DESS, AN EXECUTIVE SYSTEM FOR DISCRETE-EVENT SIMULATION, AND APPLICATION STUDIES OF THE SCHEDULING OPERATIONS IN THE PRECIPITATION CIRCUIT OF A BAYER PROCESS PLANT FOR ALUMINA EXTRACTION

DESS, AN EXECUTIVE SYSTEM FOR DISCRETE-EVENT SIMULATION, AND APPLICATION STUDIES OF THE SCHEDULING OPERATIONS IN THE PRECIPITATION CIRCUIT OF A BAYER PROCESS PLANT FOR ALUMINA EXTRACTION

By

RAYMOND H. T. WONG, B.ENG.

A Thesis

Submitted To The Faculty Of Graduate Studies

In Partial Fulfilment Of The Requirements

For The Degree

Master of Engineering

McMaster University

January, 1972

MASTER OF ENGINEERING (1972): McMASTER UNIVERSITY, HAMILTON, ONTARIO

(Chemical Engineering)

TITLE: DESS, an Executive System for Discrete-Event Simulation, and Application Studies of The Scheduling Operations in the Precipitation Circuit of a Bayer Process Plant for Alumina Extraction

AUTHOR: Raymond H. T. Wong, B.Eng. (McMaster University)

SUPERVISOR: Dean A. I. Johnson

NUMBER OF PAGES: vi, 218

SCOPE AND CONTENTS:

An executive system for discrete-event simulation of large, complex systems, such as chemical processes, is generated. Documentation on the system is provided in the form of a user's manual.

Two application studies, involving the simulation of the discrete, scheduling operations in the precipitation circuit of a Bayer Process Plant, are made to illustrate the scope and method of the executive system and to yield further information on the precipitation process.

ACKNOWLEDGEMENTS

The author expresses his gratitude to his supervisor, Dean A. I. Johnson, for his advice and assistance throughout this study.

He also thanks the plant personnel at the Arvida Works of the Aluminum Company of Canada for their keen interest and co-operation.

The financial assistance for the degree obtained through McMaster University is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

GENERAL INTRODUCTION

The development of the "simulation" technique , as it is meant in the modern sense of the word, may be traced with the advent of the computer era in the decade following World War Two. Simulation is not an exact or a specific solution technique, rather its underlying concept is that of an approach to greater understanding of the problem. It is a "trial and error" or experimental method in which a simulation model is manipulated under a variety of expected conditions to produce hypothetical but meaningful results. The simulation model is often a realistic and dynamic representation of a system which duplicates the essence of the operation or set of operations being studied. This model can be complex or relatively simple depending on the requirements of the problem. Usually, however, it would involve a system of sufficient complexity to preclude a straightforward analytical solution. Quite often, the system being studied or modeled has conditions of uncertainty. The treatment of these uncertainties on a repeated-trial basis makes it possible to reflect realistically the effect of these chance variables in the simulated results.

Discrete-event simulation is essentially one kind of dynamic system representation. In contrast to continuous-time simulation, the system dynamics are not duplicated continuously over time during the discrete-event simulation run. A continuous-time simulation model is built on a mathematical framework consisting of differential equations, whereas in a discrete-event simulation model, the operations could represent either continuous activity such as a chemical reaction, or discrete activity such as on-off switching and airplane

arrivals. With continuous activity, however, programs for handling differential equations must be provided. The basic idea of a discrete-event simulation system is primarily oriented to the representation of discretized dynamical behaviour.

CHAPTER 1

A USER'S MANUAL FOR DESS (DISCRETE-EVENT SIMULATION SYSTEM)

Section A: Introduction to Discrete-Event Simulation

(i) Definition of Terms

For purposes of exactitude, digital computer simulation may be defined as a methodology for studying systems. A system is an assembly of interacting processing and decision-making devices created to bring about a desired result. The definition of the scope and limits of a system is relative. Hence, whilst the elements in a chemical process may constitute a system which is the object of study, the various components of a set of computer programs which are used specifically as an aid to the study make up another distinct and separate system. (The latter is sometimes called an executive.) In system terminology, objects which are outside the boundaries of the system but which can influence the system are classified under the system environment.

The term entity denotes an object of interest within the boundaries of a system, and the properties describing the entities are attributes. Any process that causes changes in the system is called an activity. A description of all the entities, attributes and activities as they exist at any one point in time is given by the state of the system. A system state is often affected by changes occurring in the system. Therefore, a system moves from one state to another as its entities engage in activities that change their state. The operations that cause these activities at discrete points in time are called

events. The distinction between activity, event and state can be summarized thus. An activity is a process proceeding over time; its initiation and conclusion are events. When the event occurs, the state of the system has changed because the activity has started, or because it has changed some part of the system.

In discrete-event simulation, processes or activities are not modeled explicitly; they are represented by the terminating event and by changes in the system state, which are modeled as if they all occurred at the time of completion of the event. The behaviour of the system is hence reproduced by examining the system state at the time of each event; and the basis of a rationale for discrete-event simulation rests on the implicit assumption that it is possible to draw inferences about the system's performance from such simulated behaviour. The reproduction of the behaviour of the system is achieved through manipulation of the simulation model, which consists of a set of computer programs incorporating the mathematical equations and/or logical operations that describe the scheme of activities in the system. Given certain input values defining an initial system state, the simulation model transforms the data into realistic values of the significant operating variables and parameters in the system at some point in time. (A variable changes its value over time, and a parameter has a constant value which is not time-dependent.) In order to provide a modeling framework which meets the requirements for storage and retrieval of simulation data and control of the dynamical structure in such a simulation model, and also to facilitate the difficult task of modeling a highly complex system (such as a chemical process), DESS, an executive system for discrete-event simulation was generated.

(ii) A Rationale For DESS

The fundamental concept of a high-level simulation language or system involves the utilization of some form of an information-handling scheme which controls the dynamic manipulation of the data introduced into or generated during the simulation experiment. Since the digital computer has only hardware facilities for sequential computation in a chronological order, programming techniques must also be provided to differentiate between computed data representing simulated activities in a contemporaneous period and other data yielding information on the past or future of the present simulated time instance.

SIMSCRIPT I developed at RAND Corporation by Markowitz, et al[6] is a "bona fide" programming language which requires a compiling system that translates the command statements from SIMSCRIPT into FORTRAN. Its structure is based on FORTRAN programming principles but the philosophy of its design is oriented towards the facilitation of programming tasks that are involved in simulation work, such as, keeping track of the simulated events and entities of interest in the system, and the generation, storage, analysis and reporting of simulation results, both final and intermediate. The language therefore incorporates routines which together perform the tasks of time-keeping, stochastic sampling, the dynamic upkeeping of file-oriented information flow in the system (known as the "creation and elimination of temporary objects" in SIMSCRIPT jargon), and statistical analysis.

SIMSCRIPT I is fairly easy to learn, but there are some difficulties associate with an actual application. For example, the variables in the user-supplied routines need to be precisely defined at the beginning of the program, and the coding requirements for this is both exacting and error-prone. The assignment

of initial values to the variables at the start of the simulation also presents another major source of error. Since the format rules for data entries allow for less flexibility than in FORTRAN, considerable time and effort are required to "debug" and run even simple simulation models. The extra core-storage required by the SIMSCRIPT compiler and the added compiling time also contribute significantly to the computational costs. A major shortcoming is the lack of a separate information-handling scheme which could deal with the manipulation and storage of simulation data arising from one simulated event when that event can cause instantaneous interactions within all elements of a large, complex simulated system. (Eg., such an event could be an explosive chemical reaction with the resulting chain formation of intermediate free radicals and end-products in a brief instance.)

GASP II[4,8] (General Activity Simulation Program) developed at Arizona State University is the offshoot of the original GASP from U.S. Steel. The program is a discrete-event oriented simulation executive which is entirely FORTRAN based and therefore requires no separate compiling system other than that for FORTRAN IV. Its modular structure, made up of several interactive subprogram blocks (or "modules"), allows it to be easily maintained, modified, or extended to meet the particular needs of different applications. The basic concept of a discrete-event simulation executive system utilized in GASP is similar to that employed in SIMSCRIPT, and programs written in GASP can be readily converted to SIMSCRIPT and vice-versa, without any difficult changes.

