CONVERTING MODELS IN FORTRAN IV TO ENABLE CONTINUOUS SYSTEMS MONITORING PROGRAM (CSMP) EXECUTION CONTROL

Myron D. Paine, Oklahoma State University
Fred Witz, Computer Consultant
Frank Crow, Oklahoma State University

ABSTRACT

This paper covers the development of (1) an optimization routine to be incorporated into the CSMP system to enable the optimization of up to six undefined parameters, (2) a stand alone program developed to convert Fortran IV models so the format will be compatible with the CSMP execution routines.

By using the stand alone routine and the optimization subroutine, a Fortran IV agricultural model can be converted to a form that would enable CSMP run execution control. This method has been used to convert the USDAHL-74 watershed hydrology model and the SIMCOT cotton model to enable CSMP execution and optimization.

INTRODUCTION

Mathematical models of plant and animal growth are being developed rapidly as agricultural scientists begin to use the computer for more than statistical analysis. However, most agricultural scientists have not yet had the opportunity to develop experience with systems simulation languages like CSMP, GASP IV, and GPSS. Thus the mathematical models for plant and animal growth are often computer programs written in Fortran IV.

Because these computer programs are used to simulate the growth of plants or animals through time, the model calculations are repeated at selected intervals. Thus the models, in effect, are a continuous system model with an integration step size.

A problem in the development of agriculture models is that many physical relationships are undefined or are so complex that simple analogies must be used. The agricultural modeler is confronted with many situations where he must make intelligent guesses of functional relationships. The primary task is to verify that the hypothetical relationships represent reality. Inherent in these hypothetical relationships are a number of parameters which may be undefined.

Support for this paper, manuscript no. 310 came from project 1568 of the Oklahoma Agricultural Experiment Station.

Thus, calibration of the model requires repeated access to parameters which may be revised as experience grows. Eventually computer optimization of some undefined parameters is required. Agricultural models represent complex systems. Even the wisest programmers have difficulty providing sufficient READ and WRITE statements, with switches, to enable access to all parameters and variables of the model. The adaptation of optimization routines for the many parameters that must be adjusted is even more difficult.

The Continuous Systems Monitoring Program (CSMP) by IBM is an effective way to simulate a continuous system.[1,2] The execution phase of CSMP has advantages over executing a model in Fortran alone. The major advantages include: (1) the possibility to access parameters without rewriting the format statements and recompiling, (2) ease of changing printed output, which enables researchers to investigate subsections of the model under study, (3) the features of run control, step size, rerun, print or plot routines, output interval selection, and the choice of integration routines. The CSMP translation phase can also be used to convert small models to Fortran subroutines to be compatible with CSMP execution.

BACKGROUND

An optimization routine was written to be incorporated into the CSMP system to solve for up to six undefined parameters at a time.[3,4] The development enabled access to all parameters of a given CSMP model. The access greatly accelerated the calibration procedure.

Another problem encountered with a large model was the time required for initialization via CSMP input translation routines. The execution phase of CSMP was speeded up by reading an unformated record into the model COMMON. The unformated record contained initial values of the parameters.

The optimization routine and the initialization of parameters worked well with the CSMP control. The use of the same procedure on other agricultural models, written originally in Fortran IV, was desirable. The United States Department of Agriculture Hydrograph Laboratory (USDAHL)

Watershed Model and the Cotton Simulation Model (SIMCOT) were in the process of being calibrated against data.[5,6] Both of these models were written in Fortran IV. However, the size of the USDAHL model clearly exceeded the limits of the CSMP translation phase. Thus, a stand alone program was developed to convert Fortran IV models of any size to a format that would be compatible to CSMP execution routines.

This paper discusses the optimization routine used with CSMP execution control, the stand alone program to convert Fortran IV programs to CSMP execution, and the loader program used to expedite the initialization of parameters.

OPTIMIZATION ROUTINE

The optimization routine was originally written by Dr. John Witz and has two major sections. These sections are, (1) the calculation of the performance index for a single simulation run, and, (2) the optimization section. The optimization section provides the logic for parameter manipulation to activate either a grid search routine or a conjugate gradient routine.

PERFORMANCE INDEX CALCULATION

The squared error performance index (PI) is commonly used for model calibration. The PI user must sample simulated variables and compare the variable value with the desired value.

