INTERNAL REPORT 126

FLEX 1 USER'S MANUAL

W. Scott Overton, Jon A. Colby, Jonna Gourley
and Curtis White

Oregon State University

January 1973

Revised, September 1973

The FLEX1 Users Manual

ABSTRACT

FLEX1 is a general model processor, patterned after Klir's General Sequential System Paradigm (Figure 1). Specifically, the processor is a discrete time flux oriented realization of the general paradigm and can process non-linear, non-stationary, environmentally controlled state variable system models with explicit memory. The current version is restricted to not more than 63 state variables, 40 input variables, 200 memory variables, and 50 output variables. There are also restrictions on the number of functions identified, and other restrictions, which will be elaborated in the appropriate part of this writeup.

The system is implemented in the Oregon State University OS-3 operating system, is specific for the CDC 3300 computer, and may be operated only from a teletype.

Output may be monitored at the teletype, line printed and dump filed. Satellite programs have been written for plotting and listing from the dump file.

FLEX1 is designed to be expanded in two directions. First will be further elaboration as a single level model processor concentrating all capacity into this single level. Second will be elaboration into a multi-level, hierarchical model processor (REFLEX), capable of mixed resolution. The two capacities are being incorporated into a single processor, FLEX2, which is in the final stage of development at this printing, and will be operative in late 1973.

Table of Contents

Table of Contents
Page 2 of 2

LIST OF FIGURES

GLOSSARY OF SYMBOLS

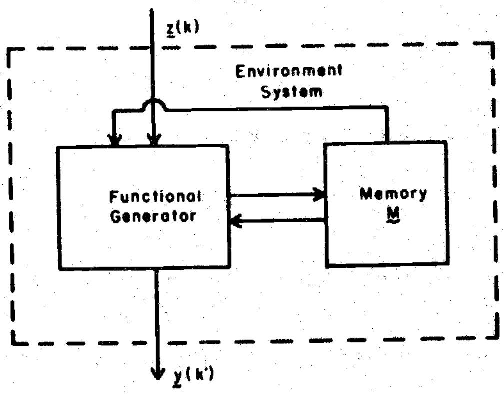
$X, Y, \ldots$	Sets
$x, y, \ldots$	Elements of Sets
$x \in X$	x is an element of X
$\underset{\sim}{X}, \underset{\sim}{Y}, \ldots$	Matrices
x_{ij}	Components of Matrices
$x_{ij} \in \underset{\sim}{X}$	x_{ij} is the component in the ith row and the jth column of $\underset{\sim}{X}$
$\underset{\sim}{x}, \underset{\sim}{y}, \ldots$	Vectors
x_i	Components of Vectors
$x_i \in \underset{\sim}{x}$	x_i is the ith component of $\underset{\sim}{x}$
k	Discrete time index
$X(k), x(k), \ldots, \underset{\sim}{x}(k), x_i(k)$	Value of that Set, Element, . . . , Vector, or Vector Component at time k
lun	Logical unit number
<lun>	Replace with an actual lun (i.e. 1 to 50)
name	A file name (unless specified differently)
<name>	Replace with an actual file name
i, j	Index numbers
<i>	Replace with a particular index number
>	FLEX1 command mode symbol - glitch (unless used in context of "greater than").

I. THE FLEX PARADIGM

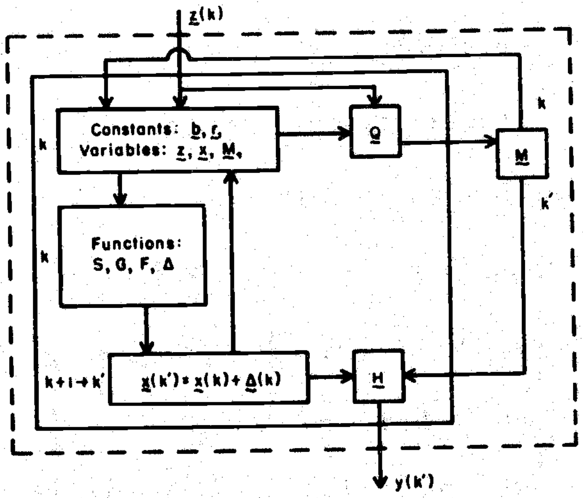
Introduction

FLEX is a realization of Klir's general paradigm which is illustrated in Figure 4.8 of An Approach to General Systems Theory (Klir, 1969). This figure has been modified as Figure 1 of this writeup. In constructing the processor, it was desirable to restrict the paradigm to discrete time, versus continuous time, and to orient it to a flux system. The flux orientation follows the conceptual structure of compartment models, which are currently fashionable in ecology. However, some forms of ecological models are awkward in this form, and the section on modelling (Section II) includes a discussion of alternative orientations possible under the current version. It is also proving difficult to model some systems conceptualized in continuous time in the form of discrete time and it is anticipated that a differential analyzer will be made available (in place of the discrete analyzer) in a future version. For the present version, however, the paradigm is strictly in discrete time, and restricted to a difference equation structure. The f functions must explicitly or implicitly define the first differences of the state variables.

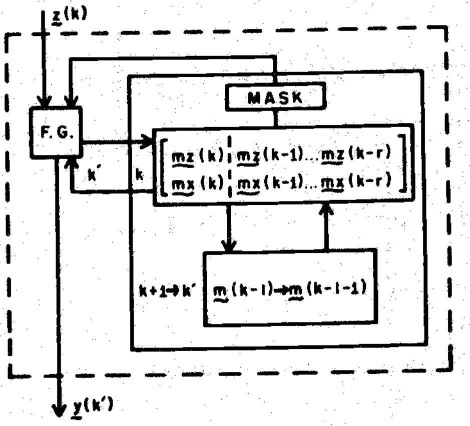
Klir's model structures have also been restricted somewhat to accommodate currently fashionable terminology. The terminology of FLEX is a mixture of Klir and Freeman (1965). Klir's principal quantities include input, output, and memory variables as used in FLEX,



a. The General Paradigm as modified



b. Elaboration of the Functional Generator



c. Elaboration of the Memory

Figure 1. The FLEX Paradigm

as well as any others that are necessary to describe the behavior of the system. Klir does not use the term state variable in his general paradigm. Freeman's "state variables" include both state and memory variables in FLEX terminology.

The arguments for the FLEX variable set are all oriented toward the problems of modelling. We feel that identification of variables according to their model function is helpful in model conceptualization. Input variables are those provided by another system or by the environment. State variables are those required to characterize the behavior of the system and are restricted to quantities occurring at a particular instant. These will be incremented by the processor at each time step. Memory variables are past values of input or state variables. Output variables are identified state variables, or functions of state variables, which are designated as system outputs, either for the purpose of study of the system, or as outputs to another system. The vectors $\underset{\sim}{z}$, $\underset{\sim}{x}$, and $\underset{\sim}{y}$ designate the input, state and output variables, respectively, and other quantities are identified in the following discussion.

The Processor Algorithm

The algorithm is the standard discrete time algorithm, structured to explicitly identify the state variable increments Δ and with provision for automatic identity of Δ from a matrix of fluxes (f equations) representing the flows of a compartment model. Letting k be the discrete time index,

$$\left.\begin{aligned} \underset{\sim}{y}(k) &= \underset{\sim}{h}(\underset{\sim}{x}(k)) \\ \underset{\sim}{x}(k+1) &= \underset{\sim}{x}(k) + \underset{\sim}{\Delta}(k) \end{aligned}\right\} \text{Model algorithm}$$

where $\underset{\sim}{\Delta}(k)$ is a column vector such that the ℓth element is defined as

$$\Delta_\ell(k) = \sum_{i=1}^{n} f_{i\ell}(k) - \sum_{\substack{j=1 \\ j\neq\ell}}^{n} f_{\ell j}(k)$$

and where,

$$f_{ij}(k) = f_{i,j}[k, \underset{\sim}{x}(k), \underset{\sim}{z}(k), \underset{\sim}{M}(k), \underset{\sim}{g}(k), \underset{\sim}{s}, \underset{\sim}{b}, \underset{\sim}{r}].$$

Further, $\underset{\sim}{h}$ is an arbitrary functional vector of the vector $\underset{\sim}{x}$ and $\underset{\sim}{M}(k)$ is defined as a matrix of past values of $\underset{\sim}{x}$ and $\underset{\sim}{z}$. That is, let $\underset{\sim}{M}(k) = [\underset{\sim}{m}(k), \underset{\sim}{m}(k-1), \ldots, \underset{\sim}{m}(k-r)]$

$$\text{where } \underset{\sim}{m}(k-\ell) = \underset{\sim}{Q}\begin{bmatrix} \underset{\sim}{z}(k-\ell-1) \\ \\ \underset{\sim}{x}(k-\ell-1) \end{bmatrix} = \begin{bmatrix} \underset{\sim}{m}\underset{\sim}{z}(k-\ell) \\ \\ \underset{\sim}{m}\underset{\sim}{x}(k-\ell) \end{bmatrix}$$

and where $\underset{\sim}{Q}$ is formed by deleting rows from the identity matrix of the vector $\begin{bmatrix} \underset{\sim}{z} \\ \underset{\sim}{x} \end{bmatrix}$, the appropriate row being deleted for each variable which will not be retained in memory. The $\underset{\sim}{Q}$ specification is implicit. The user specifies which elements of $\underset{\sim}{x}$ and $\underset{\sim}{z}$ will be retained and how long each should be retained.

Other quantities identified in the model are:

$\underset{\sim}{b}$, $\underset{\sim}{r}$,	the vectors of model parameters which may change from run to run.
$\underset{\approx}{g}(k)$,	a vector of intermediate functions defined in the same general form as the f_{ij} and such that element $g_i(k)$ may be a function of $g_j(k)$ for $j < i$. G values are calculated once and stored for the duration of the computation cycle.
$\underset{\sim}{s}$,	special functions, similar to g functions, which are available on call and for which the values are not stored by the processor.

II. MODELLING WITH FLEX1

Introduction

In this section discussion will be directed toward identifying how modelling concepts are elaborated in FLEX1 with respect to the general FLEX algorithm. First, a few of the advantages of modelling in a general algorithm will be presented. Next is a discussion of the FLEX1 structure and its relation to the algorithm. Third, the specification of a model in FLEX1 will be given a detailed treatment. After reading this section, a modeller should have little trouble conceptualizing state variable models in the FLEX1 structure.

Reasons for Modelling in FLEX

One might begin by asking why any fixed modelling paradigm and convention is desirable. There are several advantages for taking a fixed viewpoint.

First, it confers the advantage of ease of model conceptualization. If a general paradigm is used, the modeller has only to identify those parts of the structure which are relevant to a particular model and to establish a correspondence between components of the general paradigm and the components of the model. The underlying assumptions of a well known general paradigm will be well known, and will remain constant from model to model. Uniformity of structure and notation is advantageous in coupling models together, in comparing one model with another, and in critically reviewing and evaluating a model. Differences in two models are most readily apparent, and specific model features and assumptions are easily discernible.

Further, a fixed modelling convention allows for ease of communication. A standardized paradigm, with its accompanying standardized notation, makes the interpretation of any particular model realized within it extremely easy. It reduces the amount of information which has to be communicated between several modellers in various subsystem groups and between a modeller and a programmer.

Lastly, the use of a general paradigm allows construction of a general computer program in the image of that paradigm, so that all of the routine business is programmed, for once and all. It is necessary to write computer code only for those features in which the particular models in the general paradigm may change. In essence, this allows the modeller to spend his time and effort on modelling, with a minimum of time on program assembly.

These advantages are obtainable by using any fixed modelling paradigm and convention. FLEX offers some unique advantages.

FLEX is conceptualized so as to make the model <u>structure</u> readily apparent. It is based on a general paradigm which was synthesized by George Klir from study of a great variety of system models, with the explicit form of FLEX dictated by the apparent needs of ecosystem modelling. Model components are structurally and functionally identified, forming, as it were, a paradigm for ecosystem models, in contrast to the opposite extreme of unstructured models presented in order of computation or in some other such arbitrary manner. In FLEX, <u>input</u>, <u>output</u>, and <u>state</u> variables are identified and readily apparent, the hierarchy of functions is clear and unambiguous, and confusion between components should be minimized.

Further, the FLEX algorithm is designed to be a module in an expanded algorithm, called REFLEX. FLEX is utilized as an element in a Universe-Coupling structure (see Klir, 1969) which is realized in REFLEX. This allows for explicit modelling of hierarchical systems. In the FLEX paradigm, it becomes possible to uncouple various submodels from the whole and examine their behavior in a meaningful manner, isolated from the total model. A discussion of many relevant points is given by Overton (1972, 1973).

Identification of FLEX1 Structure

The general structure of FLEX1 can be seen in Figures 1, 2, and 3. Figure 1 is a particular realization of the Generalized Paradigm for Sequential Systems (Figure 4.8, Klir, 1969). Figure 1 diagrams both the particular general conceptual structure and the steps taken in a computational cycle by FLEX1. Figure 2 details this computational sequence in time. Figure 3, FLEXFORM, is the basic specification form for a model in FLEX1. The modeller utilizes this form to

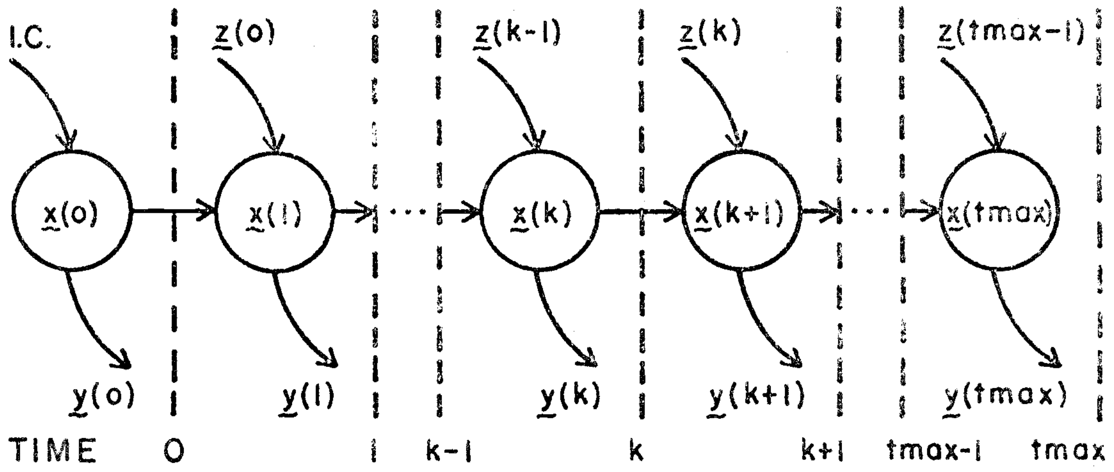


Figure 2. The FLEX1 Sequential Computation File

FLEXFORM

FLEX MODEL OF:	
INVESTIGATOR:	
DATE:	/
TIME RESOLUTION:	
QUANTITY:	

VARIABLES AND FUNCTIONS

1. X List — Description

   - - identification of state variables --

2. Z Functions — Description

   - - specification and identification of input variables - -

3. M List — Memory Specifications

   - - specifications of x and z variables retained as memory variables - -

4. G Functions — Description

   - - intermediate functions - -

5. F Functions — Description

   - - flux functions - -

6. S Functions — Description

   - - special-use functions - -

7. Y Functions — Description

   - - output variables - -

PARAMETERS

8. B Parameters: list, description and values

9. R Parameters: list, description and values

10. Initial Conditions: list, descriptions, and values

    XN - initial X values

    MN - initial memory values

11. Run Parameters

    - - to be completed by programmer using information provided elsewhere - -

Figure 3

FLEXFORM

Page 2 of 2

GENERAL RUN INFORMATION

TSTART = ____________

TMAX = ____________

Variables to be monitored:

Monitor frequency = ____________

Line printer frequency = ____________

Dump file frequency = ____________

SATELLITE PROGRAM INFORMATION

Variables to be plotted:

Plot frequency:

DIAGRAM

COMMENTS

DESCRIPTION OF BEHAVIOR

RUN LOG

Figure 3 continued

communicate, to the programmer and/or user, the correspondence which has been made between the general structure and a particular model. This form requires specification of each model component, as well as the specifications for a simulation run.