Experiences with GASP show that it is a compact system requiring relatively little core-storage. The information-handling scheme pertaining to the control of the simulated discrete events has been fully explained in the

documentation by Pritsker and Kiviat[8]. Its structure is made up entirely of FORTRAN subroutines. The actual mechanics involved, in terms of computer programming technique, consist of the storage, retrieval and updating of all data entries relating to the simulated system in a large, general two-dimensional matrix file, according to some pre-set logical order. To keep track of the dynamics through simulated time, an elaborate set of relative position pointers is used to locate the exact storage area of any one piece of information pertaining to either an event or an entity in the simulation. Routines for statistical sampling and analysis, simulation data collection and reporting are also provided as in SIMSCRIPT. GASP differs in that these routines are not in the form of inaccessible "black boxes", but are presented as distinct and separate modules, which could be altered, replaced or eliminated altogether from the total executive system. As with SIMSCRIPT, the main disadvantage with GASP is that there are no provisions for a framework wherein simultaneous complex interactions between several elements of the simulated system caused by one or more events can be easily modeled and represented by computer programming. This seriously limits the application of the executive to only simpler problems describing smaller systems. Larger systems containing several interacting elements or entities require that some means be provided for the computation of the converged solution of the simultaneous interactions. The creation of DESS is therefore in response to this need and offers an alternative to redundant exhaustive programming.

Section B: DESS (Discrete-Event Simulation System)

(i) What it Is

DESS, the acronym for Discrete-Event Simulation System represents the

final development stage of the simulation executive program resulting from a merger of GASP II[8] (General Activity Simulation Program), which provides the fundamental framework for maintaining the chronology of simulated events, and GEMCS[3] (General Engineering and Management Computation System), which forms the basis of the scheme used for sequential calculations involving contempraneous activities in the system under study. The DESS program is written in FORTRAN IV and is entirely based on a modular structure consisting of several computer subroutines, each of which serves a specific function. Communication in between subroutines is effected through the common data storage areas established over the whole program.

The basic concept of DESS concerns the description of the state of the system under simulation over a certain period in time. The description of the system state at different points in time, which could be separated by varying time-intervals, is accomplished by a provision of state parameter values, that do not change during a simulation run, and state variables, which have some initial values that change with time depending on what has previously occurred. The mechanism that controls the advancement of discretized time and the scheduling of events during the simulation run is built into the routines modified from GASP. This moves the simulation model through time to produce dynamic behaviour over discrete points in time, and also selects the appropriate events which have been scheduled. According to the logical relationships contained in the model, a set of state variables can thus be computed for each specified time instance, and this will be continued until the simulated time is exceeded. The GASP subsystem in DESS also maintains a dynamic filing array called NSET, which registers a calendar of all scheduled events, and orders the

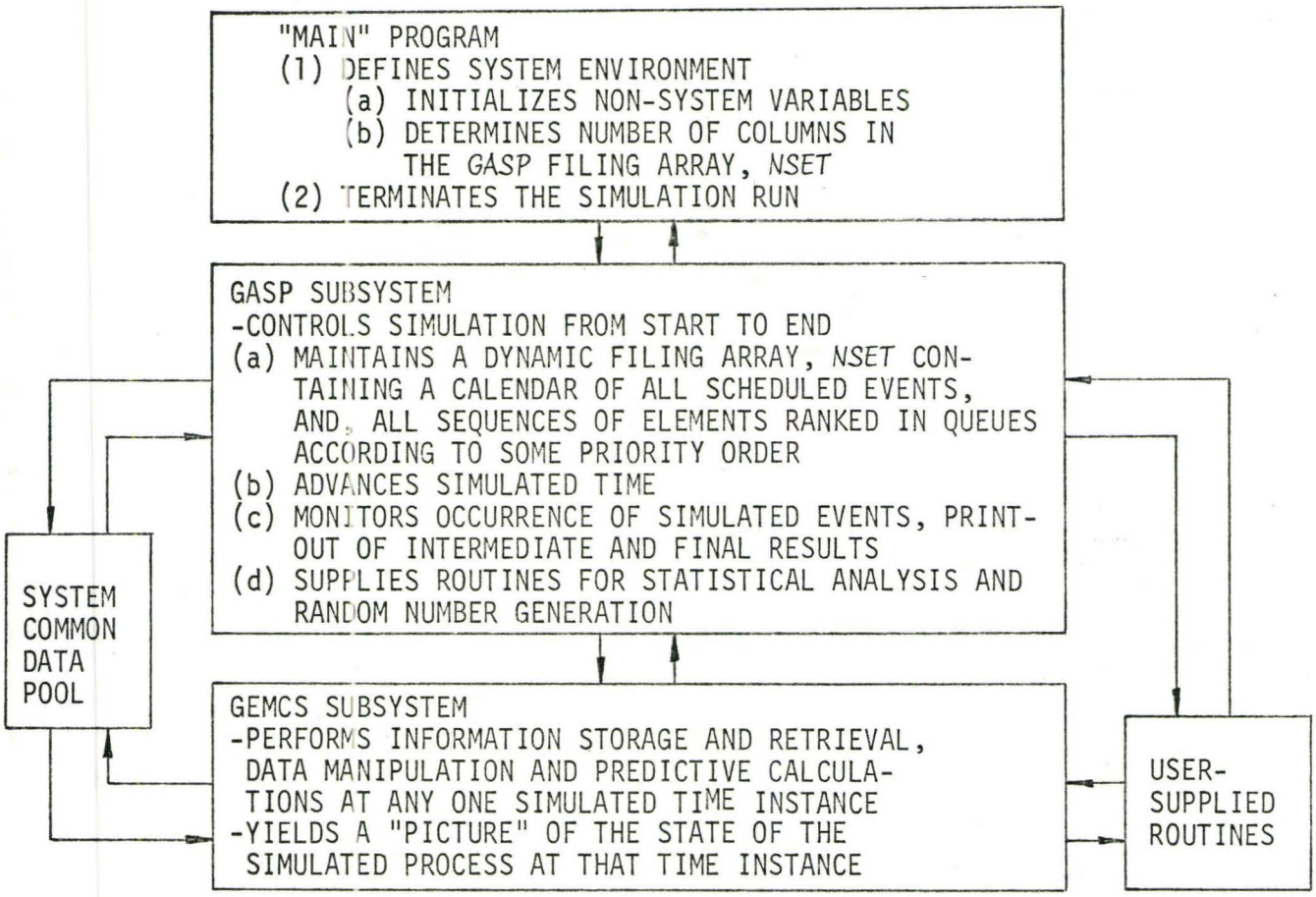
sequencing of entities ranked in queues according to some predetermined priority order (eg. first-in-first-out, last-in-first-out, etc.)

The GEMCS subsystem in DESS is the descendant from the original GEMCS program created at McMaster University by Johnson, et al[3]. Its function in the simulation executive is to provide a means of performing information storage and retrieval, data manipulation and predictive calculations at any one fixed simulated time instance. It therefore serves to yield a "picture" of the latest state of the system which is a result of the last simulated event. The method employed in the GEMCS subsystem to calculate values representing simultaneous activities assumes knowledge of certain unknown variables which enables direct sequential computation of all remaining unknown variables according to some prescribed convenient mathematical order. This is followed subsequently by several iterations of the same computing sequence, each time using the newly calculated values for the starting variables, until some criterion for convergence is satisfied or approached. The information relating to each part of the simulated system which is manipulated within the GEMCS framework is stored either as parameter values in the EN and EEN vectors that simulate the operations of computer disk storage, or as variable values in information streams which interconnect each part of the simulated system. These stream variables are contained in three dynamic matrix tables (SI, SN, and SO).

The system common data pool which allows communcation between each of the modular subprograms in DESS can be extended into user-supplied routines, so that system variables are made available if required. In addition, the dynamic filing array, NSET, is accessible not only to the GASP and GEMCS subsystems but also to the modular subroutines programmed to represent events, process equipments or other entities of the simulated system.

FIGURE 1

The DESS Algorithm



(ii) What it Does

The algorithm in Figure 1 shows the sequence of execution in the DESS program. In a simulation run with DESS, initial control is given to a "main" program, which is written by the user to set the required number of columns in NSET by the non-executable "DIMENSION" command, and to initialize any non-system variables if required. It should then transfer control of the simulation to DESS by calling the GASP subsystem.

When execution is passed on to the GASP subsystem, the routine DATAN is called to initialize all GASP variables and the array, NSET. It then withdraws information on the first scheduled event from the event-file (which thus must contain at least one entry, already read in from the data set), and proceeds as the event commands.

Using the information on the scheduled event, it goes into subroutine EVNTS to select the appropriate "event-subroutine" for the particular time instance (which is a fixed point in time). Execution can then be transferred from the EVNTS subroutine to other subroutines making simple, independent changes, or to the GEMCS subsystem to solve the information network of the process. (The process can be alternatively described as the system representing the simulated activities.) The GEMCS subsystem then yields a "picture" of the state of the simulated process at that particular simulated time instance.