The CSMP variables are specified in terms of their position in COMMON. A single dimension vector, C, was made equivalent to the portion of COMMON CSMP uses for parameters and variables to allow selection of any variables at execution time. The PI user must know the C vector subscript to be able to use a particular variable. The user needs a cross reference index giving the parameter name and its location in the C vector.

A routine to make squared error calculations for up to five variables was written. A requirement for more than five variables was not encountered during five years experience.

TYPES OF PERFORMANCE INDEX

Two major possibilities for calculating the PI during a single execution of a model occur in the dynamic section of the CSMP run and in the terminal section. CSMP translates model input statements into a Fortran IV subroutine named UPDATE. UPDATE is divided into four sections; SYSTEM, INITIAL, DYNAMIC, and TERMINAL. The subroutine UPDATE is called repeatedly during the execution run. The dynamic PI is calculated in the DYNAMIC section of CSMP UPDATE during execution of the simulation run. Both PI's are squared error calculations. The two PI's can be summed to yield a composite PI.

Normally, the dynamic and terminal PI calculation involves physical variables of the model best suited to squared error summation. However, a cost PI is often desired for agricultural models. Thus, the PI routines were written to include an optional section to compute cost performance of the model.

The equation defining the DYNAMIC squared error PI is written as:

$$PXD = \Sigma_{i=1}^{k} \Sigma_{j=1}^{LND} CMD_j * (V_j(t_i) - PVD_{ij}(t_i))^2$$

where LND is the number of variables specified, k is the number of data points; CMD_j are the weighting factors for the jth simulation variable; $V_j(t_i)$ is the jth simulation variable at data point t_i, and $PVD_j(t_i)$ is the corresponding desired data value. The data points, PVD_j and t_i are entered as input data. Normally the t_i are simulation times but other optional parameters can be used to define the data point.

The TERMINAL performance index equation is written as:

$$PXT = \Sigma_{j=1}^{LNT} CMT_j * (V_j - PVT_j)^2$$

where LNT is a number of variables specified, CMT_j are the weighting factors, V_j is the jth simulation variable and PVT_j is the corresponding data point.

The economic PI is a simple way to sum variables of interest. The weighting factors in economic situations are appropriate prices. The DYNAMIC cost performance index can be written as:

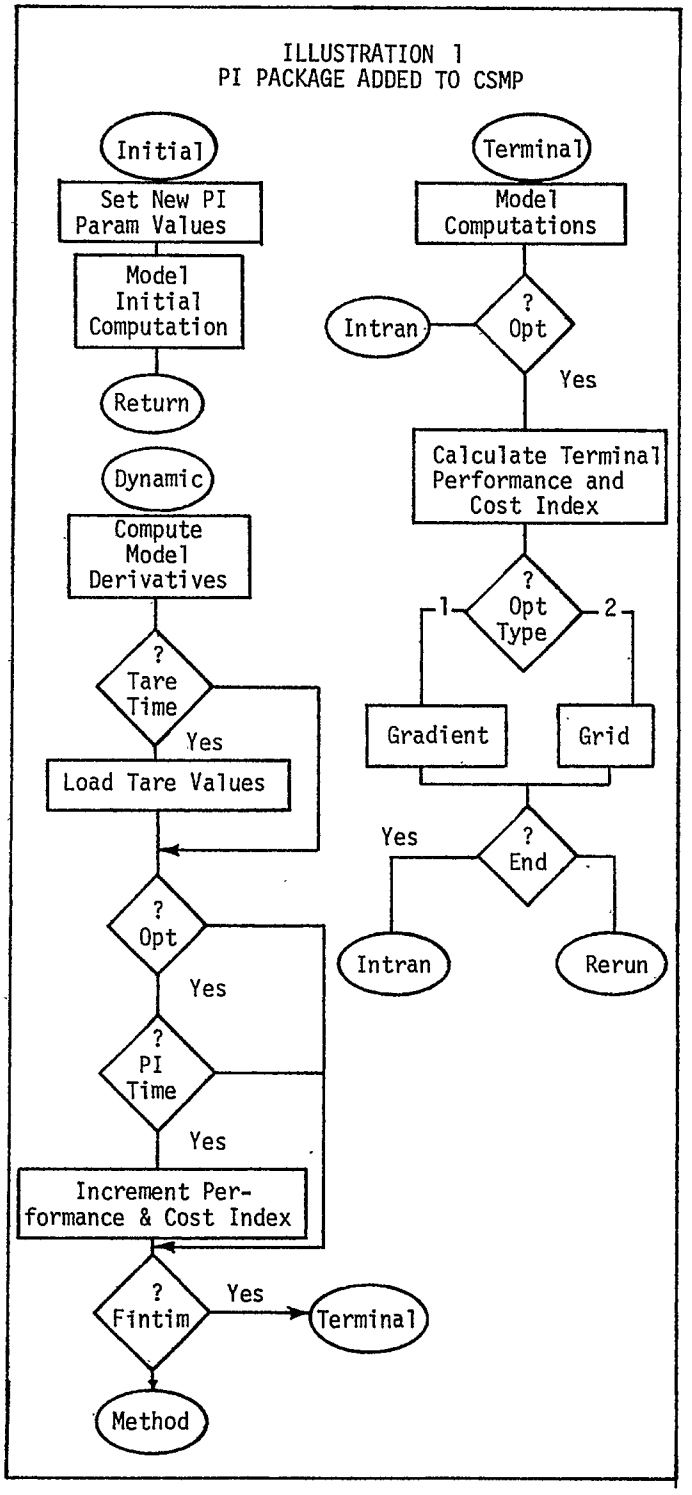
$$PCD = \Sigma_{j=1}^{LNC} COST_j * V_j(t_i), i=1, 2, \ldots k$$