FLEXFORM will be discussed in greater detail below. The first two figures will now be discussed.

In Figure 1, an input vector $\underset{\sim}{z}(k)$ enters from the environment at time k. The functional generator uses this input information, the present state of the system $\underset{\sim}{x}(k)$, past inputs or states from the memory $\underset{\sim}{M}(k)$, and certain parameters, $\underset{\sim}{b}$ and $\underset{\sim}{r}$, to calculate the update vector $\underset{\sim}{\Delta}(k)$, the amount by which the state variables will be incremented at time k. The functional generator utilizes s functions, g functions and f functions in the calculation of $\underset{\sim}{\Delta}(k)$.

The state variable vector is updated and time is advanced to $k+1$. Specified input and state variables from time k are stored in the memory. An output vector $\underset{\sim}{y}(k+1)$ is produced utilizing the new state variable vector $\underset{\sim}{x}(k+1)$, updated memory values $\underset{\sim}{M}(k+1)$ and any parameter values. FLEX1 is now ready to accept a new environmental input vector $\underset{\sim}{z}(k+1)$ and the process repeats itself until a terminating condition is reached (i.e., until the specified end of the simulation run is reached).

It is important to understand how and when time is advanced in the sequential procedure. Time is advanced just prior to the calculation of the output vector, as an integral part of the update procedure. The value of k remains the same from the output at time k, through the input and function generation, until the update portion of the cycle.

Note especially that the cycle which precedes time zero is a transformation of the initial conditions by the output processor to produce an initial output vector $\underset{\sim}{y}(0)$. This calculation does <u>not</u> involve any z's.

Specification of a Model

FLEXFORM, as previously explained, is the basic modelling tool of FLEX1. Filling in this form completely finishes the specification of a model for a FLEX1 run. A programmer can take this sheet, produce FLEX1 FORTRAN code, make computer runs and return the specified outputs to the modeller. Because of its importance, the rest of this section will be concerned with use of FLEXFORM.

<u>Time Resolution</u>. Time resolution refers to that length of time represented by the time increment from time k to k+1. The discrete time index, k, is incremented by 1 at each cycle of the sequential analyzer. However, the meaning of the increment can be varied by letting it stand for different time resolutions, i.e., one second, five minutes, four hours, five days, one year, etc.

The choice of a resolution level is an essential part of the model building process. It is dependent on and influences both the general structure and the equations which are used in a particular model. In general, a system with fast dynamics requires a model with fine temporal resolution and one with slow dynamics should be modelled with coarse resolution. Changes in resolution can require changes in functional forms and model structure.

Quantity. The state variables represent measures of the quantities modelled such as energy, nutrients, or population numbers. Several quantities can be modelled at one time.

Variables and Functions. The variables and functions outlined on the left side of page 1 of FLEXFORM specify the model structure in FLEX1. Principal variables are input variables, state variables (the X list) and memory variables (the M list). Z functions specify the input variables to the system from its environment. This information is processed by the Function Generator, represented by the g functions, f functions and s functions, to produce an update, or increment, vector. After the state variables are updated, an output vector is produced by the Y functions. Each of these structures and operations will be discussed in detail.

The X List. The x variables represent state variables. There is a limit of 63 x's, each one identified by a subscript from 1 to 63.

In the X List it is only necessary to specify which x's will be used by writing them down in a column. Incides should be used sequentially, unless there is some specific reason for skipping an index. In a model with 5 state variables, $x_1, x_2, \ldots, x_5$ should be used.

Across from each x is a place to enter a description of the state variable. Thus x_1 could be identified as canopy water storage. Acronyms can be used as a description, but fuller descriptions are desirable. These descriptions are for the convenience of the modeller or model user; they have no effect on FLEX1. Such descriptions are necessary in any model with more than a few state variables. They are also useful in translating an existing model into FLEX1.

If it is desired to constrain the values of any of the x variables within certain limits, these should be specified within the description of each state variable. In FLEX1, the lower limit defaults to 0. This is usually the desired value in biological models. However, if negative values are to be allowed, the lower limit <u>must</u> be specified to be negative. If the lower limit is reached, the state variable will be maintained at that value.

The default value for an upper limit is 10^{100}. If a variable passes its upper limit, the FLEX1 run will terminate. If an x variable should be maintained at some upper limit, the value of Δ_i must be curtailed within the g or f functions.

As indicated earlier, the time resolution may need adjustment depending on the dynamics of the model. A check may be made by specifying the maximum increment Δ, such that if this increment is exceeded, the simulation run will abort. Indicate the maximum increment within the description of the x variables.

Initial conditions for state variables are specified in the initial condition list, item 10 on the right side of FLEXFORM.

The Z Functions. The z functions are used to calculate the values of the elements of the input vector. There is a limit of 40 z variables.

The user of FLEX1 should keep in mind that since this is a single level processor, there is no external coupling, and the inputs (i.e., the z vector) are processed and generated as though they were internal functions. However, it is important that the conceptual distinction of input quantities be retained; these are quantities (variables) arising externally to the system being modelled, but subject to the system receptors. Both generation of the quantities and the essential input modifications are included in the z functions.

The z's should be indexed sequentially. The expression to be used in calculating a z value should be specified as part of the definition. This expression may be a constant, an algebraic function, or a table look-up function.

The $z_i(k)$ may be a function of the s functions and also of $z_j(k)$, where $j < i$.

In the normal FLEX1 computation cycle, z values are calculated first. On the first cycle of a run an output vector, $\underset{\sim}{y}(0)$, is calculated on the basis of initial conditions provided prior to the first entry into the FLEX1 computation cycle.

<u>The M List</u>. In this list those principal variables which are elements of the memory are identified. This includes memory for both x variables ($\underset{\sim}{m}\underset{\sim}{x}$) and z variables ($\underset{\sim}{m}\underset{\sim}{z}$). It is necessary to list the index of the variable to be retained in memory and the number of past values to be retained. Past values of all variables specified will be maintained for all past times up to the maximum number specified. Thus if ten past values of x_1 are needed and only four past values of the other memory variables are required, all are still maintained for ten past time steps.

There is a limit of 200 memory storage places. If the total number of variables to be retained in memory multiplied by the maximum number of past values for any one variable exceeds 200, then the memory specifications must be changed.

Within other equations, memory variables are indicated as follows: for the rth past value write $x_i(k-r)$ or $z_i(k-r)$. This establishes a uniform notation which simplifies the later coding (programming) details.

Initial conditions for memory variables must be specified for a run. These will be entered in the initial condition list, indicated on the right side of FLEXFORM.

The G Functions. The g's are intermediate functions. The g's are calculated sequentially and the value is stored for the remainder of that iteration. There is a limit of 70 g functions.

G functions are used to calculate and store values which are used more than once within a time step or otherwise for convenience in model specification. They may be used to simplify the calculation of values for the f functions. They may be used to represent a process or quantities such as food demand or supply. Thus, for example, in certain predator-prey relationships, the demand on a prey by each of several predators may be calculated in individual g functions and this information used to determine the total demand and later to apportion the captured prey among the predators.

G function arguments may be any of the principal variables, previously calculated values of g functions, b and r parameters, and values of special functions. Thus,

$$g_i(k) = g_i[k, \underset{\sim}{x}(k), \underset{\sim}{z}(k), \underset{\sim}{M}(k), g_j(k), \underset{\sim}{s}, \underset{\sim}{b}, \underset{\sim}{r}), \quad j < i$$

Most internal calculations can be accomplished using g functions.

The F Functions. F functions describe the fluxes between the state variables. Thus

$$f_{ij} \in \underset{\sim}{F}$$

defines a flux from x_i to x_j and this value will be added to x_j and subtracted from x_i. If there are n defined state variables, then $\underset{\sim}{F}$ is an n x n matrix; many elements are zero.

F functions can be a function of the input variables, state variables, intermediate functions, parameters and special functions. Thus

$$f_{ij}(k) = f_{ij}[k, \underset{\sim}{x}(k), \underset{\sim}{z}(k), \underset{\sim}{M}(k), g(k), \underset{\sim}{s}, \underset{\sim}{b}, \underset{\sim}{r}]$$

Notice that f functions <u>cannot</u> be functions of each other explicitly. G functions should be used to accomplish this implicitly.

The existence of the flux f_{ij} indicates a connectance between elements x_i and x_j. (However, it is possible to write $\underset{\sim}{F}$ such that connectances are not explicitly shown.) In a connectance matrix, the diagonal elements are all 0, indicating that no element is coupled to itself. In FLEX1, the diagonal elements are used to indicate inputs and outputs to that variable involving only that element. For example, reproductive increases, environmental inputs and respiration output would belong in the diagonal element if no other state variable receives the change.

As earlier indicated, the orientation to the matrix of f functions is in accordance with the compartment model paradigm, and the off-diagonal elements will be most useful when the model is so conceptualized. In the more general discrete time model formulation, we may wish to define Δ_i explicitly as f_{ii}, so that no use is made of the f_{ij}, $i \neq j$. This is also a way to get around the limitation of number of f functions. There may be no more than 63 f functions, yet the limit of 63 state variables provides a potential of 63^2 flux elements. Even though this maximum number will clearly never be needed, it is not unreasonable for a 20 variable system to call for more than 63 connectances. In such an event, a compartment model can be implemented by the device of defining some of the f_{ii} equal to the Δ_i.

The most important limitation (for FLEX1) in these large models will be the size of the overlay of the user supplied functions and subroutines. The number of allowable functions and variables is unlikely to be limiting, and should be given little concern in use of FLEX1.

It will often be desirable to monitor fluxes at equilibrium or in evaluation of cyclic balance. This can be done by assigning the appropriate quantity from which outputs can be generated to a state variable.

<u>The Special Functions</u>. Special functions are included to provide greater modelling flexibility and ease. S functions may have as arguments input variables, state variables, memory variables, parameters and other special functions. Thus

$$s_i = s_i[k, \underset{\sim}{x}(k), \underset{\sim}{z}(k), \underset{\sim}{M}(k), s_j, \underset{\sim}{b}, \underset{\sim}{r}], \quad j < i.$$[1/]

[1/]It is necessary that a sequencing exist such that $j < i$ and that any z variable called has already been computed.

S functions may be vector valued and are specified by an argument string. These can be evaluated as many times as desired during a single time step and with a variable definition of the elements of the argument string at each evaluation. S functions may be table look-up functions and, in fact, are best used to input environmental and driving data for use by the z functions. They may also be used for often duplicated functional forms.

There is no particular time in the FLEX1 computation cycle at which the value of each special function is calculated. They are available at any time in the computation cycle. There is no set limit to the _number_ of special functions allowed. There _is_ an overall limit on the space available for storage of user-defined functions, including z, g, f, s, and y functions. If many other functions are used, there will be little room for special functions.

The Y Functions. Y functions are used to calculate the output vector $\underset{\sim}{y}(k)$ from the updated $\underset{\sim}{x}(k)$ values. There is a limit of 50 output variables. This output vector is produced _after_ updating the state variables, memory variables, and incrementing time so the modeller must remember the k used here is equal to k+1 in the preceding discussion (see Figures 1 and 2). Y functions may be functions of the state and memory variables, the parameters, s functions and other previously calculated y functions. Thus

$$y_i(k) = y_i[k, \underset{\sim}{x}(k), \underset{\sim}{M}(k), \underset{\sim}{b}, \underset{\sim}{r}, \underset{\sim}{s}, y_j(k)], \qquad j < i$$

Although any output which is thus defined may be produced, the most frequent case will be where $\underset{\sim}{h}(\underset{\sim}{x}) = \underset{\sim}{I}\,\underset{\sim}{x}$, the identity function, so that $\underset{\sim}{y}(k) = \underset{\sim}{x}(k)$.

Remember that there is an output vector, $\underset{\sim}{y}(0)$, produced prior to the beginning of the FLEX1 computation cycle. This is the initial translation of the model's specified initial conditions, $\underset{\sim}{x}(0)$ and $\underset{\sim}{M}(0)$.

Parameters. On the right side of FLEXFORM is a summary of the information necessary to specify the actual numbers used in a simulation run. Only the first three of the four are of interest to the modeller. The last, the run parameter file, will be filled in by the programmer from other specified information.

The constants of the model have been divided into two vectors, the b and r parameters. In addition, the initial values of the state and memory variables must be specified.

The B and R Parameters. In this section, the b and r parameters to be used are listed. They may be described by any name or acronym the modeller wishes. The value to be used for that parameter is also listed. The default values are 0. There are limits of 100 b parameters and 20 r parameters. Parameter indices should be assigned sequentially.

R's were originally differentiated from b's in order to identify respiration constants, but there are many criteria by which the parameter vector can be conveniently partitioned into its two subvectors, $\underset{\sim}{b}$ and $\underset{\sim}{r}$.

The program allows changes in the values of the parameters during a FLEX1 simulation run. This capacity may be utilized in many ways to simplify model structure.