The subroutine DLOAD1 of the GEMCS subsystem is only called (or brought into execution) when the "present" simulated time is not greater than the simulated time at the beginning of the simulation run by 0.1 unit. Hence, GEMCS variables are initialized in DLOAD1 at the start of the simulation run. If the initial simulation time step-size is too small (ie. of the order less than 0.1), execution

would immediately terminate on encountering an "END OF FILE" card in the computer program. In that case, two alternatives are available:

(1) change the value of 0.1 in the system routine (DLOAD1) to a value less than the time step-size, or,

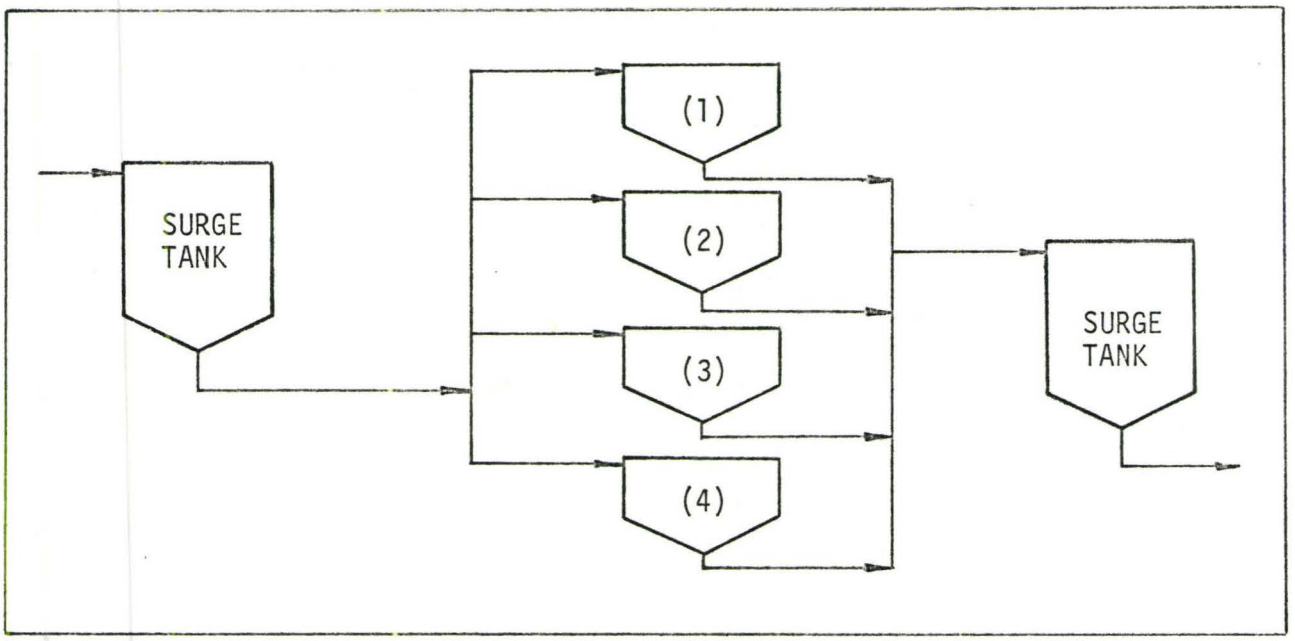
(2) increase the time step-size to a value greater than 0.1.

The other functions of the GEMCS subsystem include the scheduling (or causing) of endogenous (i e. internally caused) events for a future time. When such events are scheduled, they are entered into the event-file which is stored in NSET with values supplied for the time each event is to occur and the code number for the event. Such internal causation of future events can be carried out in a modular subroutine which has access to NSET and other system variables. Logically, events in the event-file (assigned as File No. 1 in NSET) must be ranked in the order of lowest time value first. (The time value, which is the time of scheduled occurrence of the event, is stored as the first attribute of the entry.) Hence, one can cause further calculations to be made at the same instance in simulated time by making a new entry in the event-file with the time value equal to the present time.

The GEMCS modular framework is also capable of storing and removing entities in and out of queues using parts of the GASP subsystem. (A queue or waiting line is here defined as a "line" of objects, events, customers or transactions waiting at a service facility.) When the state of the simulated system is advanced through time, in order to compute the present conditions (or state of the system), certain past information may be required (such as the need for information on the last day's performance in order to improve today's). This past information could be previously stored in a queue according to some

FIGURE 2

FLOW SCHEME IN THE BATCH-PROCESSING DEPARTMENT OF A HYPOTHETICAL CHEMICAL PLANT



ranking priority, and as each piece of information is inserted or withdrawn, the GASP subsystem is used to update the file. The actual entry and withdrawal could be performed in the user-supplied subroutines which could obtain input information from the system common data pool. The insertion of entries into the queues of entities and future events is necessary if the information would be required at a future time. This is accomplished with communication between the GEMCS and the GASP subsystems as directed in the user-supplied subroutines, with the aid of the common data pool.

The GASP subsystem besides controlling the dynamic filing array in NSET, the selection of simulation events and the advance of simulated time, also could be used for the collection and generation of run statistics and reporting of simulation results. Other routines in the subsystem facilitate the generation of values from different random sampling distributions, and these are all available to the user.

To illustrate the concept of "chronology of simulated events" maintained in the GASP subsystem, and "contempraneous activity" modeled with the GEMCS routines, consider the simple process flow diagram in Figure 2.

It shows the flow scheme in the batch-processing department of a hypothetical chemical plant. The production schedule calls for the manufacture of a certain quantity of product X. This product is easy to run, and cycle times are predictable. The time required for the crew to service one reactor (i e. load, unload and clean) is known to be normally distributed with a certain mean and standard deviation, and the reaction time can be calculated from a mechanistic model based on theory and observed plant data. Labor costs and the costs of reactor operations are known. It is desired to evaluate the optimum

production schedule for a given year using the dollar as a measuring criterion. Two, three or four reactors may be used. Apparently, the reactors and crew cannot be utilized to their maximum at the same time. Hence, the amount of idling time allowed for either will determine the total increase in costs accrued to that particular production schedule. In a simulation run testing any schedule, the chemical reactions constitute "contempraneous activity", and reaction times are computed using the GEMCS scheme. The servicing of the reactors and the setting of valves are time-dependent events which are scheduled and maintained within the GASP framework. These and other similar programmed "actions" over time represents a "chronology of simulated events".

(iii) The Data Format Used With DESS

The first part of the DESS data set consists of the data for the GASP subsystem and the second part is for the GEMCS subsystem. All numbers are read in 5F12.0 data field and alphanumeric characters in 20A4 field (GASP) or 18A4 field (GEMCS). The detailed data formats used are given in the following two pages.

GASP DATA SET

Project Title (1 card - 20A4 field)					
NPROJ	MON	NDAY	NYR	NRUNS	
NPRMS	NHIST	NCLCT	NSTAT	ID	
IM	NOQ	MXC	NEPRM	SCALE	
NCELS(1)	NCELS(2)	...	NCELS(NHIST)		(omitted if NHIST = 0)
KRANK(1)	KRANK(2)	...	KRANK(NOQ)		
INN(1)	INN(2)	...	INN(NOQ)		
PARAM(1,1)	PARAM(1,2)	...	PARAM(1,NEPRM)		(omitted if NPRMS = 0)
PARAM(2,1)	PARAM(2,2)	...	PARAM(2,NEPRM)		
:	:		:		
PARAM(NPRMS,1)	PARAM(NPRMS,2)...		PARAM(NPRMS,NEPRM)		
MSTOP TFIN	JCLR JSEED	NORPT	NEP	TBEG	
±1.0 0.0	(-ve for file initialization; +ve for skipping initialization) 0.0 (dummy values				
FILE NO. ATRIB(1)	ATRIB(2)	....	ATRIB(IM)		At least one entry for File No. 1 must be inserted.
FILE NO. ATRIB(1)	ATRIB(2)	....	ATRIB(IM)		
: FILE NO. ATRIB(1)	ATRIB(2)	....	ATRIB(IM)		
0.0 0.0	(to terminate GASP data set read in) 0.0 (dummy values)				

All the variables are fully explained in the GASP documentation[8] except for NEPRMS which has been newly introduced to limit the number of columns in PARAM.

GEMCS DATA SET

GEMCS subtitle - printed before the rest of the data set is read in (2 cards - 18A4 field)				
KPRNT (1)	KPRNT (2)	KPRNT (3)	KPRNT (4)	KPRNT (5)
KPRNT (6)	KPRNT (7)	KPRNT (8)	KPRNT (9)	KPRNT (10)
NCALC	NOCOMP (if NCALC is - ve, iterations are allowed, if NCALC is + ve, there are no iterations)			
LLST (1)	LLST (2)	....	LLST(NCALC)	
MSN	(if MSN is -ve, output stream information printed after each module is called)			
NS (1)	NS (2)	....	NS(MSN)	
NSR				
SN (1,1)	SN (1,2)	....	SN (1, NOCOMP+5)	
SN (2,1)	SN (2,2)	....	SN (2, NOCOMP+5)	
SN(NSR,1)	SN(NSR,2)	....	SN(NSR,NOCOMP+5)	
NOE				
1st set: EN (1)	EN (2)	EN (3)	EN (4)	EN (5)
EN (6)	....	EN (EN(3))		
2nd set: EN (1)	EN (2)	EN (3)	EN (4)	EN (5)
EN (6)	....	EN(EN(3))		
⋮				
NOEth set: EN (1)	EN (2)	EN (3)	EN (4)	EN (5)
EN (6)	....	EN(EN(3))		

There must be "NOE" sets altogether

The variables are the same as those explained in the GEMCS documentation[3]. Two data cards are made available for the subtitle to allow more information to be printed. For greater convenience, the executive was modified so that NCALC has to be set negative for iterative computations among the GEMCS modules.