where LNC is the number of variables specified, $COST_j$ are the prices, $V_j(t_i)$ is the jth simulation variable at data point t_i.

The instantaneous value of the DYNAMIC cost PI at specified data points is more important than accumulated PI. CSMP print routines can be used to interrogate the cost PI during the dynamic simulation. The DYNAMIC cost PI is not accumulated. A similar equation is used for the TERMINAL cost index. The TERMINAL cost index can be added to the overall PI, if desired. The PI and optimization flow chart is shown in Illustration 1.

Parameter Adjustments

The optimization package includes both a conjugate gradient routine and a grid search routine. Either routine can be used to study the effect of up to six parameters on the five variables of the PI. For each set of parameter values, the PI is calculated and passed to the proper optimization routine. For each parameter used with the conjugate gradient routine, the location in the C vector and the initial value must be specified. Also, the



information for the PI must be specified.

For the grid search, the number of steps to be taken and the percentage movement for each step must be given. The grid search parameters are calculated according to the formula

$$T_j(r) = (1 + DEL_j * s_j),\ s_j = 1, 2, \ldots N$$

where $T_j(r)$ is the jth parameter adjustment value computed from the rth run. DEL_j is the percentage increment and N is a number of steps to be taken with the jth parameter. The grid search routine prints the value of the PI for each step. The minimum value of PI in the grid and the associated steps are printed at the end of the program execution.

The conjugate gradient method also adjusts the parameters via computed $T_j(r)$ values for each step in the method. Parameter adjustments for the next run are entered in the CSMP INITIAL section by the equation

$$C(LX_j) = X_j * T_j(r-1)$$

where LX_j are the parameter locations in the C vector, X_j is the initial parameter value, and $T_j(r-1)$ is the jth parameter adjustment value calculated from the last run.

To enable adaptation of the PI package to other models, the PI and associated optimization routines were written as a Fortran IV subroutine with an ENTRY for each of the INITIAL, DYNAMIC, and TERMINAL sections of the CSMP execution phase. To make the PI subroutine compatible with the particular model under study, the parameters and variables of the PI routine must be included in COMMON during UPDATE compilation.

ARRANGEMENT OF UPDATE COMMON

A cross reference index of variable names versus location in the C vector is needed for optimization setup. Thus, a preliminary step in model calibration is the assembly of a cross reference index on cards. Experience with the PI package revealed a definition of the variable names was also useful. With the description of the variable names added to the index cards, the index becomes a dictionary. The dictionary can be sorted alphamerically by variable name and numerically by location in C vector. The two lists, plus an additional list of the variable names as arranged in COMMON, are handy references for the user.

Experience with a large dynamic model indicated the CSMP symbol table lookup procedure was time consuming. The symbol table procedure was speeded up by locating the most commonly used variables and parameters first in their respective sections of the COMMON.

Even with models initially translated by CSMP, the reorganization of COMMON and the symbol table resulted in more efficient execution. Originally, the COMMON reorganization was done by hand. The

hand procedure required careful cross checking between the COMMON area and the symbol tables. Also, the adjustment of the CSMP table lengths was important.