The Initial Condition List. This list has two sections, the XN vector, $\underset{\sim}{x}(0)$, and the MN matrix, $\underset{\sim}{M}(0)$.

Initial x values are listed first. These are the values which will be assigned to the respective x_i prior to beginning a FLEX1 simulation run, i.e., at time $k = 0$. They should be listed sequentially. The descriptions should match those of the X List. Default values are 0.

Initial memory values are listed next. These are the values which will be assigned to both the mx and mz variables in memory prior to beginning the FLEX1 computation cycle. All variables listed in the M List should have an initial condition specified. Default values are 0.

General Run Information

The information necessary to make a FLEX1 simulation run is listed. This information includes the length of time for which a model is to be simulated and what mode of output is desired.

Run Time. If a simulation run should start at some time other than 0, this time must be entered after TSTART= . TSTART need only be specified when necessary, as this quantity defaults to zero. The time for which a last output is desired is entered after TMAX= . There is no default value for TMAX, so this must be specified.

Monitor, Line Printer and Data Dump File. Output resolution may be coarser than Δt so that output costs are less. Since FLEX1 is a difference equation analyzer, the time step required for accuracy may be much smaller than the time step wanted for the most detailed output. Generally the data dump file will have the finest output resolution; this file will be available for further investigation of the run. The line printer output can have the same resolution as the dump file, but this is not necessary; the line printer could be treated as a more extensive monitor, with fewer variables and coarser time step than the dump file. The monitor should have a fairly coarse time step and fewer than six variables, since this information must be printed at the terminal. The teletype monitor is intended only to provide a rough check of progress of a run.

Output time resolutions are defined in terms of cycles. To monitor every nth cycle, enter n for the monitor. N defaults to 1 in all these cases.

Diagram. A diagram of the model should always be included. A block diagram is usually sufficient. An identification of g functions on the diagram is helpful.

Comments. Here the modeller may make comments to himself, to his programmer and to the model user. These include reasons for including or excluding certain things, what is desired from repeated runs, information sources, acknowledgements, etc. Anything which the modeller thinks is pertinent may be entered.

This is the place to talk about any unusual features of the model. A brief synopsis or abstract of the model's purpose may be deemed appropriate.

Description of Behavior. The modeller should use this space to describe how the model is designed to behave prior to the actual simulation of the model. This description is a summary of the general understanding of the behavior of the system being modelled and the simulation output should be studied to verify that the model behaves in the prescribed manner.

If the results of a simulation run are at variance with this description, that is, if the model behavior does not conform to the prescribed system behavior, a record of anomalies and surprises should be made in this section.

III. PROGRAMMING IN FLEX1

Introduction

Programming a FLEX1 run involves translating the modeller's information (FLEXFORM, Figure 3) into specific computer commands and formats. If the FLEXFORM sheet is filled in completely, the programmer should have all the information necessary for this translation. FLEX1 is partially FORTRAN based and programming FLEX1 demands the ability to write FORTRAN equations. In addition, knowledge of the use of OS-3 (Oregon State Open Shop Operating System) is necessary, as FLEX1 was written exclusively for this interactive time-sharing system. Information on OS-3 may be obtained from the OSU Computer Center and a list of relevant materials are listed in the bibliography.

Overview

An overview of the FLEX1 system may be obtained by consulting Figure 4. This figure illustrates the structural aspects of the FLEX1 system within the larger structure of the OS-3 system. In addition, the upper portion of this figure is a diagrammatic outline of the steps necessary to program a model for FLEX1. The details of this process are listed in the section Programming Details. An outline of the steps follows.

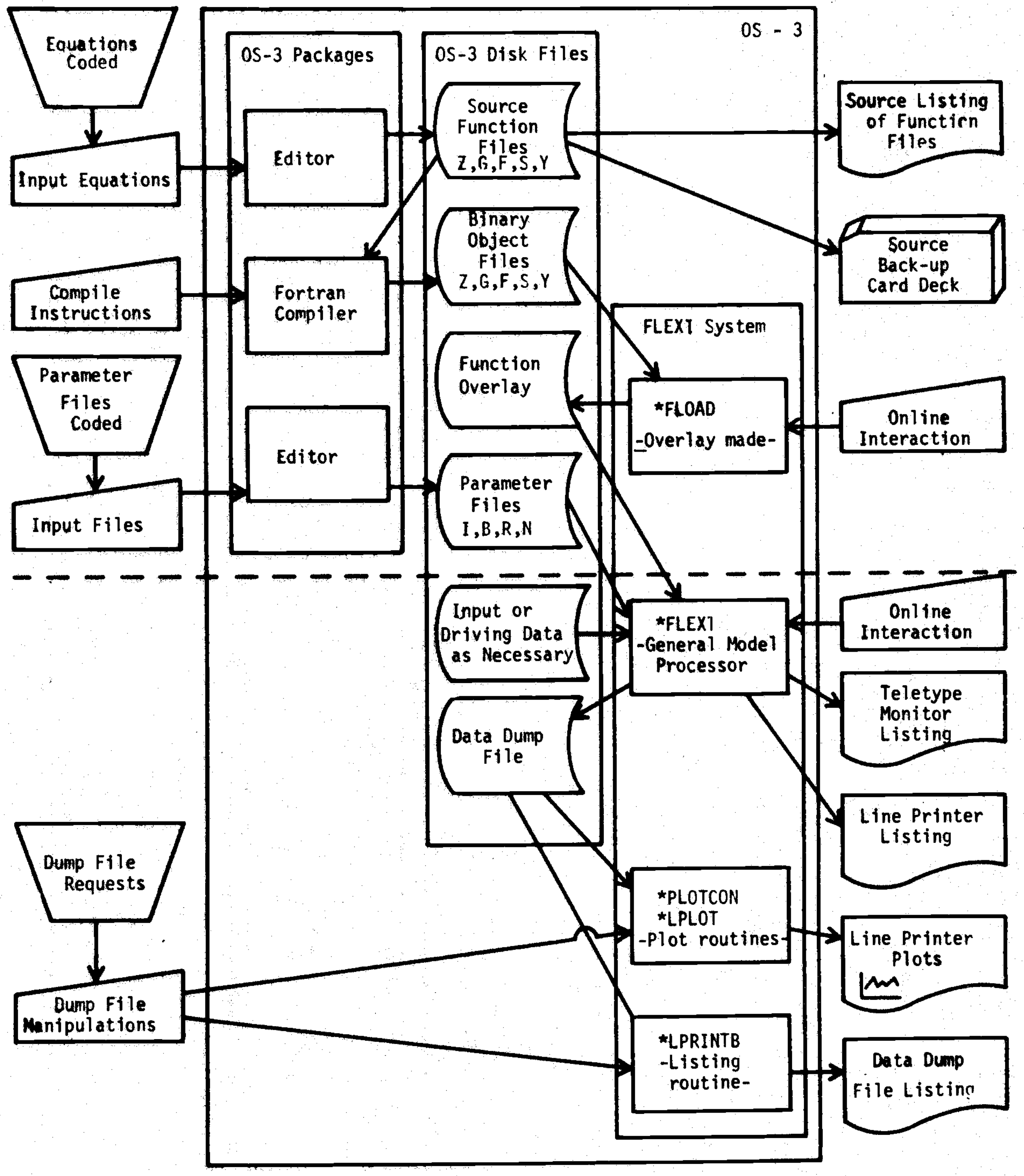


Figure 4. An Overview FLEX1 Operating System

Translate the model functions into FORTRAN code. The z, g, f, and y functions have a standardized format given below. S functions follow the general FORTRAN language format. Generate a function source file for each of the five types of functions, using the OS-3 EDITOR. Source listings and backup card decks are usually made. Compile each source file, using the OS-3 FORTRAN compiler. Use *FLOAD, part of the FLEX1 system, to incorporate all the resultant binary object files into a function overlay which is stored on a disk file under a user-specified name.

Enter the parameter information using the standard format specified below. Generate a FLEX1 command file for B, R, Initial Conditions, and Run parameters. These four files are _not_ compiled. Parameter listings and backup card decks are usually made.

When the function overlay and parameter files are complete, the model is ready for a simulation run.

Although not indicated in Figure 4, a copy of the parameter files should be appended to the function source listing and back-up card deck.

Programming Details

A. Coding the Model Functions. We are concerned with the z functions, g functions, f functions, s functions and y functions from FLEXFORM. Although we have been calling them functions, the correct mathematical terminology, they include both subroutines and functions in computer terminology. We will explain these in order. A summary of the formats is included as Appendix Section 2.

Z Functions

The number of z or input values is limited to 40. All z values are computed in one subroutine. The standard format is

```
                SUBROUTINE ZCOMP (K, X, B, R, Z)

                DIMENSION X(1), B(1), R(1), Z(1)

                Z(1) = expression

                Z(2) = expression
Calculation     .
of              .
z values        .
                Z(<n>)= expression

                RETURN

                END
```

The expression may be a constant, or a time-varying algebraic function, or a table look-up. Input data is most often read from disk data files in an s function called by the z function.

Z values are calculated first by FLEX1. The values are stored internally and may be used in other functions by simply writing Z(1) or Z(Ø1) for z_1, etc. in the appropriate equations. The values may be used by any other functions except the y output functions.

G Functions

The number of g functions is limited to 70. A separate FORTRAN function is required for each g function. Each g function is coded in the standard format

```
                    FUNCTION GØ1(K, X, B, R, Z)
                    DIMENSION  X(1), B(1), R(1), Z(1)
Calculation         .
of the              .
value of g1         .
                    GØ1 = expression
                    RETURN
                    END
```

Ø2, Ø3, . . . , 7Ø may be substituted above for Ø1; two digits are required. The calculation of the returned value may be as long and complicated as desired. G functions are defined as intermediate functions; any internal variable which will be used extensively and which retains the same value throughout a time increment should be calculated as a g function. G functions may be used for a single process, to simplify the f function representation, for calculation of demands, or for logical operations.

G values are calculated second by FLEX1. All defined g functions will be calculated and the returned values stored internally. The value is available to the f functions or <u>other</u> subsequently calculated g functions or special functions by writing G(Ø1), G(Ø2),..., G(7Ø). The use of parentheses is <u>crucial</u> and these must always be included.

F Functions

There is a limit of 63 f functions. A separate FORTRAN function is required for each defined f_{ij}. The standard format, repeated for each f function, is as follows:

```
                   FUNCTION FØ1Ø1 (K, X, B, R, Z)

                   DIMENSION X(1), B(1), R(1), Z(1)
Calculation        .
of the value       .
of f 1,1           .
                   FØ1Ø1 = expression

                   RETURN

                   END
```

and where Ø2, Ø3, . . . , 63 may be substituted for Ø1 (i.e., FØ1Ø2, . . . , F6363); four digits are required. The returned values are stored internally and will automatically be manipulated to calculate the update vector $\underset{\sim}{\Delta}(k)$ which will be added to the current state variable vector to obtain the next time step's state vector. Please note that the f functions cannot be used to calculate other functions. If the value of one flow determines that of a second, the first value should be calculated as a g function and used in both f functions. A g function used as an argument must be used with indices in parentheses (e.g., FØ1Ø1 = G(Ø5)).

The calculation of f values is the third step in the FLEX1 cycle, just prior to the calculation of the update vector and the new state variable vector.

S Functions

Special functions are included for further modelling and programming flexibility. They may be either functions or subroutines. There is no limit to the number allowed except for the limit on the function overlay size which is $14,000_8$ (octal) words.

There is no standardized format for the special functions; function and subroutine formats in the C.D.C. FORTRAN handbook are used. Special functions may not be used to define COMMON area storage. S functions are functions of time k, state, input and memory variables, b and r parameters and other special functions only. Values of g, f, and y functions may not be used.

Special functions may be used in calculating z values, g values, f values or values of other special functions. They are useful as table look-up or data read-in functions, especially for z functions, where real environmental data are entered.

There is no particular time in the FLEX1 computation cycle where the values of the s functions are calculated, nor are these values stored internally. Their value may change during a time step since the definition of the elements of the argument string may change each time a function is called.

The s functions operate the same way with regard to FLEX1 and any of its user defined function sets as functions and subroutines operate with regard to a main FORTRAN program.

Y Functions

The number of y output values is limited to 50. Y values are calculated by a subroutine. The standard format is

```
                  SUBROUTINE YCOMP (K, X, B, R, Y)
                  DIMENSION X(1), B(1), R(1), Y(1)
                  Y(1) = expression
                  Y(2) = expression
Calculation       .
of y              .
values            .
                  Y(<n>)= expression
                  RETURN
                  END
```

The expression for $y_i(k)$ may be a function of k, $\underset{\sim}{x}(k)$, $\underset{\sim}{M}(k)$, $\underset{\sim}{s}$, $\underset{\sim}{b}$, $\underset{\sim}{r}$, $y_j(k)$ for $j < i$. Usually each x is matched with a y. In some cases one may want to know the accumulated total of a certain x value or the accumulated flux over a certain pathway. In such cases, it is necessary to construct an x variable to provide this output. Y values may not be used to calculate anything except another y value.

The calculation of the y or output variables is the fifth and final step in the FLEX1 cycle. Before the y output is produced, time (k) is advanced (to k+1). Thus the k in this section is k+1 with regard to the k in the preceding sections. The first output $\underset{\sim}{y}(0)$ is a translation of the initial conditions, $\underset{\sim}{x}(0)$ and $\underset{\sim}{M}(0)$, before the first FLEX1 computation cycle begins.

B. Preparing a Function Overlay. After entering the coded functions onto disk files, debugging and compiling them, make the function overlay.

While in OS-3 control mode, indicated by a # sign, type

*FLOAD,F= <overlay name>, <file name>,..., <file name> (CR)

After F= specify an eight character file name. The overlay will be saved under this name.

After the overlay name, enter the names of the binary object files used, ZCOMP subroutine, G functions, F functions, YCOMP subroutine and S functions. Separate file names with a comma. The use of a special library may be included by adding ", L = <library name>" after the last file name. Type a carriage return (CR) to terminate the line. The pound sign (#) will be typed after the overlay is made. File protect all overlays. Guidelines are offered for file structure naming in A Note on Naming Files and Overlays below.

The name of the overlay will be used with the command FUNLOAD within the FLEX1 system. This name must be entered correctly to be loaded and used by FLEX1.

There is a size limit of $14,000_8$ or 6144_{10} words on the overlay. If too many functions are specified, either the structure must be reworked or part of the model deleted. *FLOAD error messages are discussed in Appendix Section 3.