CHAPTER 2

FIRST CASE STUDY: DISCRETE-EVENT SIMULATION OF SUCCESSIVE "STEADY-STATES" IN THE PRECIPITATION CIRCUIT OF THE BAYER PROCESS

Section A: Introduction to System

In this first, simple case study, the production control aspects of the precipitation circuit in the Bayer Process are represented as a set of linear algebraic equations, which are solved at various simulated process times after certain variables have been assigned some pre-determined values from samplings of normal distributions with different statistical parameters. The model makes use of the dynamic filing array, NSET, and the system common data pool to allow interaction within the simulated process over the whole range of time. This ability allowed the predictive properties governing the process (such as the control information feedback and the recycle of seed-material in the precipitation section) to be easily modeled in the simulation.

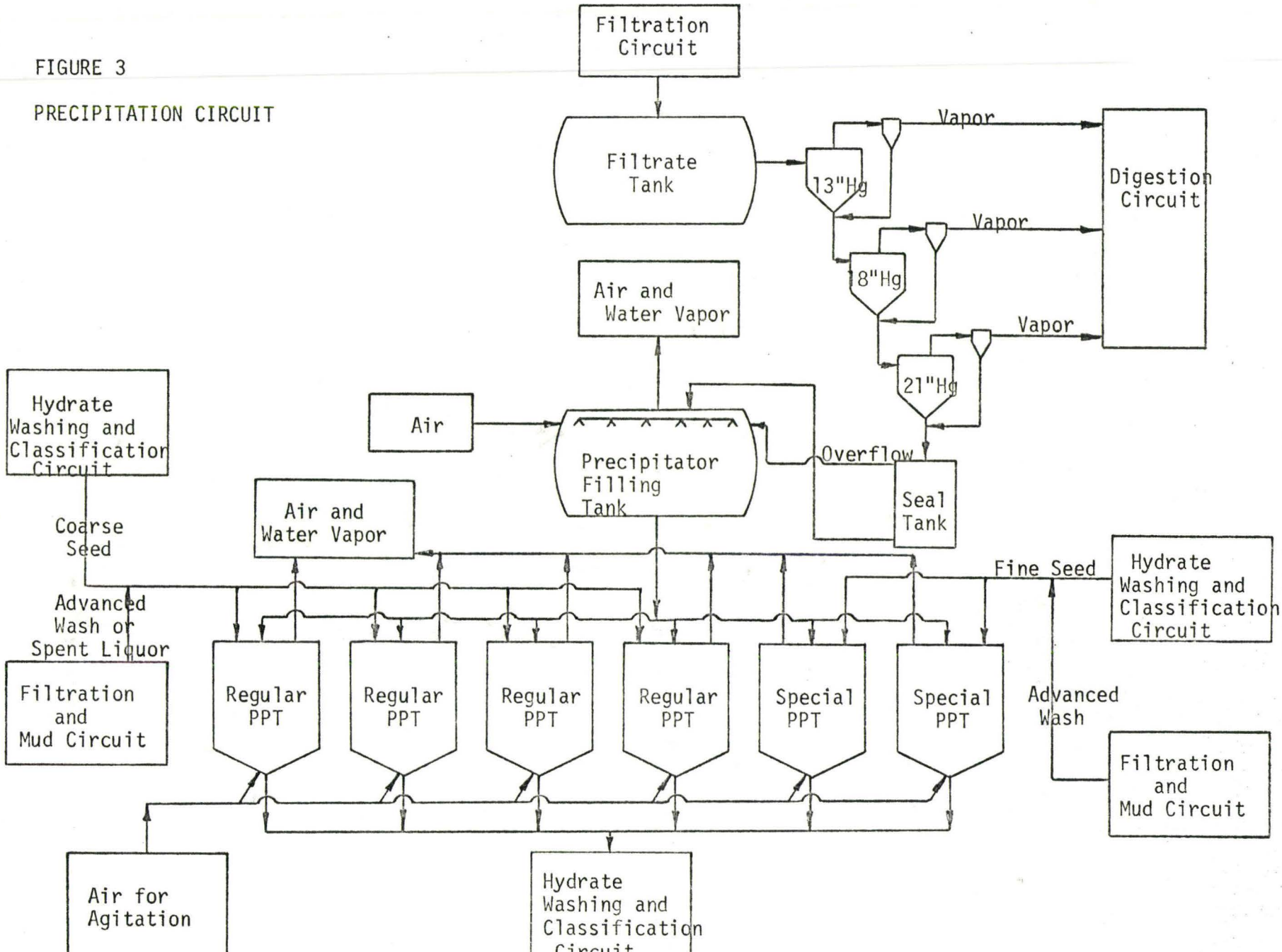
The Bayer Process[1,2,7] for extracting alumina from bauxite was devised by Karl Josef Bayer. The process is based on a steep solubility curve, eg. at 300°F a 3N NaOH solution will dissolve almost twice as much alumina as one at 150°F.

$$Al(OH)_3 + OH^- \underset{\text{heat}}{\overset{\text{cool}}{\rightleftharpoons}} Al(OH)_4^-$$

The bauxite is slurried with caustic solution, and heated in a series of digester reactors (80-90 psig) for 35-40 minutes. Then the slurry is treated to remove

FIGURE 3

PRECIPITATION CIRCUIT



the "red mud" (mostly clay minerals and iron oxide), and the liquor is cooled to 155°F. It is seeded with previously precipitated $A\ell(OH)_3$ and the excess alumina crystallizes on the seed (mineral from "gibbsite") during 24 to 36 hours. After separation of the precipitate, the used or "spent" liquor is recycled to the digester, and the solids are washed. Calcination gives very pure, white $A\ell_2O_3$, suitable for electrolytic reduction to aluminum by the Hall-Héroult process.

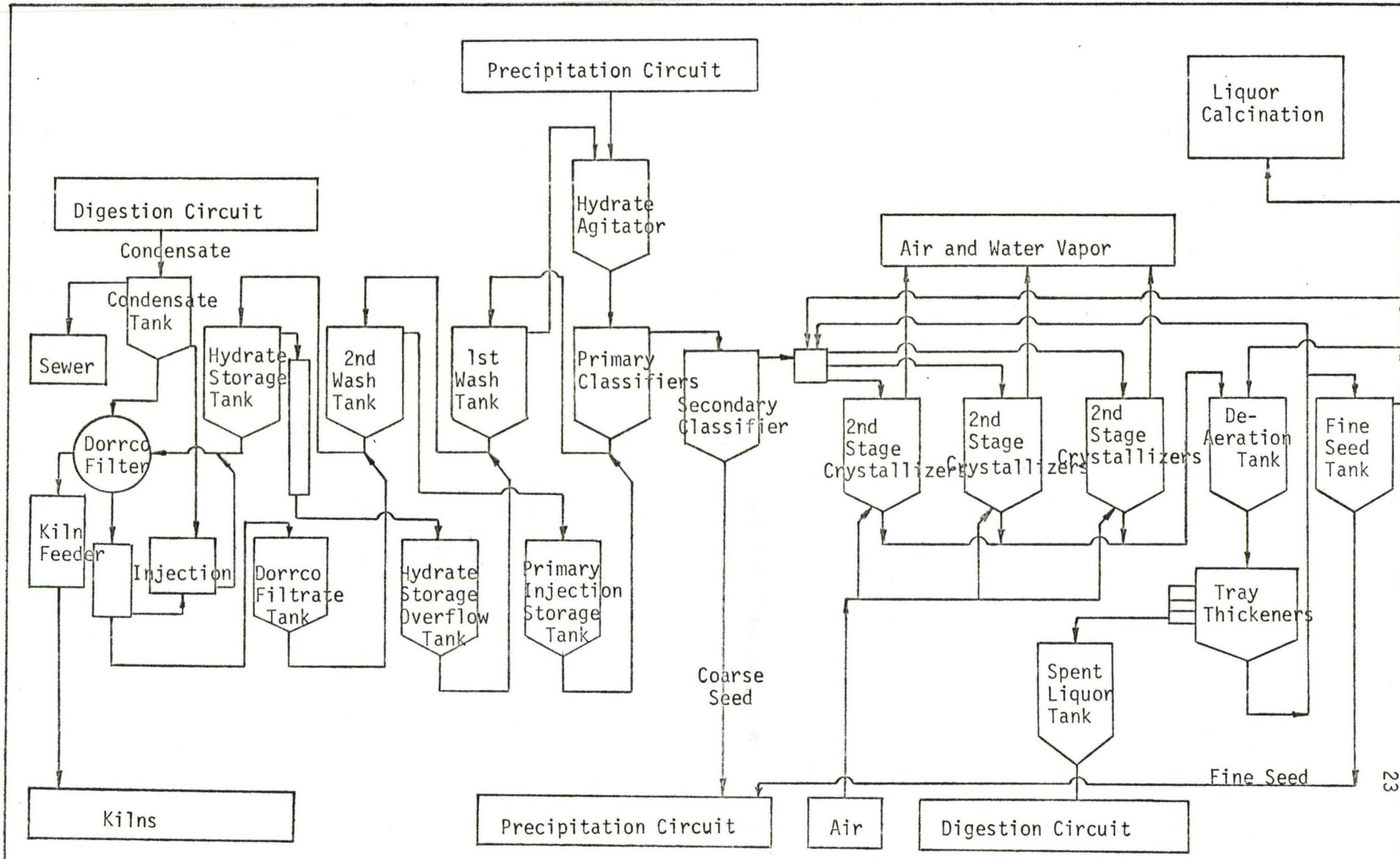
Figure 3 shows a schematic representation of the precipitation circuit in the Bayer Process. The temperature of the filtrate from the filtration circuit is approximately 208°F (in the filtrate tank) and this is reduced to about 155°F to achieve the supersaturation necessary for re-precipitation as alumina trihydrate in the precipitators. This cooling is achieved by flashing the liquor in three successive stages operating at approximately 13-in., 18-in., and 21-in. of mercury vacuum, respectively. The vapor from these flash units is used to preheat the spent liquor recycles to the digester circuit, in a train of heat exchangers. This flashing operation generates vapor amounting to about 4% of the incoming filtrate and reduces the temperature to about 170°F. The final tempering is achieved in a spray cooler into which air is injected to obtain approximately a 10°F drop and evaporate approximately 1% of the water from the liquor.