FORTRAN MODEL CONVERSION TO CSMP

The availability of other agricultural models, in particular the watershed model, USDAHL-74, and the cotton simulation model, SIMCOT, in Fortran IV languages, sparked the desire to execute these models by CSMP. The USDAHL-74 model clearly exceeded the limits of translation capabilities of CSMP. The total SIMCOT package would have also.

Furthermore, to use the models efficiently, a dictionary of variable names had to be assembled. The desire to use CSMP and the necessity to assemble a dictionary led to development of a stand alone program named LONER. LONER was written to convert dictionary listings to a compatible symbol table and to create a default data set. Also, LONER computed the required CSMP table lengths.

The large models involved numerous parameters. Many of the parameters were semi-permanent. Most of the parameters were not changed during a single execution. An unformatted record containing the values of these parameters, was read into the COMMON area at the beginning of the program. The input translator of CSMP was then used only for changes in parameters. Thus, the execution of the CSMP input routine was reduced to a minimum. Therefore, use of a default record greatly reduced execution time.

The Fortran subroutines were then compiled using a modified COMMON corresponding to the symbol table. The CSMP table lengths were entered by a loader subroutine. The loader routine also added the symbol table and the default parameters to the COMMON area.

CSMP DESCRIPTION

Some of the subtleties of the CSMP program must be understood to describe the details of the LONER program. CSMP uses a symbol table to determine the location of variable names in UPDATE COMMON. The CSMP symbol table is a packed listing of the variable names in the program. CSMP input and output procedures look up the variable names during the execution phase by comparing the given name to the symbol table list. The location of the name in common is thus determined.

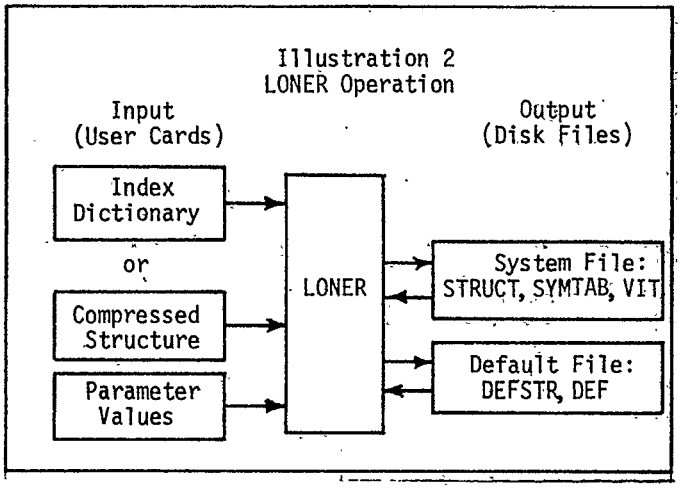
CSMP routines allow use of single dimension vectors. The vectors are named in the symbol table. The number stored in the COMMON location identified by the symbol table is an indirect address of the first item in the vector. The numbers stored in the section of COMMON located by vector names are called the Vector Indirect Table.

The CSMP symbol table is arranged by sections with the integrator output, input, and initial conditions first, followed by parameters entered as CSMP data, followed by variables, followed by vector names, and finally a list of integers. During input, CSMP interrogates the integer list in reverse order by starting at the last entry in the symbol table and reading backward through the integer list. After making the integer or floating point decision, the CSMP routine starts searching at the beginning of the symbol table to locate the position of the parameter value in COMMON. The CSMP routine requires internal numbers defining table lengths to accomplish the symbol table search. Normally, CSMP sets these numbers in the SYSTEM segment of UPDATE.

USER PREPARATION

The input into LONER can be of four types; (1) the complete dictionary card listing of the program variables and parameters, (2) the compressed form of card input consisting of an already defined symbol table, vector indirect table, and inter table with respective lengths of these tables, (3) compiled structure and defaults stored on disk, and, (4) parameter values on cards. Illustration 2 shows the inputs and outputs of LONER.

Illustration 2
LONER Operation



To prepare for using LONER the first time, the programmer assembles a card dictionary of the variable and parameter names in the Fortran IV subroutines. Then the dictionary cards are sorted by hand into the proper COMMON sequence of UPDATE and the desired priority. (The COMMON sequence of UPDATE is integrator output, integrator input, initial conditions, parameters, variables, and vectors). Within each CSMP section, the programmer lists the variables or parameters in order of priority with those to be accessed most often placed first. The dictionary listing, arranged by hand, is then used as input to LONER.