C. Entering the Parameter Information. Although parameter information can be typed directly into the FLEX1 processor during a simulation run, speed and accuracy will be improved by entering FLEX1 commands into a file and using a single typed command to bring them all into FLEX1. For convenience in retrieving and changing commands, four files are created, the I-file, the B-file, the R-file and the N-file. Each file contains information from one section on the FLEXFORM. After all information for a file is typed in, terminate with the command INPUT= 6Ø. This command returns control to FLEX1.

Appendix Section 2 contains a summary of the FLEX1 commands which may be used in each file.

FLEX1 has a free-format input processor. The general form of FLEX1 commands is COMMAND= <value>. One element of a vector may be entered by COMMAND(i)= <value>. One element of a matrix is entered by COMMAND(i, j)= <value>. Separate commands from one another by one or more spaces. A vector of values may be entered by COMMAND( )= <1st value>, <2nd value>, . . . , <last value>. Separate values within a command with commas. Matrices can be entered by row, COMMAND(i,)= <1st value>, <2nd value>, . . . , <last value>, or by column, COMMAND(, j)= <1st value>, <2nd value>, . . . , <last value>, or by elements.

Go to a new line wherever the material can be separated, preferably between commands, or between successive values of a command.

Do not break a name or a value.

Numbers may be entered with or without decimal points. They should be separated by commas. Decimal points do not have to align nor is it necessary to use the same columns repeatedly. Thus, for example, a parameter B-file might be:

```
B( ) = 1.273, .Ø13, 1392, 14Ø.59, 82
INPUT=6Ø
```

This sets $b_1 = 1.273$, $b_2 = .013$, $b_3 = 1392$, $b_4 = 140.59$ and $b_5 = 82$.

C.1. Run Parameter I-File. The run parameter I-file is a catch-all file with a large variety of information included. It includes (a) M List from the FLEXFORM with the MN section of the Initial Conditions, (b) constraints on values of various x variables from the X List, (c) output specifications from the general run information section of the FLEXFORM. The commands will be discussed in the order in which they must be entered on the file.

C.1.a. Memory Specifications. From M List on FLEXFORM a variety of information must be obtained. In FLEX1 the memory is a matrix with the rows representing variables and the columns the time delays. There is a limit of 200 memory variables. If the total number of variables retained in memory multiplied by the maximum number of past values for a variable exceeds 200, the modeller should be notified and the length of the maximum lag reduced or fewer variables retained.

The memory matrix is dimensioned by the maximum number of lags desired. The variable LAG is the length of time that all the variables will be retained in the memory. It must be entered before any other memory specifications. Next the indices of the x and z variables which will be delayed must be entered. LAGX( )= is followed by the indices of the x variables to be delayed, separated by commas. LAGZ( )= is followed by the indices of the z variables to be maintained in the memory, also separated by commas. All of this information can be entered on one line as follows

LAG = _____ LAGX( ) = _, _, _ LAGZ( ) = _, _, _

Initial values for the memory must be next. These are listed in the MN section of the Initial Conditions. These may be entered by row vectors. Thus, for LAG=3 LAGX( )=1 we might have

XD(1,) = 12., 11., 1Ø.

XD(1, 1) corresponds with $x_1(k-1)=12.$, XD(1, 2) with $x_1(k-2)=11.$, XD(1, 3) with $x_1(k-3)=10$. The values may have decimals specified or not, and must be separated with commas. XD, and ZD, values may also be input by column, such as

XD(, 1)=1, 9, 5

where XD(1, 1) = $x_1(k-1) = 1$, XD(2, 1) = $x_2(k-1) = 9$. and XD(3, 1) = $x_3(k-1) = 5$. Entry by column vectors is often more convenient if fewer than 3 or 4 lags are used.

Next, the initial ZD values should be entered, following the same format as the XD variables. Thus, for LAG=4 LAGZ( )=2, 5 we might have .

ZD(2,) = 1, 4, 7, 1Ø ZD(5,) = 1., 3., 12.2

Since no value is entered for ZD(5, 4), its value defaults to 0. This value might not be needed in the model. That is, LAG=4 might refer to z_2 needing 4 delays.

C.1.b. X Limits. If the X List from FLEXFORM specifies that a variable x_i is to be maintained between certain limits, these limits are entered next.

For a lower limit, the command XL(<i>)=, followed by a value, is used. If this limit is reached, the value of x_i will be maintained at that limit. The default lower limit is 0.

For an upper limit, the command XU(<i>)=, followed by a value, is used. If this limit is passed or reached, the FLEX1 run will abort. Default value is 10^{100}.

Finally, it is possible to specify the amount by which it is permissible for a variable to change. The command XE(<i>)=, followed by a value, is used for x_i. If

$$|\Delta_i| > XE(i)$$

and $XE(i) \neq 0$

then the simulation run will abort. If XE(i) is 0 (the default value) no checking is done.

C.1.c. Output Specifications. As illustrated in Figure 4, three specific outputs are produced by FLEX1. These are the teletype monitor listing, the line printer listing and the data dump file. Each of these will be explained in turn, although they all have similar commands.

By convention, y values are usually the only variables sent to the line printer and data dump file. State variables are added as necessary to accommodate the output of z's, f's and g's. However, it is possible to output the values directly to all units and not just the monitor.

The line printer output is commonly used as a record of the dump file. It is strongly recommended that the two outputs agree in variables, order of variables and frequency of output. (Note that once the output order of variables is established for one output unit, all other units will maintain that same order for these variables.)

First, enter the command YMAX=, followed by the number of y variables defined in YCOMP. This number should be the largest index value used if the y's are not numbered sequentially.

Next, enter the teletype information. The monitor time resolution, from the FLEXFÒRM, is entered with the command TTYPRT=_. This will default to 1 if not specified. The variables to be monitored are entered with TTY= followed by what is to be monitored. For example

TTY=G(1)   TTY=F(2, 2)   TTY=Y(7)

and so forth. Anything may be monitored, but the more variables monitored the longer and more costly the run.

Next, line printer information is entered. LPRT=___ is the command for entering the line printer time resolution. This will default to 1 if not entered. The variables to be printed are entered by commands of the form

LP=Y(7)   LP=G(1)   LP=F(8, 8)

All Y variables which are defined should usually be line-printed. The logical unit number (<lun>) assigned to the line printer must be entered with the command LPLUN=<lun>. This lun should be equipped to the LP and labeled before a FLEX1 run.

Lastly, information for the data dump file must be entered. First, the dump file time resolution is entered with DPRT=___. This should be the same as that entered for the line printer and will also default to 1. Variables to be dump filed are entered as above using D=___. For example

D=Y(1)   D=Y(22)   D=X(5)   D=G(7)

There is a limit of 79 variables which may be dumped during a run. The lun assigned to the dump file must be entered with the command DUMPLUN=<lun>. This lun should be equipped to a file before a FLEX1 run.

Line printer and dump file lun equipping need be done only one time, before the first call to FLEX1. If several runs are made, each run's information on the dump file is separated from the next by a file mark.

Some of the run parameter information may be omitted. For example, if the memory is not used, simply skip that set of instructions.

C.2. Parameter B-File. There is a limit of 100 b parameters. These may be entered one at a time, i.e., B(1)=__ B(2)=__ B(3)=__ etc. or as a vector, i.e., B( )=__,__,__. Values should be separated by commas. Numbers must be entered so that the end of a line comes between two numbers, not in the middle of one. All values not specified default to 0.

C.3. Parameter R-File. There is a limit of 20 r parameters. Input form is the same as for b's, either one at a time, R(1)=__, R(2)=__, or as a vector R( )=__,__ . All values not specified default to 0.

C.4. Parameter N-File. Initial x values may be entered one at a time or as a vector. XN(1)=__ XN(2)=__ or XN( )=__,__. Values not specified will default to 0.

All of the above input files must terminate with a command to return to FLEX1. This command is INPUT=6Ø.

A Note on Naming Files and Overlays

The translation of several models into FLEX1 has demonstrated the necessity for a file naming convention to keep the user and programmer from getting lost in a maze of file names. The following convention has proved useful.

First, construct a 3 to 5 alphanumeric character code for the model. Thus for a litter decomposition model we might use LIDE. This code will form the root name of our files.

Function file names for source decks are obtained by appending a letter and number. If this is the first Z subroutine, i.e., the first of several options, use LIDEZ1 as its file name; if it is the fourth F function file of several alternate files, use LIDEF4, etc. G functions become LIDEG_, Special functions become LIDES_, and Y functions become LIDEY_.

Compiled FORTRAN binary files are named with a * prefix. Source FORTRAN files are named without a * prefix. Binary decks are public files, so they may be used from a different job number. Back-up card decks of the source files should be maintained.

For the suffix of parameter files, use I, B, R, or N along with a number. Prefix these with a * so that they may be used from another job number. For example *LIDEI4, *LIDER3, *LIDEN12, etc.

Function overlay names have an asterisk * prefix and a two digit (no letter) suffix. Thus *LIDEØ1, *LIDEØ2, etc.

Data dump file names should have a star prefix and a D with a number suffix, i.e., *LIDED1, *LIDED2, etc.

Such a name structure carries with it implicitly the FLEX structure, allowing a simple mnemonic code with varying endings to stimulate one's memory. This is the only time that acronyms are necessary. The acronym should be constructed using the preceding rules. With a knowledge of the three to five letter code, a run may be made very quickly without consulting a long list of file names.

It is an advisable practice to file protect all files, especially public files (those whose name begins with a *).

IV. USING FLEX1

Introduction

The actual use of FLEX1 may be divided into three parts. The relationship of these parts is diagrammed in the lower portion of Figure 4, An Overview FLEX1 Operating System.

First, a variety of information is input to the FLEX1 processor. This includes the overlay and parameter file names. Such input requires on-line interaction of the user with FLEX1.

Second, the actual simulation run is made. The FLEX1 computation cycle is repeated until a terminating condition is reached (i.e., time has achieved its final value, the upper limit on a state variable has been exceeded, etc.). During the run, a monitor listing of some of the system's variables is possible. A complete line printer listing of the run is produced, if desired, and all output variables are dumped to a file.

Third, the dumped data may be manipulated to produce line printer plots. These plots, in addition to the line printed output, are helpful in understanding the behavior of a model.

Preliminaries

Before a call is made to FLEX1, the logical unit (<lun>) numbers assigned to the line printer (LPLUN=<lun>) and to the dump file (DUMPLUN=<lun>) in the run parameter I-file (see previous section for details) must be equipped to the line printer (LP) and a file (FILE). Thus,

```
#EQUIP, <lun>=LP, <lun>=FILE
```

It is necessary that line printer output be labelled. Thus,

```
#LABEL, <LP lun>/ SAVE FOR <user identification>
```

Calling FLEX1

While in OS-3 control mode, type *FLEX1. Thus,

```
#*FLEX1
```

The FLEX1 system will respond by typing a heading and the statement ENTER INPUTS/COMMANDS. FLEX1 has its own control mode which is represented by a glitch (>). After FLEX1 has typed its command glitch, the user is ready for the first phase of actual use, the input of information and file names to FLEX1.

Commands

The various commands used to input information to FLEX1 will now be detailed. The order of their entry is important, so these commands will be discussed in the order in which they should be entered. This restriction on order of input should be regarded as a checklist, a way to insure that all the information needed for that run has been specified. A summary of this order appears in Appendix Section 2.

More than one command may appear per line (after each command glitch). The two exceptions to this rule are the INPUT=<name> and the

TITLE=<some title> commands. When either of these two commands are used, it must terminate that line. Each line may be up to 136 characters long, including spaces and editing symbols. As in the previous section, in which the parameter files were constructed, each command is separated from the next by a space. It is possible to edit commands prior to carriage return. Consult the Editing section below.

Each line of command input terminated by a carriage return (CR) is processed as a whole by FLEX1. After the commands and edit symbols are processed, FLEX1 may print one of several possible messages. A command glitch is then typed indicating that FLEX1 is ready for further input. If an error message occurs, consult Appendix Section 3 for the proper response.

A. Number of State Variables. This command is used to tell FLEX1 how many state variables are being used. The form of the command is

NUMBER=__

This command must always be the first command given. The number entered following the equal sign is the integer value of the largest x index used. Thus, if x_1, x_2, x_4 and x_5 are being used, but x_3 is not, the proper command is NUMBER=5.

B. Time Limits. It is assumed that most models will be run with time (k) having an initial value of 0. If a different starting time is desired, enter

TSTART=__

This will set the value of k equal to the integer specified. Default value of TSTART is 0, so enter only if necessary.

A model will be run until the termination time is reached, if no other terminating conditions are reached first. This time must always be greater than the TSTART time. Terminating time is entered with the command

TMAX=__

Output is provided for k=TMAX, so this should be the time when the last output is desired.

After each iteration, time (k) is incremented by one. This step size cannot be changed.

C. The Function Overlay. This command tells FLEX1 the name under which the function overlay was saved. The command form is

FUNLOAD=<overlay name>

The overlay name is a file name, up to eight characters long. The function overlay specified will be loaded into FLEX1 and used for that simulation run. This must be the same file name that was used with the F=<name> parameter in *FLOAD.

D. Parameter Files. The names of the four parameter files are input in sequence to FLEX1. For each file, use the command

```
INPUT= <name of I-file>  (CR)

INPUT= <name of B-file> (CR)

INPUT= <name of R-file> (CR)

INPUT= <name of N-file> (CR)
```

After each of the above commands, FLEX1 will access that file and read commands from it. The last command in each file is INPUT=6∅ which will return control to the user. Wait for the command glitch before entering subsequent commands.

E. The Title. It is advisable to label each simulation run with its own title. This makes it possible to establish the identity of each run without excessive trouble. It also allows other user oriented information to be entered.

The title may be at most 80 characters in length. This includes blanks. The command form is

```
TITLE=__
```

followed by the title. This command must terminate that input line.

F. Logs and Summaries. The word log is used in its meaning of a record of events. This command is used to indicate whether line printer and dump file records of this run are being kept. In debugging or exploratory simulation, the monitor output is often all that is desired.

This command, as well as the next, is actually a pair of commands. They are

LOG

NOLOG

If no log is being kept (i.e., no line printer or dump file output) then state NOLOG. This command will remain in effect until a LOG command is entered or the end of a modelling run is reached, at which time the default value (LOG) is automatically set. If neither command is used, LOG is assumed.

A summary of the run conditions (i.e., number of state variables, initial conditions, parameter values, etc.) will be produced by FLEX1 automatically. This information is also available on the teletype. If the command SUMMARY is entered, this summary will be sent to and printed on the teletype. If NOSUMMARY is entered, this summary will still go to the line printer and dump file.