The supersaturated liquor with a ratio approximately 0.640 (ratio is defined as weight of anhydrous alumina/weight of sodium ion content expressed as equivalent anhydrous sodium carbonate) is fed to batch precipitators which are 62.7 feet high and 26.0 feet in diameter. Agitation is achieved by blowing air up a stand-pipe in the reactor. This "lift-pump" effect circulates the magma

from the bottom to the top of the precipitator. The "regular" precipitators are charged with supersaturated liquor and a slurry of coarse seed material with 50% solids is added (17% by weight of the coarse seed is below 44 microns in diameter). After about 24 hours, at which time the temperature would be reduced to about 145°F and the ratio to about 0.345, the charge is dumped into a de-aerator surge tank (also called the hydrate agitator). The "special" precipitators ("usually" consisting about 20 of the operating precipitators out of a total of 59) are charged with the same amount of liquor material, except that the seed slurry charge is much finer and only amounts to about 20% solids by weight. (Approximately 75 weight precent of the fine seed is below 44 microns, and 10 weight percent below 10 microns in diameter.) The ratio of the flow leaving these precipitators is higher than that of the flow leaving the "regulars". The measured value in this case ranges around 0.360 - 0.365. The material from the "special" precipitators is also transferred to the de-aerator tank or hydrate agitator.

All operations downstream from this point are continuous. The alumina slurry is transferred to the primary classifier, the underflow from which proceeds through a three-stage countercurrent washing operation. The overflow serves as feed for the secondary classifier, the underflow of which is drawn as needed to be used as coarse seed for the "regular" precipitators. The overflow is mixed with recycled liquor from the fine seed tray-thickeners located downstream, before it is pumped continuously to a number of crystallizers (also referred to as second-stage precipitators) to give a 6 to 8 hour residence time there and a concentration of about 100 gm./litre solids. The size and shape of these crystallizers is identical to the primary batch precipitators. The slurry from

FIGURE 4 HYDRATE WASHING AND CLASSIFYING CIRCUIT



these crystallizers goes to continuous tray-thickeners where it is concentrated to about 250 gm./litre. The underflow is then transferred as needed, mixed with advanced wash liquor to yield 20% solids by weight, and used as seed slurry to the "special" precipitators. Some of the overflow from the fine seed holding tank is returned to the input to the second stage crystallizers; some of it is removed as a purge stream to be calcined to remove organic material. Some of the underflow from the tray-thickeners may be returned to the input to the secondary crystallizers. The overflow from the tray-thickeners becomes recycled spent liquor and is returned to the digester circuit. Figure 4 shows a schematic representation of the hydrate washing and classification circuit. (Owing to the seed material recycles, the operations in the precipitation circuit interact very closely with conditions in the classification section.)

A specific objective in this case study is to investigate the effects of the variations of the operation control variables, the uncertainties in the system parameters, the randomness of external environmental factors, the fluctuations of input flow conditions to the circuit,on the output variables,and the ramifications in modeling the control aspects as a set of "steady-state" equations. The feedback information from the continuous downstream process which affects the adjustment of control variables in the batch circuit, and the interaction between the actual physical flow conditions downstream and the input flow variables caused by different material recycles in the process are also studied in this case. The degree of significance of the actual physical lags in the recycle and the lags in the information feedback governing the setting of the control variables are to be estimated. In this respect, discrete-event simulation is performed with the division of the process time into several subsets

of non-uniform time-intervals. This implicitly assumes that control of the process operations with the model could be effectively maintained using the prediction of discrete-events one or more time-steps ahead.

Section B: Discussion of the Model

This simulation study concerns the application of the set of "steady-state" control equations, which govern the flow regulating operations in the precipitation circuit, to an estimation of the dynamic effect of the flow variabilities and parameter uncertainties in the circuit over time, and the usefulness of such a model. Consider the set of equations which necessarily describes the mean or average conditions in the plant.

(1) $VSEEDS*NS = X*THETAS$

(2) $VSEEDR*NR = Y*THETAR$

(3) $\frac{VFILTS*NS}{THETAS} + \frac{VFILTR*NR}{THETAR} = \underline{Z}$

(4) $VFILTS + VSEEDS = VS$

(5) $VFILTR + VSEEDR = VR$

(6) $THETAS = THETAR$

(7) $VS = VR$

(8) $NS + NR = NT$

(9) $\frac{NR}{NS} = Q$

(The notation used is explained fully in Appendix 2.) It is noted that six mutually independent variables must be assigned values in order to solve the equations. X, Y and $\underline{Z}$ represent the average flowrate values that must exist over

the period of the mean residence time of the reaction contents in the precipitators. This residence time value is governed in the same way by the operations for both the "special" (THETAS) and the "regular" precipitators (THETAR). Therefore, it is assumed that the mean value that can be used for the average state calculations must be the same for THETAS and THETAR. Hence equation (6) is justifiable. The same argument may be used to reason the assertion in equation (7) that the total volume of the reaction contents in both types of precipitators ("specials" and "regulars") are equal. In the model, it is assumed that the plant operator actually has direct control over the operations so that he can choose to set the average precipitator volume and the average number of "special" and average number of "regular" precipitators to be used over the period of the mean residence time. (Calculations for the average state of the system are performed at time points separated by periods equal to the changing mean residence time values.) Obviously, the mean flowrates into the circuit cannot be directly controlled from within, and operations must be adjusted to suitably accommodate the process material entering the circuit. In actual plant practice, the total volume that is filled in any one precipitator is fixed and any random variations that may occur are observed to be small. Therefore, the two key controlling variables are the number of "special" precipitators and the number of "regular" precipitators used in the circuit at any one time. However, in spite of the original simplifying assumption, the plant operator does not have direct control of the actual number of precipitators used. In actuality, the effective number of precipitators used at any one instance (both "special" and "regular") is, in itself, a function of the incoming flowrates and the constraints on the flow pattern within the circuit caused by

the physical plant layout and the availability of "free" (ie. empty) volume, both in the precipitators and the surge tanks, (which cannot be altered or directly controlled), and also of certain plant operating policies governing the filling and emptying of the precipitators, (which are directly determined and controlled by plant operating personnel). Apparently, certain factors governing this function (eg. the availability of "free" volume) are a result of past operations and conditions in the plant. Also, the plant policy for filling and emptying at any one time must involve some forecasting of future product requirements, as well as a consideration of the maximum utilization of every cubic foot of available precipitator volume, and the minimization of the levels in the surge tanks and their fluctuations. It is therefore appreciated that in assuming the number of "special" and "regular" precipitators to be simply related to the present seed flows and of certain past information (on previous seed flowrate and number of precipitators used) the model oversimplifies the real situation, although it yields some insight into the nature of the average decision-making needs in the plant. This aspect of assigning precipitators for the batch operations in the circuit and the factors determining the actual number of "special" or "regular" precipitators used at any one time is the subject of the second case study and is further discussed in Chapter 3.