Section identification cards used to identify the COMMON section of UPDATE are added to the dictionary. The programmer uses other control cards to execute phases of LONER. A control card or section identification card is identified by a colon in

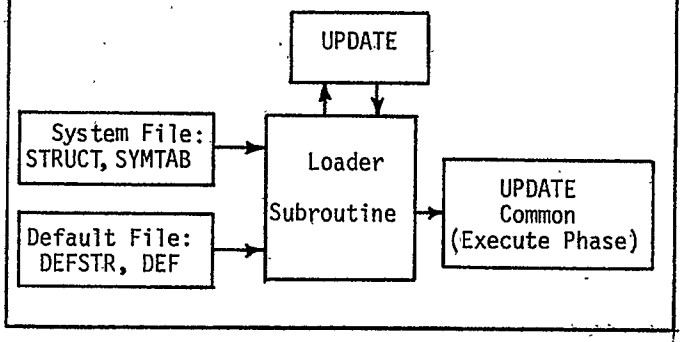
column 1. Comment cards are identified by an asterisk in column 1.

The output from LONER consists of two main files, usually written to disk. These two files are the system file and the default file. The system file contains three records; the model structure, the symbol table, and the vector indirect table. The structure record is a fixed length record containing the table lengths of the model and other descriptive information.

The structure of the model is further defined in the symbol table and the vector indirect table. Both records are variable length. The symbol table record contains the packed variable and parameter names including vector and the integer names. The vector indirect table record contains the indirect addresses of the beginning of the vector represented by the name in the symbol table.

The default file contains two records. The structure of the default is a fixed length record which defines the length of default and contains information to insure the default file is compatible with the structure of the model being run. The default record contains the parameter values to be used unless changed by the user. The LONER program can, upon command, create the default file. The user must input parameter values into LONER to set values other than zero.

Illustration 3
Load System Operation



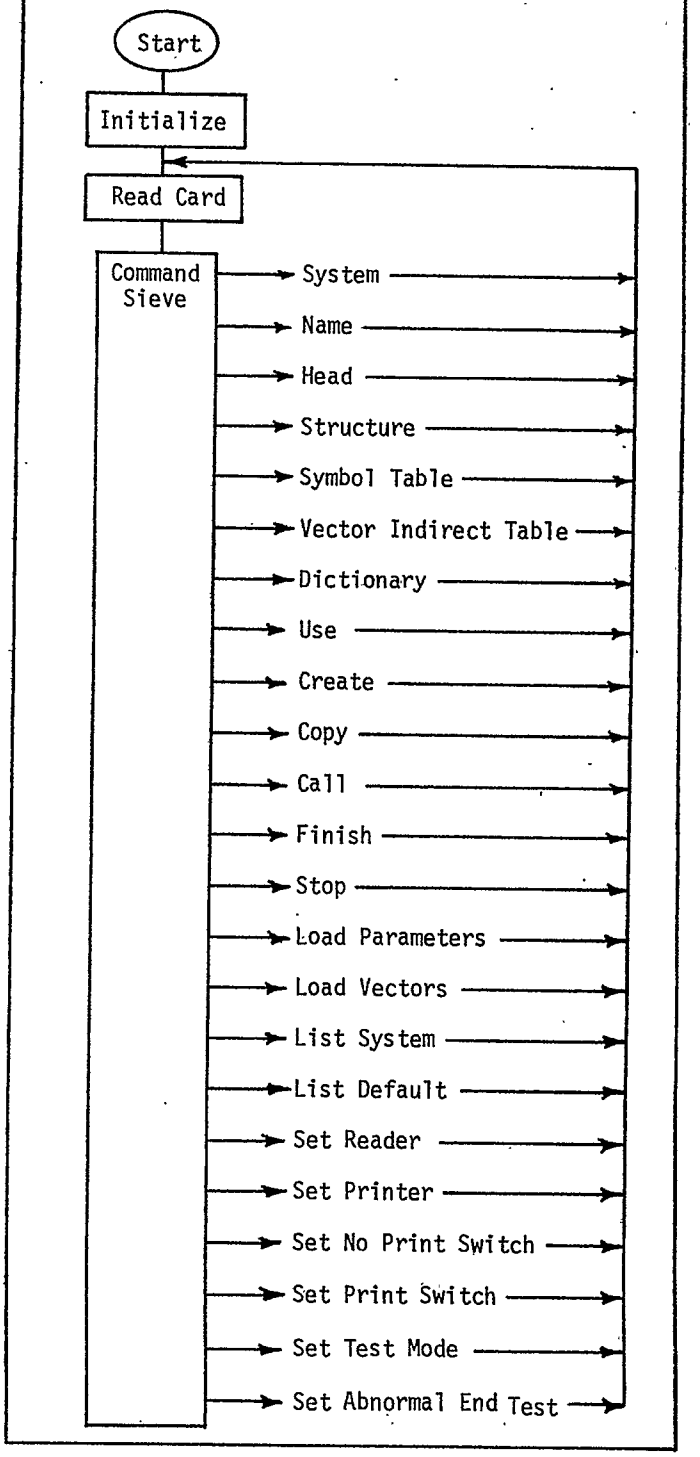
To execute the model, a short loading subroutine, called from the UPDATE SYSTEM section reads the structure information from the system file and inserts table lengths in the correct location in COMMON. Then the symbol table is read into the correct location in COMMON. With the structure of the model defined, the loader routine interrogates the structure of default record from the default file. If the information is compatible with the model structure, the default record is entered into COMMON. The loader program also contains subroutines enabling the user to insert parameters or vectors directly into COMMON. The loader initialization of the model is faster than using the CSMP input translation package.

