIS MANNER WILL APPEAR ON THE PRINTER OUTPUT
```