In the present case study, the plant operator is assumed to be able to set the values for the number of "specials" and "regulars" directly using simple, linear algebraic equations. (Refer to Subroutine OPERAØ1 in Appendix 2.) To account for the uncertainty in forecasting future product requirements and the variations that might arise from physical plant constraints limiting the operator's decision, the values for the number of "specials" and the number of

"regulars" calculated from the linear equations in the subroutine OPERAØ1 are taken to be the means of normal distributions from which samplings are then obtained to represent the average values for the period. Given values for the number of precipitators to be used and the flowrates entering the circuit, the other variables describing operations could then be computed from the other equations contained in the model.

Section C: Discussion of the Simulation Results

The simulation results obtained in this case study showed that the average conditions in the plant, represented by the state variables which are contained within the set of "steady-state" equations, vary only over a small range of values. The significance of this, however, cannot be accurately gauged because of two inherent shortcomings in the simple model. These are:

(1) the arbitrary assumption that flow variabilities are predictable and follow the simple pattern of a normal distribution; and,

(2) the assumption that the average number of "special" and "regular" precipitators at a given instance can be estimated at all.

Of these, the first difficulty is recognised before and the assumption is made because even though the maximum flow fluctuations in the plant had estimated values, it is not known how these fluctuations occur. (Ie., the pattern of the fluctuations is unknown.) The second assumption is subsequently found to be seriously in error and this is the major finding in this case study. Before discussing that point in detail, a summary of the simulation data obtained is given next.

The liquor flowrate entering the precipitation-circuit has a mean flow

of 48000.0 cu.ft./hr. with a standard deviation of 1000.0 cu.ft./hr. (The maximum is 53000.0 cu.ft./hr. and the minimum is 43000.0 cu.ft./hr.) The number of "special" precipitators used over the period of simulated time has a mean value of 22.0. The initial average value given in the data set is 20.0 and the standard deviation 5.0. (The maximum is 35.0 and the minimum 5.0.) The number of "regular" precipitators used over the period of simulated time has an average value of 38.0. The initial value in the data set is 40.0 with a standard deviation of 5.0. (The maximum is 55.0 and the minimum 25.0.) The settings of the mean number of precipitators are governed by the amount of fine and coarse seed produced. Hence, the mean number of "special" precipitators is increased if the amount of fine seed product is increased relative to the amount of coarse seed product; the number of "regular" precipitators is decreased proportionately. The residence time of the charge in the precipitators over the period of simulated time has an average value of 27.2 hours. The total filled volume of one precipitator averages at 37005.0 cu.ft. (The average value given in the data set is 37200.0 with standard deviation 2300.0, maximum 44000.0 and minimum 30000.0). The calculated values for VFILTS (filtrate volume in "specials") have a mean of 22830.5 cu.ft., VFILTR (filtrate volume in "regulars") a mean of 24593.1 cu.ft., VSEEDS (fine seed volume in "specials") has an average value of 14174.5 cu.ft., and VSEEDR (coarse seed volume in "regulars") an average of 12411.9 cu.ft.

It can be observed from this that the process appears to be fairly stable in spite of the flow variations and the uncertainty in the set values for the number of "special" and "regular" precipitators. In plant experience, conditions are known to be less predictable and "well-behaved". Apparently, there must be

factors that account for this inconsistency between observed reality and the results from the simulation model.

First of all, the state variables calculated for each time instance in this model do not represent the values of those variables that would be calculated in a true dynamic model. The values from this model represent the averages over the time period which has been arbitrarily chosen to lie between the two time points when the system state is computed. This method of computing averages is comparable to viewing the process at successive "steady states", each one of which is equivalent to the average state of the system from the last time point up to the latest time point, when the calculations are made. This then is the implication of the set of control equations incorporated in the model. Operations within the precipitation circuit are not continuous, but are batchwise in nature. Therefore, all variables in the system constantly assume different values and they do not necessarily nor do they all change values at the same point in time. In order to simulate these operations in a successive "steady state" manner, it is apparent that the averages of the flow variables and the set operating conditions must be accurately known or estimated.

The flows of the seed charge into the circuit are not continuous since seeding is a batch operation. Consequently, there is virtually no means of effectively estimating the average values of these flowrates for any period of time. Apparently, there is no available plant data which could be used to evaluate these variables. Some of the difficulties involving the estimation of the number of "special" and the number of "regular" precipitators that are being used in the process were mentioned in the previous subsection. Basically, however, the problem lies with the impossibility in estimating the effects caused

by the very real presence of precipitators that are not "circulating" (ie. carrying full reacting contents) either as "specials" or as "regulars", but are being filled, seeded, discharged, or cleaned. Altogether, the effects of this combination of precipitators at different stages of the batch processing scheme could probably be represented by an equivalent but fictitious number of "specials" and "regulars", but, again, this is not a simple problem that could be intuitively solved. Hence, the conclusion arrived at in this case study is as follows. The fallacy in this model and in any model which makes use of representative "steady states", is not that average conditions do not exist in the precipitation circuit, but, is that these conditions are not readily estimated or calculated without a study of the detailed batch operations themselves and the actual hour-by-hour change of the values assumed by the discrete control variables of the process. Since none of these variables can in reality be directly adjusted, a relationship between them and the directly controlled operations in the plant that govern their values must exist; this latter is the subject of the next case study which employs a model of the batch operations in the precipitation circuit to simulate the dynamics of the system.

CHAPTER 3

SECOND CASE STUDY: DISCRETE-EVENT SIMULATION OF THE DYNAMICS OF THE FLOW-SCHEDULING OPERATIONS IN THE PRECIPITATION CIRCUIT OF THE BAYER PROCESS

Section A: Description of the Operations

In the precipitation circuit of the Bayer Process at the Arvida works, there are altogether 59 precipitators, all of which are operated in parallel to one another. These precipitators are classified into five different groups according to their mode of operation and to the one pump serving each group.

The first group consists of 9 precipitators which can be operated only as "specials", ie., they can be charged only with fine seed material. The second group has 14 precipitators, the third has 10, and the fourth has 13 precipitators; these can be operated either as "specials" or as "regulars", ie., they can be charged either with coarse or with fine seed material. The fifth (ie. last) group is made up of 13 precipitators which can be operated only as "regulars". Only one pump is available to each group for the discharge of the precipitator contents (called "draw-off"). Therefore, at any given time, no more than one precipitator in any one group can be emptied. The discharge from any precipitator is pumped into two hydrate agitators with a total capacity of 66588 cu.ft. (The dimensions of each hydrate agitator are the same as the dimensions of a precipitator, with a cross-sectional area of 531 sq.ft., a total height of 62.7 ft. and a volume of 33294 cu.ft.) The draw-off of each precipitator can vary between 200 cfm. and 350 cfm. The value of the draw-off flow depends on the

levels of the hydrate agitators. Normally, there are three precipitators drawing-off at any given time and each of these three precipitators empties with the same flow; so all draw-offs are equal.

In the plant, the underflow of the hydrate agitator is changed by manual operations depending on the level of the spent liquor tank, which is a tank after the classification circuit. The value of this flowrate varies around 1100 cfm. The pregnant liquor from the filtration circuit has a flowrate of 800 $\pm$ 50 cfm. (The variations may be assumed to be a random number from a uniform distribution, ie. they are purely random values lying within an upper and a lower limit.) It enters two filling tanks in equal separate streams (400 $\pm$ 25 cfm.). (For filling tank no. 1, the cross-sectional area is 1964 sq.ft., the height 50 ft., and volume 98200 cu.ft.; for filling tank no. 2, the cross-sectional area is 1257 sq.ft., the height 40 ft., and the volume 50280 cu.ft.) The optimal policy is to maintain a zero level in any of these two tanks.