DESCRIPTION OF LONER

Input Control Loop

The flow diagram for LONER input control loop is given in Illustration 4. Basically, LONER is a group of sections accessed in a given order by control cards. The subsections perform their procedure and return control back to the command

Illustration 4
Loner Command Loops



input loop which reads the next control card. Thus, the description of LONER is a description of the individual subsections.

Loading the System Information

System

The system section initializes the program and assigns output files for the system. The system section is required whenever system information is to be read from cards and written to disk.

Name and Head

The name of the model is used for disk file security and to insure the proper default is being used with the corresponding model structure. The name on the following card will appear both in the structure and the default structure records on the system and default files, respectively. The heading command is for user convenience. The command enables print of the heading describing the dictionary or model on the printed output. Both commands are optional.

Structure

The structure section reads four cards. If the cards are all blank the LONER program will do the necessary counting. If they are not blank, the input structure, resulting in table lengths for CSMP execution, overrides the counts generated by the LONER program. The operation of the structure section is straight forward, it reads the numbers of integrators, parameters, variables, number of indirect vectors, number of vector lengths and the number of integer names. The use of default is set and the appropriate table lengths are calculated. The CSMP limits are checked to see if the model will fit.

Symbol Table

The symbol table section is used only for input of the packed symbol table. This section reads the symbols in compressed form, fills out the symbol table with blanks if required. An optional parameter on the control card specifies the number of cards to follow. If specified, the number of cards is checked. Any excess cards are printed and ignored.