These commands operate as a pair,

SUMMARY

NOSUMMARY

and entering one negates the other. One command will remain in effect until the other is entered or until the end of a simulation run is reached, at which time the default value (SUMMARY) is automatically set. If neither is used, SUMMARY is assumed.

G. Listing Values. While in FLEX1 command mode it is possible to ask for the defined value of several variables. These include the initial x values, the initial memory values, b and r parameter values, etc. (A complete list is included in Appendix Section 1.) The command form is

LIST=__

The blank is replaced with that variable whose value is desired. For example, LIST=B(2) (CR) would cause the value of b_2 to be printed. Thus, the value of a parameter may be checked while in FLEX1 command mode.

H. Simulate. After the above information has been entered into FLEX1, the command

SIMULATE

will begin the simulation run. This command freezes the current parameter values for that run, so that they may be changed only by using the CLEAR command.

I. Repeated Runs. When a simulation run has been completed, FLEX1 will type

END OF SIMULATION RUN

ENTER INPUTS/COMMANDS

followed by a command glitch. Use of FLEX1 may be terminated by typing

STOP

If an additional run is desired, enter

CLEAR

This command will reset all FLEX1 variables to their pre-use defined values. In effect, it is as if the user typed STOP and then made a call to FLEX1. The system will type

[FLEX1 Heading]

ENTER INPUTS/COMMANDS

followed by a command glitch. Now the user may input his new information. The same order and types of inputs are necessary. Obviously, many runs can be made without ever leaving FLEX1 if all the parameter files and overlays needed have been made prior to the first call.

Editing

FLEX1 uses three symbols for editing of input. These are the backslash (\), the at sign (@), and the back arrow (←). The use of each will now be detailed.

Often the charcter typed is not the one which the user intended. The backslash operates as a back spacer. Each time it is used, a character is deleted from the line or character string. This deletion is done in a consecutive manner. For example

```
INN\PUU\T-\=*LII\DEI1
```

would be read, by FLEX1, as

```
INPUT=*LIDEI1
```

Another example is

FOANL\\\\UNLOAD=*LIDEØ1

The first backslash deletes the L, the second the preceding N, the third the preceding A and so forth. The backslash here operates exactly the same as the backslash in the OS-3 EDITOR.

Sometimes so many mistakes have been made that it is desirable to begin the line again. There are two ways to do this. The first way uses the <u>at</u> sign (@). This operates for FLEX1 exactly the same way that it does for the OS-3 EDITOR. That is, the <u>at</u> sign deletes all characters preceding it. After typing this symbol, follow it with the line as it should have appeared. For example,

N-\=12 TRNTL@NUMBER=12 TSTART=1Ø TMAX=15 (CR)

The second way to begin again involves the use of the back arrow. The back arrow deletes the entire line in which it appears and a new line is started by a FLEX1 command glitch. The back arrow should always be followed immediately by a carriage return (CR).

The length of a line in FLEX1 is limited to 136 characters including spaces and editing symbols. If the user has entered part of a line and decides to start over, he should use the <u>at</u> sign if fewer than, say, 50 characters have been entered, and the back arrow if more.

<u>Manual Interrupt</u>

It is possible to interrupt FLEX1 after the SIMULATE command has been given. Depress the BREAK key and type MI after the OS-3 system returns the pound sign. After this command, control is returned to the FLEX1 system which finishes that section of output interrupted when BREAK was hit, and returns to its internal command mode typing

INTERRUPTED

ENTER INPUTS/COMMANDS

followed by the command glitch.

Output specifications may now be changed, including monitor specifications. In the Appendix on FLEX1 Commands is a complete list of those variables which may be altered after interrupting FLEX1 without having to type CLEAR. Of course, if CLEAR is entered all information must be input to FLEX1 again.

Output Specification

A. The Teletype Monitor Listing. It is advisable to make a summary on the teletype of the first FLEX1 simulation of a model. This is useful in case any debugging of the model functions is necessary. Since this output slows FLEX1 down quite a bit, as soon as summaries are no longer needed at the teletype, they should be deleted (NOSUMMARY).

The more variables monitored, the slower the simulation run. Only monitor the most crucial variables. If no monitoring is needed, and many runs are to be made, a great deal of time may be saved by not monitoring and just using the line printer and dump file output.

B. The Line Printer Listing. Ordinarily, all of the y output variables should be sent to the line printer with the YMAX command (see PROGRAMMING IN FLEX1 section on output specifications). This is the complete output of that FLEX1 run and records the changes in the output variables through time. This listing should be used for model examination. The same LPLUN can be specified for consecutive runs.

C. The Data Dump File. If satellite program examination of the output is desired a dump file should be made. All specified output will be sent to the lun entered by the DUMPLUN command. As noted previously, this lun should be equipped to a file prior to the first call to FLEX1.

If several consecutive runs are made, the same DUMPLUN can be used each time. FLEX1 will output the model results to that lun in the sequential order that the runs are made, each output block separated from the next by a file mark. Putting several runs on one file helps save storage room and makes no difference regarding any file manipulations.

After the last run has been made on FLEX1, the user, while in OS-3 control mode, should save the dump file by some name. For example, if the command DUMPLUN=1Ø had been used, then prior to calling FLEX1, he should enter

```
#EQUIP, 1Ø=FILE
```

and after the STOP command and return to OS-3 control mode, enter, for example,

```
#SAVE, 1Ø=*LIDED1 .
```

The dump file should always be saved under a public name so that it may be accessed from other job numbers. Remember to file protect all public files.

Data Dump File Manipulations by Satellite Programs

After the dump file has been saved, it may be used to produce line printer plots or to generate line printer listings. There are two line printer plot programs and each will be discussed separately. The dump file listing program will also be detailed.

A. Satellite Program *PLOTCON. This program produces line printer plots from a FLEX1 dump file with plot specifications entered conversationally from the teletype. The calling procedure is

```
#EQUIP, 1=<dump file name>
#*PLOTCON
```

The first question is "INSTRUCTIONS?" If instructions are needed, type "YES". The program will explain what information is requested by the program.

At the end of the program execution, the logical unit number (<lun>) of the plots are printed. Two copies of the plots are usually made. The following procedure is used:

```
#EQUIP, 1∅=LP
#LABEL, 1∅/SAVE FOR <user identification>
#DATE, 1∅
#COPY, I=<plot lun>/R,O=1∅, S=∅
#COPY, I=<plot lun>/R,O=1∅, S=∅
```

Some lun other than 1∅ may be used for the line printer. Substitute the other lun for every appearance of 1∅.

<u>B. Satellite Program *LPLOT</u>. This program produces line printer plots from a FLEX1 dump file and a plot specification file. The calling procedure is:

```
#EQUIP, 1=<dump file name>
#EQUIP, 2=<specification file name>
#*LPLOT
```

The program prints a model title, then asks "PLOT?" If a plot is desired, type "YES"; if not, type "NO." When the first plot is indicated the program reads information from the specification file and plots the data. For subsequent plots, in addition to asking "PLOT?", the program asks "SAME SPECS?" If the specifications desired for a plot are the same as the previous plot, type "YES." The program will process the plot. If specifications are different, type "NO" and the program will read a set of information from the specification file.

Specification File

1. Title Card	Col.	1-80	identifying title
2. Specification Card	Col.	1-2	number of variables selected for plotting $(1 \leq n \leq 10)$
		3-6	starting time
		7-10	time step multiple
		11-14	stopping time
3. Variable card(s) (use as many as number of selected variables)	Col.	1-2	position in dump file
		3-10	variable identifying alpha name (1st character of name is plotting symbol)
		11-20	maximum value
		21-30	minimum value

The time variables can be used to select a subset of the data. Rather than starting at the first time step on the dump file, a later starting time may be specified. The stopping time can be used similarly to cut off a plot before the end of the dump file. The time step multiple (n) is used to skip records; every nth record is plotted. Note that this refers to records on the dump file. That is, if these records are for every fifth time step and if the plot time step multiple is two, then every other record will be plotted, i.e., every tenth time step.

C. Satellite Program *DMPRNT. If a line printer listing of a simulation run is desired from the dump file, it may be generated using the data dump file and the FLEX1 Satellite *DMPRNT program. This is run from a teletype as follows.

While in OS-3 command mode enter

```
#EQUIP, 1Ø=<name of FLEX1 dump file>, 6=LP
#LABEL, 6/SAVE FOR <user's name>
#REWIND, 1Ø
#MFBLKS=1ØØØ
#LOAD, *DMPRNT
RUN
```

The dump file will now be read and a copy of it sent to the line printer and saved at the Computer Center under the user's name.

REFERENCES

Control Data Corporation Computer Systems Fortran Reference Manual, CDC 3100, 3200, 3300, and 3500; 1966. (Especially Chapter 7.)

Freeman, H. 1965. Discrete-Time Systems. John Wiley, New York.

Klir, G.J. 1969. An Approach to General Systems Theory. Van Nostrand Reinhold, New York.

Overton, W.S. 1972. "Toward a general model structure for a forest ecosystem." In Proceedings - Research on Coniferous Forest Ecosystems - a symposium. Franklin, J.F., L.J. Dempster and R.H. Waring (Eds.), U.S. Government Printing Office, stock number 0101-0233.

Overton, W.S. 1973. The Ecosystem Modelling Approach in the Coniferous Biome. (presented) Symposium on Ecosystem Modelling, Athens, Georgia. March, 1973. (In Press, edited by B.C. Patten)

OS-3 Editor Manual, (by Fred Dayton). O.S.U. Computer Center Publication ccm-70-7(R); 1971.

OS-3 Reference Manual, O.S.U. Computer Center Publication ccm-70-8R; 1971.

1. A SUMMARY OF COMMANDS

The following is a summary of the commands introduced in Sections III and IV above. The command is accompanied with a short description, example and default values. In addition, symbols are used to indicate if the command can be used in conjunction with the LIST command (*) or if the value may not be changed after the command SIMULATE without use of the command CLEAR (+).

The commands are listed in the order in which they should be used. A nested structure is used so that, for example, after the INPUT command is listed, the various files (I-file, B-file, etc.) to which it applies have their commands listed, also in the correct order.

COMMAND	DESCRIPTION AND EXAMPLE	DEFAULT	LIST	CHANGE
NUMBER=_	Number of x's. NUMBER=1Ø	abort	* use N	+
TSTART=_	Initial time value. TSTART=23	0	*	+
TMAX=_	Termination time value. TMAX=52	none	*	
FUNLOAD=<name>	Load the function overlay saved under <name>. FUNLOAD=*LIDEØ1	abort		

INPUT=<name>
=<lun>
	Read commands from <name> or <lun>.
	INPUT=*LIDEI3
	INPUT=6Ø
		lun=60		

LAG=_	Maximum delay of any x and/or z. LAG=1Ø	0	*	

COMMAND	DESCRIPTION AND EXAMPLE	DEFAULT	LIST	CHANGE
LAGX( )=_,_,... LAGZ( )=_,_,...	Indices of lagged x or z variables. LAGX( )=1,3,5 LAGZ( )=12 LAGX( )=must precede LAGZ( )=.	none none		
XD(,)=_,_,... ZD(,)=_,_,...	Initial value of the delayed x or z. XD(5,)=54,67,93 ZD(12,1)=14	undefined undefined	* *	+ +
XL( )=_,_,...	Lower limit for x. If passed, x is reset to given value. XL(3)=-1	0		
XU( )=_,_,...	Upper limit for x. If exceeded, run ends. XU( )=1,Ø,2ØØ	10^{100}		
XE( )=_,_,...	Change norm. If $\|\Delta_i(k)\|$>XE(i) and XE(i)≠Ø, the run ends. If XE(i)=Ø, no check. XE(4)=1.2	no check		
YMAX=_	Number of y's dumped to the dump file. YMAX=15	dump all x's	*	
TTYPRT=_	Monitor print interval. TTYPRT=5	1/50 run	*	
TTY=_	Variables to be monitored. TTY=Y(1) TTY=F(1,2)			
LPRT=_	Line printer interval. LPRT=3	1	*	
LP=_	Variables to be line-printed. LP=Y(1) LP=Y(7)	none		
LPLUN=<lun>	Line printer lun. LPLUN=44	no output	*	

COMMAND	DESCRIPTION AND EXAMPLE	DEFAULT	LIST	CHANGE
DPRT=_	Dump file print interval. DPRT=5	1	*	
D=_	Variables to be dump filed. D=Y(1) D=Y(1Ø)	all x's if YMAX not specified		
DUMPLUN=<lun>	Dump file lun. DUMPLUN=4Ø	no dump	*	
B( )=_,_,...	Constant parameters. B(1)=7 B(8)=9 B( )=1,3,1Ø,4	0	*	+
R( )=_,_,...	Constant parameters. R(1)=3. R(2)=4.79 R( )=4.3ØØ,1,7.2	0	*	+
XN( )=_,_,...	Initial values of x. XN(3)=5.2 XN( )=7.5	0	* use X	+
TITLE=_	Identification for run. TITLE=MODEL RUN 17	none		
LOG NOLOG	Line printer listing is being made (LOG) or not (NOLOG). LOG	LOG		
SUMMARY NOSUMMARY	Summary to be printed on the monitor (SUMMARY) or not (NOSUMMARY). NONSUMMARY	SUMMARY	*	

LIST=_	Print current value of item. Item may be any in this section with an asterisk (*) in the list column.
	LIST=N
	LIST=X(3)
		none		

COMMAND	DESCRIPTION AND EXAMPLE	DEFAULT	LIST	CHANGE
SIMULATE	Begin simulation run. SIMULATE	none		
CLEAR	Reset to original entry condition. CLEAR	none		
STOP	Simulation session is terminated. STOP	none		

2. FORMAT SUMMARY

Z Functions

```
                          SUBROUTINE ZCOMP (K, X, B, R, Z)
                          DIMENSION X(1), B(1), R(1), Z(1)
                        { .
                        { .
                        { .
Calculate               {
z values                { Z(i)=expression
                        {
                        { .
                        { .
                        { .
                          RETURN
                          END
```

G Functions

```
                          FUNCTION GØ1 (K, X, B, R, Z)
                          DIMENSION X(1), B(1), R(1), Z(1)
                        { .
Calculate               { .
value of g1             { .
                        { GØ1=expression
                          RETURN
                          END
```

Note: If G's are used by other functions, reference them as G(Ø1), ..., G(7Ø) (with parentheses).

F Functions

Calculate value of $f_{8,11}$

```
FUNCTION FØ811 (K, X, B, R, Z)
DIMENSION X(1), B(1), R(1), Z(1)
.
.
.
FØ811 = expression
RETURN
END
```

S Functions

See CDC Computer Systems Fortran Reference Manual.