In regards to the filling operations for the precipitators, there are different policies in the plant, but the so-called 1/1 and 2/1 policies are probably the most used. The 1/1 policy consists of alternately filling one precipitator as a "regular"; and the next as a "special", then a "regular", etc. The 2/1 policy requires every two consecutive, "regular" fillings to be separated from the next two by a "special" filling. The type of policy in operation at any time depends on the quality of the solid products downstream. Regardless of whichever policy is used, certain rules and restrictions must be observed in the filling operations. At any instance, no more than six precipitators can be filled with pregnant liquor or charged with seed. The seed charge is added

after the pregnant liquor, and not simultaneously. At most, only two precipitators can be filled with coarse seed and one with fine seed, at the same time. There are altogether a total of six inlets to the circuit, a maximum of three being available for liquor-filling, a maximum of two for coarse seed charging and one for fine seed charging. Hence, whilst three or fewer precipitators are being seeded, a maximum of three others can be filled simultaneously. Filling tank no. 1 can only supply liquor to groups one, two, three and four, but not to group five. Conversely, filling tank no. 2 can only supply liquor to groups two, three, four and five, but not to group one. This gives rise to five possible cases where the filling of precipitators must be cut or postponed, if at the end of the filling of a precipitator:

(1) there are no empty precipitators in any group;

(2) the next precipitator to fill is a "regular" and there are no empty precipitators in groups 2, 3, 4 and 5;

(3) the next precipitator to fill is a "special" and there are no empty precipitators in groups 1, 2, 3 and 4;

(4) the next precipitator to fill is a "regular" and there is an empty precipitator only in group 5 and the filling tank no. 2 already fills two precipitators; (the filling tank no. 2 cannot fill more than two precipitators at the same time and the filling tank no. 1 cannot fill in group 5);

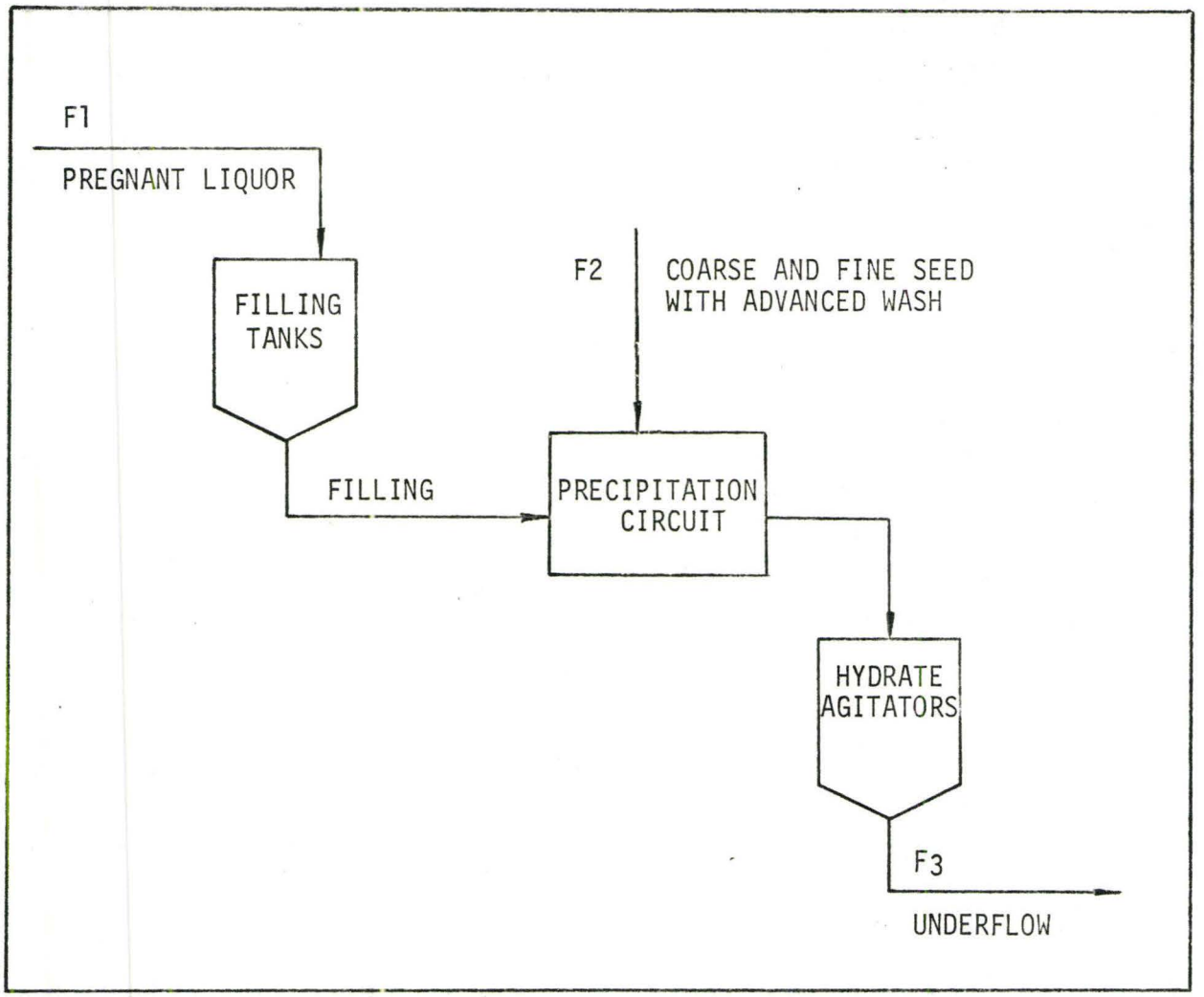
(5) the next precipitator to fill is a "special", there is an empty precipitator only in group 1 and the filling tank no. 1 already fills two precipitators; (the filling tank no. 1 cannot fill more than two precipitators at the same time and the filling tank no. 2 cannot fill in group 1).

The dimensions of a precipitator are 62.7 ft. in height and 26 ft. in diameter. For a "special", the precipitator is filled to 49.7 ft. with pregnant liquor; then 10 ft. of fine seed charge is added. For a "regular", the height of liquor is 46.7 ft. and the height of coarse seed 13 ft. The fine seed mean flowrate (solids and diluting liquor) is 115 cfm. and the coarse seed mean flowrate (solids and diluting liquor) is 75 cfm. The underflows of the filling tanks or the filling flows are 400 $\pm$ 25 cfm. if the filling tanks are near empty. However, if for one reason or another, the filling flow is cut, then the level of one filling tank will rise. If this happens, this filling tank is subsequently emptied by increasing the new filling flow by 5 cfm. per foot of height in this filling tank.

The "drawing-off" operations are also governed by certain policies related to plant physical limitations and policies used to control production. At most, only three precipitators in the circuit can be emptied at any given time, and only one must be in a given group even though more than one in that group may be ready for discharging. The policy is to empty the ones which have been waiting longest for the pump-discharging facility. Another restriction in discharging is caused by the levels in the hydrate agitators. If the levels are too high at the end of the draw-off of a precipitator, the next draw-off is delayed until the levels have suitably decreased. Therefore, for a certain period of time, there may be only two draw-offs. The levels of the hydrate agitators are controlled by varying the flows during the draw-off, by varying the agitator underflow, or by delaying the next draw-off at the end of a draw-off. It is noted that a draw-off cannot be stopped completely when it is running. The so-called "circulation time" of the contents in a precipitator is a direct result

FIGURE 5

SIMPLIFIED DIAGRAM OF FLOWS TO AND FROM THE PRECIPITATION CIRCUIT



of the operations and is computed from the beginning of the seeding to the beginning of the draw-off.

For purposes of cleaning and maintenance, six or fewer of any of the precipitators must remain empty at any given time. Hence, excluding the maximum number of precipitators which are either being filled or drawn-off, or are empty, only about 48 precipitators are in production generating alumina hydrate crystals at any time.

Section B: Assumptions in the Modeling

Figure 5 shows a simplified diagram of the flows to and from the precipitation circuit. In the model, only one hydrate agitator twice as high as the ones in the actual plant is used. This is the first simplifying assumption. The total height of this hypothesized agitator is 125.4 ft. and the area is 531 sq.ft. The level in the tank is kept between 90 ft. and 110 ft. by checking the level at each five minutes and making the necessary adjustments. If the level is between 90 and 100 ft., no action is taken. If the level becomes too low (under 90 ft.) each draw-off is increased by 0.1 of their present value (maximum draw-off flow = 350 cfm.). If the level becomes too high (over 100 ft.) each draw-off flow is decreased by 0.1 of their present value (minimum = 200 cfm.). If the level exceeds 110 ft. at the end of the draw-off of a precipitator, the next draw-off is delayed until the hydrate agitator level is below 110 ft.

The underflow of the hydrate agitator is changed depending on the level of the spent liquor tank, which is situated after the classification circuit. As the scope this case study is confined to operations in the precipitation circuit, both the classification circuit and the spent liquor tank have been excluded in

the model. Therefore the underflow of the hydrate agitator is adjusted in another way. Consider the material balance in the precipitation circuit as shown in Figure 5. Clearly, if the accumulation term is zero, at any time, it is necessary that:

$$F3 = F1 + F2$$

F2 is a batch flow, since seeding is a batch operation, and it is hence not possible in the plant to ascertain in advance its value for any given time instance. If $F3 < F1 + F2$, the total volume used in the precipitation circuit will increase, or, stating this in another way, the total "free" volume will decrease. If $F3 > F1 + F2$, the reverse will happen and the total used volume will decrease, whilst the total "free" volume will increase.