Vector Indirect Table

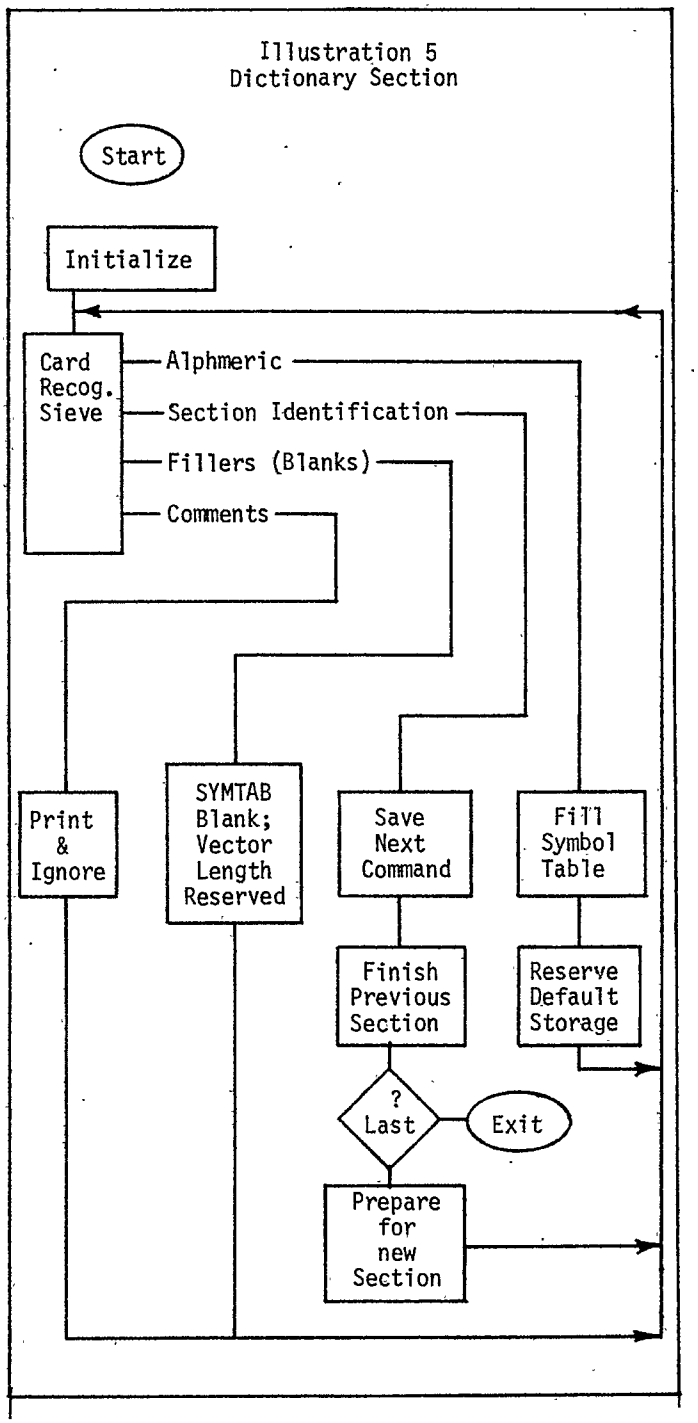
The vector indirect table is used only for the compressed form of the input data. This section of the routine reads the cards containing the vector indirect addresses. As with the symbol table section, the vector indirect table fills in where necessary with zeros. As with the symbol table input, the number of cards to follow can be specified.

Dictionary

The dictionary section of LONER, shown in Illustration 5, reads the card dictionary and creates a symbol table and the vector indirect table.

After the initialization of the dictionary section, a card is read and the first word on that card is interrogated for recognition. If the card is alphameric, the dictionary section fills the symbol table and reserves storage space in the default table.

Illustration 5
Dictionary Section



Section identification cards are identified by a leading colon, the same as command cards. However, the section identification cards are dispersed through the dictionary and identify the appropriate CSMP section of the following variable names. CSMP sections are timer, the integrators, parameters, variables, vectors, and integers. An end card is used to identify the end of the last section.

The dictionary section counts the number of symbol names and reserves the appropriate default storage for each section. The limits of a CSMP section cannot be set until the dictionary section receives a section identification card indicating a new section will start. The dictionary routine then saves the identification information and proceeds to finish the computations for the previous section. Upon returning from the previous section, the dictionary routine checks if the program end has been reached. If not, the routine prepares for the next section and continues.

Filler cards are identified either with a plus, for a single dimension variable, or ampers and for vector storage. When these cards are encountered, the symbol table is left blank, and the appropriate length is reserved in default. As elsewhere in the LONER program, the dictionary section recognizes comment cards by an asterisk. Comment cards are printed in the output listing but ignored otherwise.

The section identification card for integers is optional. If it is included, the variable names following will be identified in the integer section of CSMP symbol table. The other option is to identify integers by using a negative identification number on the dictionary cards. The dictionary section of LONER recognizes the negative number and later generates an integer table for these variables. Any number of variable names can be integer. However, CSMP can recognize only 20 names. Thus, the user may want to specify 20 most important names to be entered through CSMP input.

Use

The use section bypasses the dictionary or compressed input section and recovers a previously compiled system from disk storage. Thus, the routine goes to the systems file and reads in the fixed length structure record. The structure information enables the reading of the variable length symbol and vector indirect table.

Loading Default Information

Once the user of LONER has defined a system either by using the compressed form on cards, the long dictionary listing, or recovering the system from disk storage, the default information can be processed.

Create

The create section creates an empty default and inserts the vector indirect table.

Copy

The copy section enables the LONER user to copy an old default record from the fault file.

Call

At this point in LONER execution, the user must have a system and an empty or an old default defined. The CALL section enables the user to read in new parameters or new vectors to either (1) fill an empty default, or (2) modify an old default.

Finish

At this stage in LONER execution, the user has a compiled system and updated default. The finish section results in the new default file written to disk.