Y Functions

Calculate y values

```
SUBROUTINE YCOMP (K, X, B, R, Y)
DIMENSION X(1), B(1), R(1), Y(1)
.
.
.
Y(i) = expression
.
.
.
RETURN
END
```

I-file

```
     LAG=__
     LAGX=__,__, ...
     LAGZ=__,__, ...
     XD( , )=__,__, ...
     ZD( , )=__,__, ...
     XL( )=__,__, ...
     XU( )=__,__, ...
     XE( )=__,__, ...
     YMAX=__
     TTYPRT=__
     TTY=_
     LPRT=__
     LP=__
     LPLUN=<lun>
     DPRT=__
     D=__
     DUMPLUN=<lun>
     INPUT=60
```

B-file

```
     B( )=__,__, ...
     INPUT =60
```

R-file

```
    R( )=__,__, ...

    INPUT=60
```

N-file

```
    XN( )=__,__, ...

    INPUT=60
```

General Run Commands

```
    NUMBER=__     TSTART=__     TMAX=__     FUNLOAD=<name>

    INPUT=<I-file name>

    INPUT=<B-file name>

    INPUT=<R-file name>

    INPUT=<N-file name>

    TITLE=______________________________

    LOG or NOLOG

    SUMMARY or NOSUMMARY

    SIMULATE
```

3. ERROR MESSAGES

This is an exhaustive list of FLEX1 error messages and their meanings. VSYM represents an internal variable symbol of FLEX, BASESYM represents any command keyword (i.e. CLEAR, XD(,), XN( ), etc.) and SYM represents any symbol (i.e. misspelled commands, etc.).

1. "Error on VSYM while processing VSYM invalid definition of array bounds in VARDEF." This indicates an error in the FLEX1 processor and should be brought to the attention of one of the authors for appropriate action.
2. "Error on VSYM while processing VSYM error in VARDEF -- invalid type." FLEX1 error. See 1.
3. "Error on SYM while processing BASESYM:
   a. symbol not defined." FLEX1 was expecting another command but the new SYM was not a command. Recover by re-entering entire command.
   b. invalid symbol." See 3a.
   c. invalid character after symbol." In some cases an '=' sign is needed after a command and this did not appear. Or parentheses were mismatched. Recover by re-entering entire command.
   d. non-integer subscript." An integer subscript was expected (i.e. X(8)), but something else occurred. Recover by re-entering entire command.
   e. subscript exceeds array bounds." A subscript was larger

than the maximum allowed (i.e. X(84)). Recover by re-entering entire command.

f. value must be integer." SYM was probably not a number. Recover by re-entering entire command.

g. value must be real." SYM was probably not a number. Recover by re-entering only the quantity(ies) on the right side of the '=' sign.

h. variable cannot be specified again." Some commands cannot be specified twice without using CLEAR. SYM is the erroneous command. This message occurs only if another command has been processed between the two specifications (i.e. you may change NUMBER if you do so immediately after the first entry). Recover by entering CLEAR.

i. value must be a symbol." FLEX1 does not recognize the SYM because it has a digit as its first character. Recover by re-entering entire command.

j. end of file - processing not complete." An end of file was reached before completing the processing of a command. Recover by re-entering quantity(ies) on the right side of the '=' sign.

k. character string too long." This type of SYM must be 8 characters or less. Recover by re-entering entire command.

l. equal sign not found." The next character after a command was not an '=' sign. Recover by re-entering entire command.

m. invalid definition of array bounds in VARDEF." See 1.

n. another variable must be specified first." To process the command another command must precede it, e.g. NUMBER must be specified prior to x limits, initial x's, or any memory specification; LAG, prior to XD or ZD. Recover by re-entering entire command.

o. unable to equip saved file." There was no saved file with the given name or it was a public file and someone else was using it. Unequip a lun from 1 to 50 or try again later.

p. number of subscripts specified as invalid." Either subscripts were used with a non-subscripted command or the wrong number of subscripts were indicated (too many or too few commas). Recover by re-entering entire command.

4. "Error on print spec, SYM ignored." The variable requested cannot be selected for output, or the subscript is non-numeric or less than 1, or there is an invalid number of subscripts specified. This print request is ignored.

5. "Too many print selections." More than 79 variables have been selected for output. Only the first 79 will be used.

6. "Invalid lun - SYM ignored." If LP lun was not between 1 and 50, no log (LP listing) could be made. If dump was not between 1 and 50, but was equipped, the run would abort. Recover by re-assigning luns.

7. "Invalid function file." An empty file, a busy public file or a non-binary (uncompiled) function file was used. Recover by re-entering entire command.

8. "Invalid X dimension." NUMBER was read as 0 or negative or greater than 63. Recover by immediately re-entering entire command.

9. "Too many past values specified." Caused by memory block overflow (i.e., total number of variables in memory times maximum past values retained is greater than 200). Recover by entering CLEAR.

10. "Error condition prevails -- enter CLEAR command." Self-explanatory.

11. "Abnormal termination of modelling run." Some non-recoverable error has occurred. Recover by starting afresh.

12. "Error." An error has occurred. Precedes further error clarification.

13. "X( )=SYM in model number 1." X( ) has decreased below its lower limit and been reset to that lower limit. The model number is irrelevant in FLEX1.

14. "X( ) is out of range." X( ) has increased above its upper limit. The simulation run is terminated.

15. "Change in X( ) exceeds limit." Δ( ) exceeded the error limits. The simulation run is terminated.

16. "F and G functions too deeply nested." FLEX1 keeps track of the nesting of functional dependence. Although the g functions need not be indexed in numerical order, so that g_i never references a g_j $j \leq i$, it must be possible to construct such an order. If nesting of functional dependence exceeds 50 or if a function references itself using parentheses, this message will occur. Recover by changing functions and starting over.

17. "Variable not valid for LIST command." Only those variables and commands in the Appendix Section 1 command summary are valid. Recover by re-entering entire command.

For the most part, FLEX1 error messages are self-explanatory. The major difference lies between the instruction

REENTER

and the instruction

REENTER VALUE

In the first case the command, equal sign, and the value must be typed again (as well as anything which followed that command on the same input line). In the second case, only what was entered on the right side of the equal sign need be re-entered.

If several error messages are printed in a row, hit the break key and start over. When in doubt, enter CLEAR and begin again.

*FLOAD error messages.

There are two messages used by *FLOAD. The first

INVALID OR NO F PARAMETER

indicates that either (a) no F=<name> command appeared in the parameter string or (b) no lun is available. The solution to (a) is obvious. If this was included, then unequip some lun between 1 and 10 and try again. This seldom occurs.

The second error message is

TOO MANY FUNCTIONS OR NO OVERLAY NAME SPECIFIED

If more than 70 g functions or 63 f functions were specified, this will occur. If no name was used in the F=<name> command this will also occur. (This is needed since F=<lun> can slip past (a) of the first message, but a <name> is required.)

MODEL II

FLEX MODEL OF: Overton Hydrology Model (Revised)

INVESTIGATORS: Overton, White

DATE: 7/11/73

RESOLUTION: Daily

QUANTITY: Water equivalents (m^3/ha)

VARIABLES AND FUNCTIONS

1. X LIST

	Description
x_1	Canopy storage (m^3/ha)
x_2	Snow storage (m^3/ha)
x_3	Soil Water (m^3/ha)
x_4	Ground water (m^3/ha)
x_5	Stream flow ($m^3/ha/day$)
x_6	Dummy — observed stream flow ($m^3/ha/day$)
x_7	Dummy — output $\Sigma(g_{15}-z_3)^2$
x_8	Dummy — output $\Sigma(\ln g_{15} - \ln z_3)^2$
x_9	Daily transpiration (m^3/ha)
x_{10}	Monthly transpiration (m^3/ha)
x_{11}	Yearly transpiration (m^3/ha)
x_{12}	Daily evaporation (m^3/ha)
x_{13}	Monthly evaporation (m^3/ha)
x_{14}	Yearly evaporation (m^3/ha)
x_{15}	Daily precipitation (m^3/ha)
x_{16}	Monthly precipitation (m^3/ha)
x_{17}	Yearly precipitation (m^3/ha)
x_{18}	Monthly comp. stream flow ($m^3/ha/month$)
x_{19}	Yearly comp. stream flow ($m^3/ha/yr$)
x_{20}	Monthly obsr. stream flow ($m^3/ha/month$)
x_{21}	Yearly obsr. stream flow ($m^3/ha/yr$)

2. Z FUNCTIONS

Description

$$z_1 = \left\{ \begin{array}{ll} S2(k) & , \; S3(k)>38 \\ (.1667)(S3(k)-32)S2(k) & , \; 32\le S3(k)\le 38 \\ 0 & , \; S3(k)<32 \end{array} \right\}$$ Precipitation as rain (m^3/ha)

$$z_2 = \left\{ S2(k)-z_1 \right\}$$ Precipitation as snow

$$z_3 = \left\{ S5(k) \right\}$$ Observed stream flow $(m^3/ha/day)$

$$z_4 = \left\{ b_9 z_1 \right\}$$ Precip. direct to forest floor (m^3/ha)

4. G FUNCTIONS

$$g_1 = \left\{ (b_{10}-x_1)(1-e^{b_{11} z_1 \Delta t}) \right\}$$ Incremental input to canopy (m^3/ha)

$$g_2 = \max \left\{ \begin{array}{l} S1(k)+1 - (S1(k)+1)^{z_1/b_{12}} \\ 0 \end{array} \right\}$$ Adjusted potential evapotranspiration (m^3/ha)

$$g_3 = \min \left\{ \begin{array}{l} x_1+g_1 \\ g_2(1-e^{-b_{13}(x_1+g_1)}) \end{array} \right\}$$ Evaporation from canopy storage (m^3/ha)

$$g_4 = \left\{ \begin{array}{ll} 0 & , \; x_3<b_1 \\ (\frac{g_2-g_3}{b_2-b_1})(x_3-b_1) & , \; b_1\le x_3\le b_2 \\ g_2-g_3 & , \; x_3>b_2 \end{array} \right\}$$ Transpiration (m^3/ha)

$$g_5 = (1-b_9)z_1-g_1$$ Drip (m^3/ha)

$$g_6 = \max \left\{ \begin{array}{l} (S3(k)-32)(254\ RAD(S4(k,1))+.014(z_4+g_5)) \\ 0 \end{array} \right\}$$ Potential snow melt (m^3/ha)

$$g_7 = \min \left\{ \begin{array}{l} z_2+x_2 \\ g_6 \end{array} \right\}$$ Actual snow melt (m^3/ha)

$$g_8 = \max \left\{ \begin{array}{l} (1-e^{-b_3 \Delta t})(x_3-b_5) \\ 0 \end{array} \right\}$$ Amount of x_3 available for flow (m^3/ha)

$$g_9 = \min \left\{ \begin{array}{l} b_8-x_4 \\ g_8 \end{array} \right\}$$ Percolation from x_3 to x_4 (m^3/ha)

$$g_{10} = g_8-g_9$$ x_3 lateral flow (m^3/ha)

$g_{11} = \max \begin{Bmatrix} (1-e^{-b_4\Delta t})(x_4-b_6) \\ 0 \end{Bmatrix}$ x_4 lateral flow (m^3/ha)

$g_{12} = z_4+g_5+g_7$ Possible infiltration (m^3/ha)

$g_{13} = \min \begin{Bmatrix} b_7-x_3 \\ g_{12} \end{Bmatrix}$ Actual infiltration (m^3/ha)

$g_{14} = \max \begin{Bmatrix} g_{12}-g_{13} \\ 0 \end{Bmatrix}$ Surface runoff (m^3/ha)

$g_{15} = (g_{10}+g_{11}+g_{14})/\Delta t$ Stream input (m^3/ha/day)

5. <u>F FUNCTIONS</u> <u>Description</u>

$f_{1,1} = g_1-g_3$ Input, evap. (m^3/ha)

$f_{2,2} = z_2-g_7$ Snow, melt (m^3/ha)

$f_{3,3} = g_{13}-g_4-g_8$ Infiltration, trans, outflow (m^3/ha)

$f_{4,4} = g_9-g_{11}$ Percolation, lateral flow (m^3/ha)

$f_{5,5} = g_{15}-x_5$ Update (m^3/ha/day)

$f_{6,6} = z_3-x_6$ Update (m^3/ha/day)

$f_{7,7} = \{(g_{15}-z_3)^2\}$ Dummy

$f_{8,8} \{ [\ln \{\max (10^{-7}, g_{15})\} - \ln \{\max (10^{-7}, z_3)\}]^2 \}$ Dummy

} Non-dimensional

$f_{9,9} = g_4-x_9$ Daily transpiration (m^3/ha)

$f_{10,10} = \begin{Bmatrix} g_4 & , \text{ otherwise} \\ g_4-x_{10} & , \text{ S4(k,0)} \neq \text{(k,1)} \end{Bmatrix}$ Monthly transpiration (m^3/ha)

$f_{11,11} = \begin{Bmatrix} g_4 & , \text{ otherwise} \\ g_4-x_{11} & , \text{ S4(k,0) 6 and S4(k,1)=7} \end{Bmatrix}$ Yearly transpiration (m^3/ha)

$f_{12,12} = g_3-x_{12}$ Daily evaporation (m^3/ha)

$$f_{13,13} = \begin{cases} g_3 & \text{, otherwise} \\ g_3 - x_{13} & \text{, } S4(k,0) \neq S4(k,1) \end{cases}$$ Monthly evaporation (m^3/ha)

$$f_{14,14} = \begin{cases} g_3 & \text{, otherwise} \\ g_3 - x_{14} & \text{, } S4(k,0)=6 \text{ and } S4(k,1)=7 \end{cases}$$ Yearly evaporation (m^3/ha)

$$f_{15,15} = z_1 + z_2 - x_{15}$$ Daily precipitation (m^3/ha)

$$f_{16,16} = \begin{cases} z_1 + z_2 & \text{, otherwise} \\ z_1 + z_2 - x_{16} & \text{, } S4(k,0) \neq S4(k,1) \end{cases}$$ Monthly precipitation (m^3/ha)

$$f_{17,17} = \begin{cases} z_1 + z_2 & \text{, otherwise} \\ z_1 + z_2 - x_{17} & \text{, } S4(k,0)=6 \text{ and } S4(k,1)=7 \end{cases}$$ Yearly precipitation (m^3/ha)

$$f_{18,18} = \begin{cases} g_{15} & \text{, otherwise} \\ g_{15} - x_{18} & \text{, } S4(k,0) \neq S4(k,1) \end{cases}$$ Monthly computed stream flow (m^3/ha/month)

$$f_{19,19} = \begin{cases} g_{15} & \text{, otherwise} \\ g_{15} - x_{19} & \text{, } S4(k,0)=6 \text{ and } S4(k,1)=7 \end{cases}$$ Yearly computed stream flow (m^3/ha/yr)

$$f_{20,20} = \begin{cases} z_3 & \text{, otherwise} \\ z_3 - x_{20} & \text{, } S4(k,0) \neq S4(k,1) \end{cases}$$ Monthly observed stream flow (m^3/ha/month)

$$f_{21,21} = \begin{cases} z_3 & \text{, otherwise} \\ z_3 - x_{21} & \text{, } S4(k,0)=6 \text{ and } S4(k,1)=7 \end{cases}$$ Yearly observed stream flow (m^3/ha/yr)