Two supporting subroutines are required. The subroutines enable LONER to read or write a variable length record in unformatted mode into or out of a fixed dimensioned array. The subroutines accomplish this without resorting to a slower implied do loop.

Other Loner Execution Commands

The load parameters and load vector sections cause the program to read parameters and vectors, respectively. The list system and list default sections cause LONER to print out the system records and the default records, respectively. The stop section ends LONER execution.

Set Commands

A number of commands are provided for user convenience to enable the user to set the reader and the printer file numbers, to set print or no print switches, to set a test mode and to test abnormal endings. The test mode prevents output to either system or default file. The abnormal ending test is used primarily for LONER modification by a programmer. When LONER encounters an abnormal end, the routine prints out an organized dump.

DISCUSSION

The LONER program has been used in three major applications. Although the USDAHL-74 model duplicated runoff data on one experimental watershed near Oklahoma State University, the model failed to give accurate predictions for a second watershed. The Fortran IV subroutines of USDAHL program were converted to use with CSMP execution. The conversion enabled effective simulation of a model consisting of about 2,000 cards and 15 subroutines. CSMP execution and the PI package was used to locate model parameters to enable the model to predict accurately on the second watershed.

The discrepancy between the model performance and real data occurred when excessive runoff after a long dry spell was predicted by the model and real data indicated very little runoff. A series of PI investigations indicated the equation for soil cracking was significant. However, the soil cracking parameters were internal variables calculated from other input data. When the Fortran subroutine made the calculation, the input variables were destroyed. Thus, the PI package could not function. Rewriting of the Fortran subroutine was suspended because of personnel termination.

The LONER routine was used to enable CSMP execution of the SIMCOT routines for germination and immergence. The SIMCOT germination and emergence program was compared to field data collected by Oklahoma State University. The germination and plant growth simulation was easily compared visually to the actual data.

Several elements in the model were erratic. The CSMP ability to select different output parameters and frequencies enabled the researcher to systematically inspect the model performance. An integration-step size incompatibility was detected in the equations defining moisture at various soil depths. Improvement of the model required structural changes. Personnel graduation terminated the modeling efforts.

The third major use of the LONER program has been to assemble the COMMON area of a physiological beef model, BOSC05. The physiological model is a combination of the well developed thermal and growth sections of the old BOSCOM model, a newly developed stomach section, a newly developed anaerobic digestion section, and a tentative protein pathway model. Combining the four models into one comprehensive model was facilitated by LONER. The symbol table, COMMON and default size have been defined. Structure organization and parameter calibration of the entire model is proceding.

The PI package has enabled calibration of the parameters of the BOSC05 model. When calibration becomes erratic or difficult, the CSMP capability to print out selected variables at desired frequencies has enabled the researcher to identify needed structural changes. A subsection of the model structure can be easily changed, if necessary, by recompiling the proper subroutine. Each BOSC05 subroutine has ENTRY calls, for the INITIAL, DYNAMIC, and TERMINAL sections of CSMP.

Conclusions

(1) The execution routines of CSMP combined with a general purpose PI package enhances the development and calibration of agricultural models.

(2) LONER compiles CSMP compatible structure and default records from a dictionary list.

(3) LONER enables conversion of large Fortran IV programs to the CSMP execution format without using CSMP translation.

(4) The default file, created with the aid of LONER, allows a significant reduction in model initialization time over CSMP input translation or the usual formatted read statements.

REFERENCES

1. IBM Application Program, 1968, System/360 Continuous System Modeling Program (360A-CX-16X) User's Manual H20-0367-2, IBM Corporation, White Plains, NY 10601.

2. IBM Application Program, 1967, System/360 Continuous System Modeling Program (360A-CX-16X) System Manual GY20-0111-0, IBM Corporation, White Plains, NY 10601.

3. Paine, M. D., 1971. Mathematical Modeling of Energy Metabolism in Beef Animals, Ph.D. thesis, Oklahoma State University, Stillwater, Oklahoma.

4. Witz, John Alva, 1972. Systems Modeling and Computer Simulation of the Beef Feedlot Animal, Ph.D. thesis, Oklahoma State University, Stillwater, Oklahoma.

5. Holtan, H. N., et al., 1974. USDAHL-74 Revised Model of Watershed Hydrology, Plant Physiology Institute Report No. 4, Agriculture Research Service, U. S. Department of Agriculture.

6. ARS-S-52, 1975, Computer Simulation of a Cotton Production System, Agriculture Research Service, U. S. Department of Agriculture.