6. SPECIAL FUNCTIONS

	Description
S1(k) = CP(S4(k,1))S3(k)	Potential evapotranspiration (m^3/ha) (CP = m^3/ha/°F)
S2(k) = Tabulated data	Precipitation (m^3/ha)
S3(k) = Tabulated data	Temperature (°F)
S4(k,m) = Tabulated data	Month that day k+m is in (1 to 12)
S5(k) = Tabulated data	Observed stream flow (m^3/ha/day)

7. Y FUNCTIONS

Description

$y_1 = x_1$

$y_2 = x_2$

$y_3 = x_3$	
$y_4 = x_4$	
$y_5 = x_5$	
$y_6 = x_6$	
$x_7 = x_7$	
$y_8 = x_8$	
$y_9 = x_9$	(m^3/ha/day)
$y_{10} = x_{10}$	(m^3/ha/month)
$y_{11} = x_{11}$	(m^3/ha/yr)
$y_{12} = x_{12}$	(m^3/ha/day)
$y_{13} = x_{13}$	(m^3/ha/month)
$y_{14} = x_{14}$	(m^3/ha/yr)
$y_{15} = x_{15}$	(m^3/ha/day)
$y_{16} = x_{16}$	(m^3/ha/month)
$y_{17} = x_{17}$	(m^3/ha/yr)
$y_{11} = x_{18}$	(m^3/ha/month)
$y_{19} = x_{19}$	(m^3/ha/yr)
$y_{20} = x_{20}$	(m^3/ha/month)
$y_{21} = x_{21}$	(m^3/ha/yr)

PARAMETERS

8. <u>B PARAMETERS</u>

List	Value	Description
b_1	1.82×10^3 m^3/ha	Wilting point
b_2	2.96×10^3 m^3/ha	Transpiration resistance point
b_3	1.48 day^{-1}	x_3 flow rate

b_4	3.75 day^{-1}	x_4 flow rate
b_5	2.96×10^3 m^3/ha	x_3 retention capacity
b_6	9.97×10^3 m^3/ha	x_4 retention capacity
b_7	4.59×10^3 m^3/ha	x_3 storage capacity
b_8	11.9×10^3 m^3/ha	x_4 storage capacity
b_9	.25	Prop. of rain direct to forest floor
b_{10}	100 m^3/ha	Max. canopy storage
b_{11}	.0075 $ha/m^3/day$	$b_{11} = \frac{(1-b_9)}{b_{10}}$
b_{12}	762 m^3/ha	Assume g_2 = 0 with 3" = 7.62 cm rain
b_{13}	.3 ha/m^3	Assumed evaporation rate

10. <u>IC (N) INITIAL CONDITIONS</u>

	List	Value	Description (see X List)
XN:	x_1	0	
	x_2	0	
	x_3	2.3114×10^3 m^3/ha	
	x_4	10.97×10^3 m^3/ha	
	x_5	0	
	x_6	0	
	x_7	0	
	x_8	0	
	x_9	0	
	x_{10}	0	
	x_{11}	0	
	x_{12}	0	
	x_{13}	0	

x_{14}	0
x_{15}	0
x_{16}	0
x_{17}	0
x_{18}	0
x_{19}	0
x_{20}	0
x_{21}	0

GENERAL RUN INFORMATION

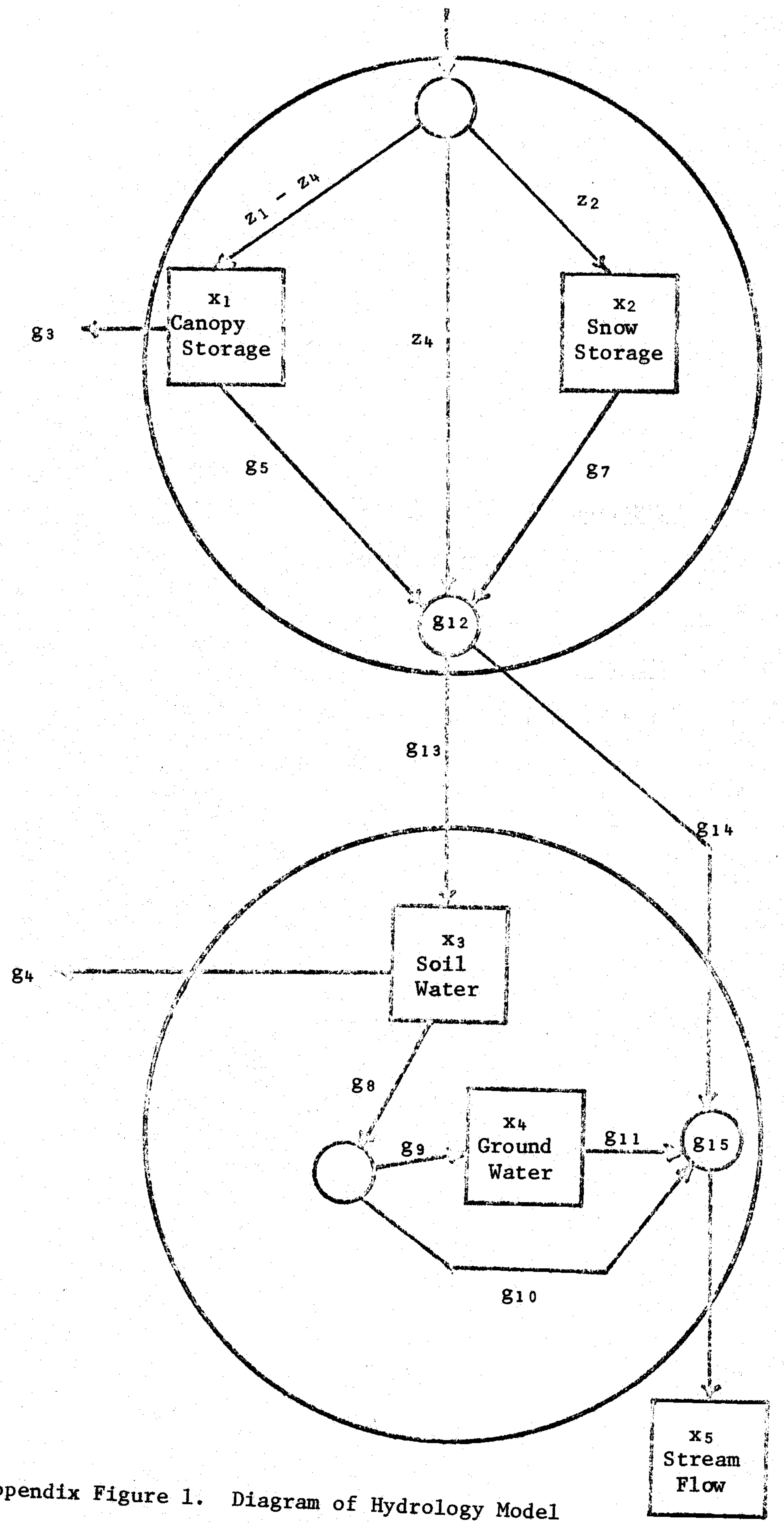
TSTART = 0 (July 1, 1958)

TMAX = 731

Dump, LP: all y's

Plot: y_i , $i = 1, \ldots, 6$

No monitor



Appendix Figure 1. Diagram of Hydrology Model

Comments

This model is the second in the series of hydrology models worked on by the Oregon Central Modelling Project. It represents the first reworking of the hydrology model developed by Paul Riley and associates at Utah State University and reported as Biome Internal Report 52.

Several features distinguished Model II from its predecessor. The order of model events and couplings have been made explicit. Canopy charge has been made a function of present canopy storage and the current amount of precipitation. Atmospheric demand for water has been made a function of precipitation, allowing for atmospheric saturation. Water intercepted by the canopy which does not add to canopy storage has been defined as canopy drip to the forest floor. In addition, thrufall direct to the forest floor has been distinguished from canopy intercepted precipitation. Melted snow now flows direct to forest floor. Lastly, the model has been parameterized to Watershed 10 physical characteristics.

Behavior

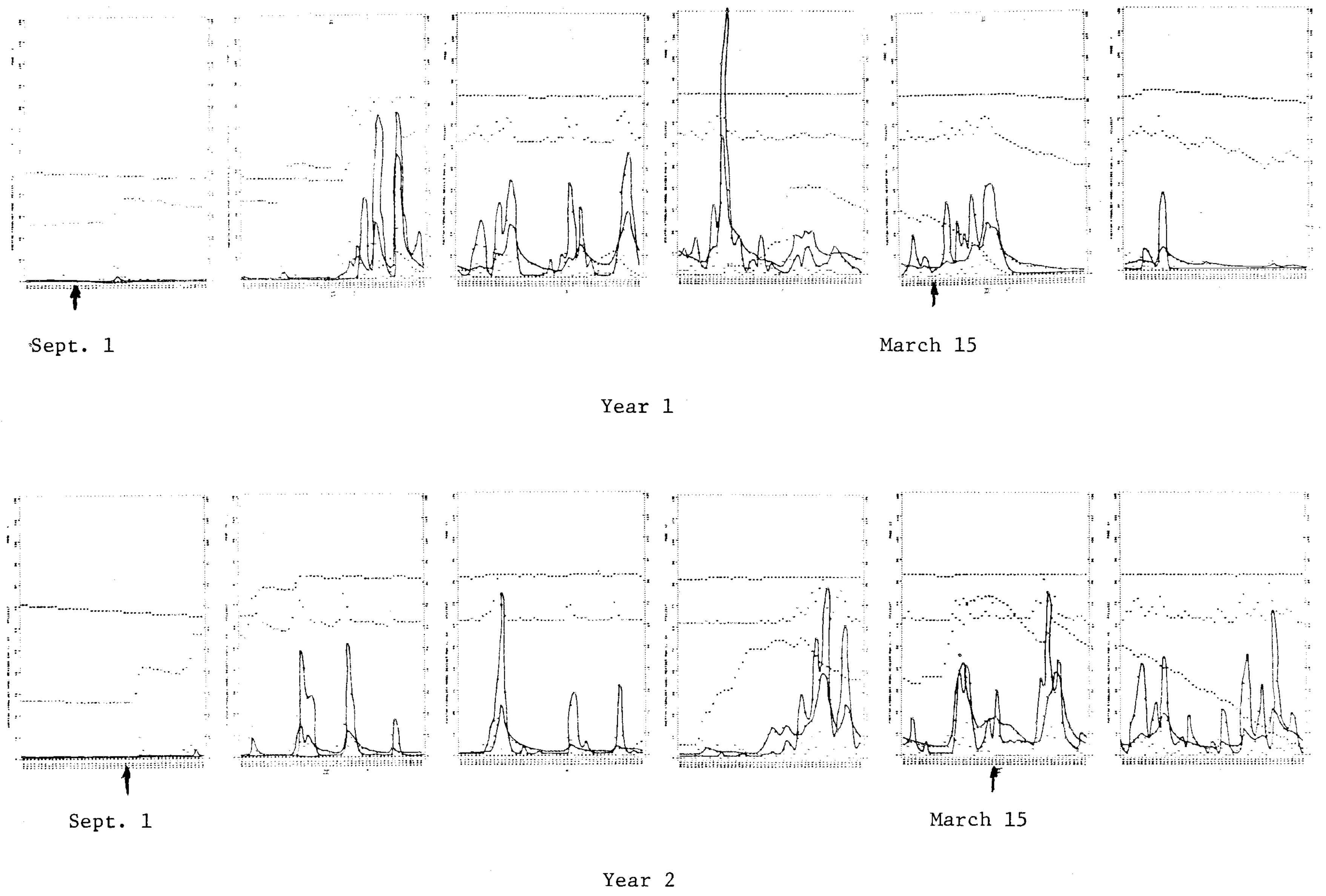
The mass balance is changed, most noticeably by the reversal in relative magnitude between evaporation and transpiration. This is a change in the wrong direction.

There is little change in objective function one, but there is a dramatic improvement in objective function two (two orders of magnitude). This is due to the maintenance of some flow throughout the summer.

Flow peaks match a little better than before. However, the charge and discharge of the stream is still too rapid. Also, Model II does not catch the early fall recharge peaks.

Several runs were made with different storage capacities for ground water. This did not noticeably change behavior.

A run was made with the transpiration resistance point changed from equality with field capacity to wilting point plus one-fifth the difference between field capacity and wilting point (i.e., w.p. + 1/5 (f.c. - w.p.)). This increased transpiration by about 10%, but did not change the relative magnitude of evaporation and transpiration. Ground water was depleted to wilting point more rapidly than before.



Appendix 2 Figure 2. Model Behavior

NOTE:

In all hydrology models two sets of data, with monthly values, are used. The first, identified as RAD in the potential snow melt equation, expresses the effect of radiation on melting snow. The second, identified as CP in the special function S1, expresses the atmospheric demand for water (i.e. potential evapo-transpiration) depending on air temperature. Values are given in the table below.

	RAD (inches)	CP (inches/°F)
January	.0208	.00029
February	.0184	.00075
March	.0136	.00181
April	.0088	.00233
May	.0040	.00329
June	.0024	.00340
July	.0028	.00416
August	.0068	.00381
September	.0112	.00290
October	.0168	.00152
November	.0192	.00057
December	.0212	.00029

5. Sample Program with Runs

```
      SUBROUTINE ZCOMP(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      Z(1)=0.
      Z(3)=S5(K)
      IF (S3(K) .GT. 38.) Z(1)=S2(K)
      IF (S3(K) .GE. 32. .AND. S3(K) .LE. 38.) Z(1)=
     1.1667*S2(K)*(S3(K)-32.)
      Z(2)=S2(K)-Z(1)
      Z(4)=B(9)*Z(1)
      RETURN
      END
      FUNCTION G01(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G01=(B(10)-X(1))*(1.-EXP(-B(11)*Z(1)))
      RETURN
      END
      FUNCTION G02(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G02=0.
      IF (Z(1) .GE. B(12)) RETURN
      G02=S1(K)+1.-(S1(K)+1.)**(Z(1)/B(12))
      RETURN
      END
      FUNCTION G03(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G03=G(2)*(1.-EXP(-B(13)*(X(1)+G(1))))
      IF (G03 .GT. X(1)+G(1)) G03=X(1)+G(1)
      RETURN
      END
      FUNCTION G04(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      IF (X(3) .LT. B(1)) G04=0.
      IF (X(3) .GT. B(2)) G04=G(2)-G(3)
      IF (X(3) .GE. B(1) .AND. X(3) .LE. B(2)) G04=
     1(G(2)-G(3))*(X(3)-B(1))/(B(2)-B(1))
      RETURN
      END
      FUNCTION G05(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G05=(1.-B(9))*Z(1)-G(1)
      RETURN
      END
      FUNCTION G06(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      DIMENSION RAD(12)
      DATA ((RAD(I),I=1,12)=.0208,.0184,.0136,.0088,.0040,.0024,
     1.0028,.0068,.0112,.0168,.0192,.0212)
      M=S4(K,1)
      G06=(S3(K)-32.)*(254.*RAD(M)+.014*(Z(4)+G(5)))
      IF (G06 .LT. 0.) G06=0.
      RETURN
      END
      FUNCTION G07(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G07=Z(2)+X(2)
      IF (G07 .GT. G(6)) G07=G(6)
      RETURN
      END
      FUNCTION G08(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G08=(1.-EXP(-B(3)))*(X(3)-B(5))
      IF (G08 .LT. 0.) G08=0.
      RETURN
      END
      FUNCTION G09(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G09=B(8)-X(4)
      IF (G09 .GT. G(8)) G09=G(8)
      RETURN
      END
      FUNCTION G10(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G10=G(8)-G(9)
      RETURN
      END
      FUNCTION G11(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G11=(1.-EXP(-B(4)))*(X(4)-B(6))
      IF (G11 .LT. 0.) G11=0.
      RETURN
      END
      FUNCTION G12(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G12=Z(4)+G(5)+G(7)
      RETURN
      END
      FUNCTION G13(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G13=B(7)-X(3)
      IF (G13 .GT. G(12)) G13=G(12)
      RETURN
      END
      FUNCTION G14(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G14=G(12)-G(13)
      RETURN
      END
      FUNCTION G15(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      G15=G(10)+G(11)+G(14)
      RETURN
      END
      FUNCTION F0101(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0101=G(1)-G(3)
      RETURN
      END
      FUNCTION F0202(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0202=Z(2)-G(7)
      RETURN
      END
      FUNCTION F0303(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0303=G(13)-G(4)-G(8)
      RETURN
      END
      FUNCTION F0404(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0404=G(9)-G(11)
      RETURN
      END
      FUNCTION F0505(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0505=G(15)-X(5)
      RETURN
      END
      FUNCTION F0606(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0606=Z(3)-X(6)
      RETURN
      END
      FUNCTION F0707(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0707=(G(15)-Z(3))**2
      RETURN
      END
      FUNCTION F0808(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      AA=AB=.0000001
      IF (G(15) .GT. AA) AA=G(15)
      IF (Z(3) .GT. AB) AB=Z(3)
      AC=LOGF(AA)-LOGF(AB)
      F0808=AC*AC
      RETURN
      END
      FUNCTION F0909(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F0909=G(4)-X(9)
      RETURN
      END
      FUNCTION F1010(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1010=G(4)
      IF (S4(K,0).NE.S4(K,1)) F1010=G(4)-X(10)
      RETURN
      END
      FUNCTION F1111(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1111=G(4)
      IF (S4(K,0).EQ.6..AND.S4(K,1).EQ.7.) F1111=G(4)-X(11)
      RETURN
      END
      FUNCTION F1212(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1212=G(3)-X(12)
      RETURN
      END
      FUNCTION F1313(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1313=G(3)
      IF (S4(K,0).NE.S4(K,1)) F1313=G(3)-X(13)
      RETURN
      END
      FUNCTION F1414(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1414=G(3)
      IF (S4(K,0).EQ.6..AND.S4(K,1).EQ.7.) F1414=G(3)-X(14)
      RETURN
      END
      FUNCTION F1515(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1515=Z(1)+Z(2)-X(15)
      RETURN
      END
      FUNCTION F1616(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1616=Z(1)+Z(2)
      IF (S4(K,0).NE.S4(K,1)) F1616=Z(1)+Z(2)-X(16)
      RETURN
      END
      FUNCTION F1717(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1717=Z(1)+Z(2)
      IF (S4(K,0).EQ.6..AND.S4(K,1).EQ.7.) F1717=Z(1)+Z(2)-X(17)
      RETURN
      END
      FUNCTION F1818(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1818=G(15)
      IF (S4(K,0).NE.S4(K,1)) F1818=G(15)-X(18)
      RETURN
      END
      FUNCTION F1919(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F1919=G(15)
      IF (S4(K,0).EQ.6..AND.S4(K,1).EQ.7.) F1919=G(15)-X(19)
      RETURN
      END
      FUNCTION F2020(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F2020=Z(3)
      IF (S4(K,0).NE.S4(K,1)) F2020=Z(3)-X(20)
      RETURN
      END
      FUNCTION F2121(K,X,B,R,Z)
      DIMENSION X(1),B(1),R(1),Z(1)
      F2121=Z(3)
      IF (S4(K,0).EQ.6..AND.S4(K,1).EQ.7.) F2121=Z(3)-X(21)
      RETURN
      END
      SUBROUTINE YCOMP(K,X,B,R,Y)
      DIMENSION X(1),B(1),R(1),Y(1)
      DO 5 I=1,21
    5 Y(I)=X(I)
      RETURN
      END
      FUNCTION S1(K)
      DIMENSION CP(12)
      DATA ((CP(I),I=1,12)=.00029,.00075,.00181,.00233,.00329,
     1.00340,.00416,.00381,.00290,.00152,.00057,.00029)
      M=S4(K,1)
      S1=254.*S3(K)*CP(M)
      RETURN
      END
      FUNCTION S2(K)
      DIMENSION STWO(18)
      KK=K+1
 5    IF (KK .GT. 18) GO TO 100
      IF (KK.EQ.18) ITIME=0
      IF (KK .EQ. 1) GO TO 150
 25   S2=254.*STWO(KK)
      RETURN
 100  KK=KK-18
      GO TO 5
 150  IF (ITIME.EQ.1) GO TO 25
      READ(5,110) (STWO(I),I=1,18)
      ITIME=1
      GO TO 25
 110  FORMAT(18F4.2)
      END
      FUNCTION S3(K)
      DIMENSION STHREE(18)
      KK=K+1
 5    IF (KK .GT. 18) GO TO 100
      IF (KK.EQ.18) ITIME=0
      IF (KK .EQ. 1) GO TO 150
 25   S3=STHREE(KK)
      RETURN
 100  KK=KK-18
      GO TO 5
 150  IF (ITIME.EQ.1) GO TO 25
      READ(6,110) (STHREE(I),I=1,18)
      ITIME=1
      GO TO 25
 110  FORMAT(18F4.1)
      END
      FUNCTION S4(K,MK)
      DIMENSION ISFOUR(24)
      IF (K .NE. 0) GO TO 15
      IF (FIRST .EQ. 1.) GO TO 15
      READ(7,105) (ISFOUR(I),I=1,24)
105   FORMAT(24I3)
      FIRST=1.
      MM=0
      M=1
 15   IF (K .LE. MM) GO TO 50
      MM=MM+ISFOUR(M)
      M=M+1
      GO TO 15
 50   S4=M+6-1
      IF (K+MK .GT. MM) S4=S4+1.
 75   IF (S4 .GT. 12.) GO TO 200
      RETURN
200   S4=S4-12.
      GO TO 75
      END
      FUNCTION S5(K)
      DIMENSION SFIVE(20)
      KK=K+1
  5   IF (KK .GT. 20) GO TO 100
      IF (KK .EQ. 1) GO TO 150
 25   S5=40.6*SFIVE(KK)
      RETURN
100   KK=KK-20
      GO TO 5
150   READ(8,110) (SFIVE(I),I=1,20)
      GO TO 25
110   FORMAT(10F6.3)
      END
```

I-file, N-file and B-file

```
00001:YMAX=21
00002:LPLUN=44 LP=Y1 LP=Y2 LP=Y3 LP=Y4 LP=Y5 LP=Y6 LP=Y7 LP=Y8
00003:LP=Y9 LP=Y10 LP=Y11 LP=Y12 LP=Y13 LP=Y14 LP=Y15 LP=Y16 LP=Y17
00004:LP=Y18 LP=Y19 LP=Y20 LP=Y21
00005:DUMPLUN=40 D=Y1 D=Y2 D=Y3 D=Y4 D=Y5 D=Y6 D=Y7 D=Y8 D=Y9
00006:D=Y10 D=Y11 D=Y12 D=Y13 D=Y14 D=Y15 D=Y16 D=Y17 D=Y18 D=Y19
00007:D=Y20 D=Y21
00008:INPUT=60
```

```
00001:XN()=0 0 2.3114E3 10.97E3 0 0 0 0 0 0
00002:0 0 0 0 0 0 0 0 0 0
00003:0
00004:INPUT=60
```

```
00001:B()=1.82E3 2.96E3 1.48  3.75 2.96E3 9.97E3 4.59E3 11.9E3
00002:.25 100 .0075 762 .3
00003:INPUT=60
```

Sample Run

```
#EQUIP,44=LP,40=FILE
#EQUIP,5=*S2INP,6=*S3INP,7=*S4INP,8=*S5INP
#LABEL,44/SAVE FOR FLEX1 USER

#*FLEX1




*****************************************************

   FLEX1   GENERAL MODEL SIMULATOR   OS3 VER 1.0

*****************************************************

ENTER INPUTS/COMMANDS
>NUMBER=21 TMAX=50 FUNLOAD=*OHM02
>INPUT=*OHMI2


ENTER INPUTS/COMMANDS
>INPUT=*OHMB2


ENTER INPUTS/COMMANDS
>INPUT=*OHMN2


ENTER INPUTS/COMMANDS
>TTYPRT=1 TTY=Y(3) TTY=Y(4) TITLE=FLEX1 SAMPLE RUN
>LIST=N
     21
>LIST=B(3)
     1.4800
>LIST=X(3)
  2311.4000
>TTY=Y(5) TTY=Y(6) TTYPRT=10
>SIMULATE
```

```
09/26/73  12:13 PM  FLEXI SAMPLE RUN
      09261213

STATE VARIABLES    21     TTY PRINT INTERVAL  10
INITIAL TIME        0     DATA DUMP INTERVAL   1
TERMINATION        50     LP PRINT INTERVAL    1
TIME STEP (DT)      1

CONTROL INPUT LUN  60     CONTROL LIST LUN    61
LINE PRINTER LUN    0     FUNCTION INPUT LUN  54
DATA DUMP LUN      40

 ERROR CHECKING
 LOG ON LUN  0


                  ( 1)         ( 2)         ( 3)         ( 4)         ( 5)
                  ( 6)         ( 7)         ( 8)         ( 9)         (10)
                  (11)         (12)         (13)         (14)         (15)
                  (16)         (17)         (18)         (19)         (20)
                  (21)

COMPUTE SEQ         1            2            3            4            5
                    6            7            8            9           10
                   11           12           13           14           15
                   16           17           18           19           20
                   21

INITIAL X          0E 00        0E 00  2.3114E 03   1.0970E 04        0E 00

                   0E 00        0E 00       0E 00        0E 00        0E 00

                   0E 00        0E 00       0E 00        0E 00        0E 00

                   0E 00        0E 00       0E 00        0E 00        0E 00
                   0E 00

UPPER LIMIT    1.0000E100   1.0000E100   1.0000E100   1.0000E100   1.0000E100
               1.0000E100   1.0000E100   1.0000E100   1.0000E100   1.0000E100
               1.0000E100   1.0000E100   1.0000E100   1.0000E100   1.0000E100
               1.0000E100   1.0000E100   1.0000E100   1.0000E100   1.0000E100
               1.0000E100
```

```
LOWER LIMIT       0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00

ERROR LIMIT       0E 00        0E 00        0E 00        0E 00        0E 00

                  0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00        0E 00        0E 00        0E 00        0E 00
                  0E 00        0E 00        0E 00        0E 00        0E 00

                  0E 00

B CONSTANTS   1.8200E 03   2.9600E 03   1.4800E 00   3.7500E 00   2.9600E 03

              9.9700E 03   4.5900E 03   1.1900E 04   2.5000E-01   1.0000E 02
              7.5000E-03   7.6200E 02   3.0000E-01



09/26/73   12:13 PM   FLEX1 SAMPLE RUN
       09261213


FLOW FUNCTIONS - * = DEFINED

         1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
F( 1,J)  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 2,J)  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

F( 3,J)  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 4,J)  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 5,J)  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 6,J)  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 7,J)  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0  0
F( 8,J)  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0  0

F( 9,J)  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0  0
F(10,J)  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0  0

F(11,J)  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0  0
F(12,J)  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0  0
F(13,J)  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0  0
F(14,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0  0
```

```
F(15,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0  0

F(16,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0  0
F(17,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0  0

F(18,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0  0

F(19,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0  0

F(20,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *  0
F(21,J)  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  *


INDICIES OF DEFINED G FUNCTIONS

      1    2    3    4    5    6    7    8    9   10   11   12   13   14   15


YCOMP SUBROUTINE IS DEFINED
ECOMP SUBROUTINE IS DEFINED



09/26/73  12:13 PM  FLEX1 SAMPLE RUN
      09261213



PAGE    1   09/26/73   12:13 PM   FLEX1 SAMPLE RUN
      09261213

    TIME        Y( 3)         Y( 4)          Y( 5)          Y( 6)

       0 2311.400000 1.09700E 04              0              0

      10 2089.815223 9970.000000              0       4.547200

      20 1960.153471 9970.000000              0       3.735200
```

```
     30 1888.829055 9970.000000          0     2.720200
X(13) = -9.31323E-10  IN 1

X(16) = -9.31323E-10  IN 1


     40 1856.438658 9970.000000          0     1.948800

     50 1840.674564 9970.000000          0     1.827000

END OF SIMULATION RUN


ENTER INPUTS/COMMANDS
>CLEAR




**************************************************

   FLEX1   GENERAL MODEL SIMULATOR   OS3 VER 1.0

**************************************************

ENTER INPUTS/COMMANDS
>STOP

END OF FLEX1 SESSION
#
```