In order to measure the total "free" volume in the plant, (ie. empty volume or unused capacity) a variable called the "total free height" is defined to be the sum on all the precipitators of the "free height" of each of the precipitators. Also, by definition used in the plant:

Free height of a precipitator=maximum height of a precipitator (62.7 ft.) - height always left free at the top (3.0 ft.) - present height of this precipitator (H),

$$= 59.7 - H \text{ (feet)}.$$

With N precipitators in the circuit:

$$\text{Total free height} = \sum_{i=1}^{N} (59.7 - H_i)$$

$$= N \times 59.7 - \sum_{i=1}^{N} H_i$$

This total free height in the precipitation circuit is an indication of the total volume that is available for filling. In the plant that is studied, there are always about five precipitators on the spent caustic for descaling. These five precipitators are not included in what is called "short-term scheduling" in the plant (filling, seeding, circulating, drawing-off and waiting empty); so in the second assumption of this case study, one precipitator is removed from each of the five groups. Hence, there will be 54 precipitators in groups of 8, 13, 9, 12 and 12 precipitators. To facilitate the operation of the precipitators, it is necessary to have three or four precipitators empty all the time. With the precipitators filling, seeding and drawing-off, the feasible range of values for the total free height is between 250 and 300 ft. If the total free height becomes too low or too high, the circulating hours will either increase or decrease too much, respectively. With insufficient free volume, it is also expected that the scheduling of the precipitators will become difficult. In order to regulate the total free height, it is checked at every two hours in the simulation model and adjustments are made. If it is too high (>350 ft.), the underflow of the hydrate agitator is decreased, and if it is too low (<200 ft.), the same flow is increased. This is <u>not</u> done in the plant, but since the present model does not include the classification circuit, the underflow of the hydrate agitator cannot be controlled in the same manner as in actual plant operations. This is the third important assumption in modeling.

The fourth assumption is that the flowrates of the coarse and the fine seed streams used in the batch-seeding of the precipitators are constant. This approximates actual flow conditions in the plant. It is noted that although the average seed flowrate into the circuit over a long time period must vary considerably, since the flowrate must be zero when no precipitators are being seeded, the flowrate is fairly constant during the seed charging.

The fifth assumption concerns the filling of the precipitators using the policy of "longest-empty-soonest-filled". A similar policy ("longest-in-circulation-soonest-discharged") which is used with the "drawing-off" operations, is apparently needed so as to ensure that the precipitator contents have had sufficient time allowed for the crystallization reaction. However, with the filling operations, the "longest-empty-soonest-filled" policy is not important; it is inconsequential if any one precipitator is preferentially kept empty for a longer period of time than others in the same group. This policy is not strictly observed in the plant. It is, nevertheless, a logical way to fill the precipitators and is adopted in the model.

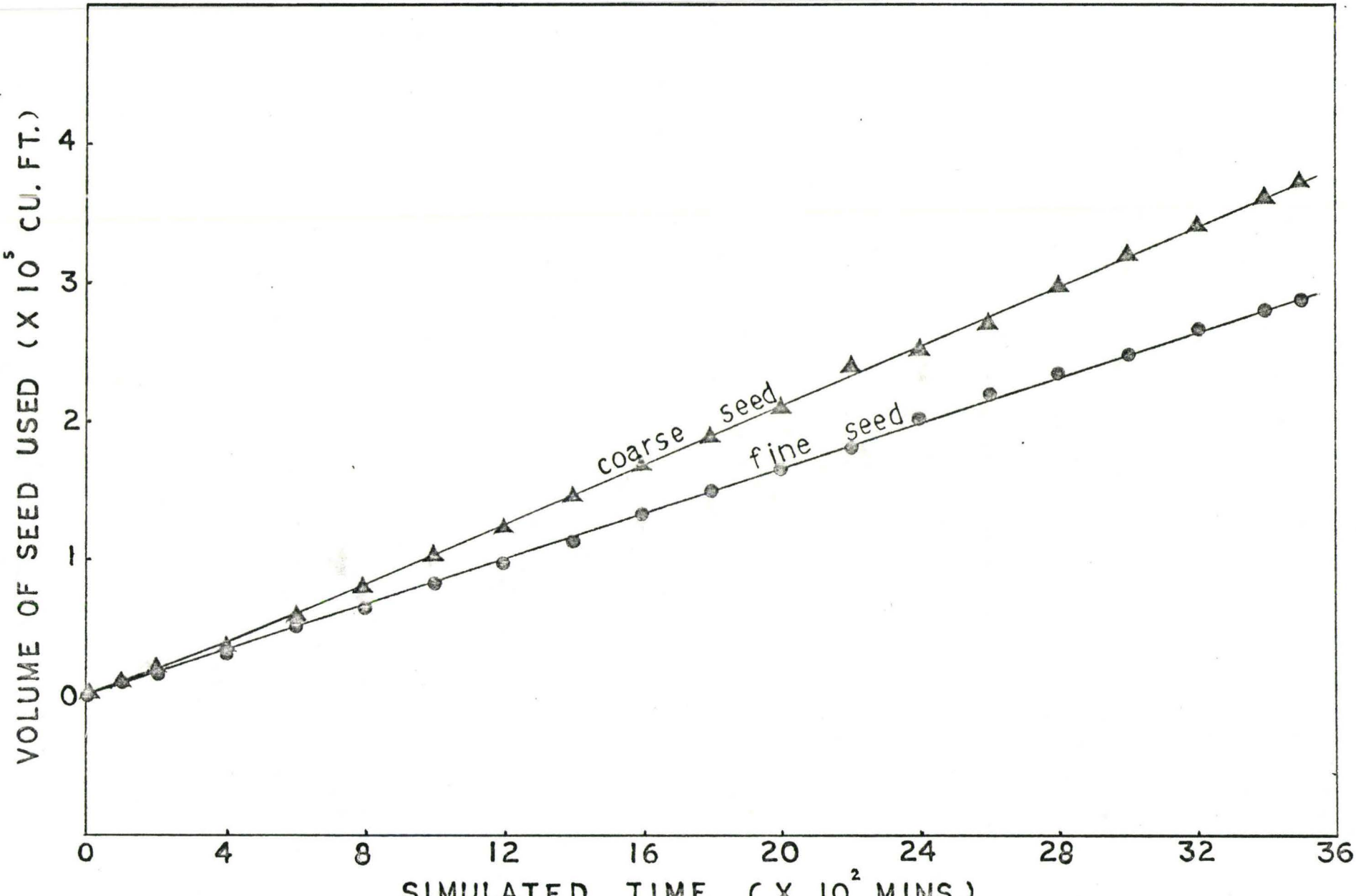
Section C: Discussion of the Simulation Results

(i) Initial System State

An initial system state is defined at the start of the simulation using the data set. The simulation run is scheduled to cover the period between an initial reference time point at 0.0 minutes and a final time point at 3500.0 minutes. Two initial event entries were made in the event-file; the first is to call a subroutine (VALVØ2) to initialize certain non-system variables (ie. variables that do not belong to the DESS esecutive); and the second is to call the subroutine GEMCS which transfers control of the simulation to the GEMCS

subsystem. 54 precipitators are read into three other files maintained in the array NSET. These files are numbered 2, 3 and 4. (File No. 1 must always be the event-file.) Precipitators in file 2 are those that are empty and are waiting for filling with pregnant liquor. Those in file 3 have been filled with liquor, either up to 46.7 ft. if they are "regular" or up to 49.7 ft. if they are "specials"; these precipitators are waiting to be charged with coarse or with fine seed material, respectively. File 4 contains precipitators that are "circulating", ie. those that are in the process of being charged with seed, and, those that are completely filled with liquor and seed and are left aside to enable the crystallization reaction to proceed. Of these 54 precipitators, 8 are in group 1 of the five physical groups in the plant; 13 in group 2, 9 in group 3, 12 in group 4 and 12 in group 5. It has been assumed that five precipitators (one from each group) are being descaled, and are therefore not included here since they are not available for short-term scheduling. The distribution of the precipitators from each group into the three files is arbitrary and artificially randomized. Similarly, the division of the precipitators in files 3 and 4 into the "regular" or "special" types is on a random and arbitrary basis. Since it is necessary that some precipitators to be discharged should have been in circulation for longer than 24 hours, the times at which the precipitators in file 4 began circulation are put backwards from the initial reference time point of 0.0 minutes. Again, the separation of these times at equal 38 minute intervals is arbitrary, but this gives the first precipitator to be discharged a circulation time of well over 24 hours (which is regarded by plant personnel as near optimal).

FIG. 6: SEED VOLUME VS. TIME



(ii) System Performance

(a) Coarse Seed Volume; Fine Seed Volume; Number of Circulating Precipitators; Circulation Times

Figure 6 shows the total cumulative volume of coarse and fine seed materials that have been used in the circuit at different time-points during the simulation. These values are required by the model in order to perform material balance calculations; in effect, they are a record of the batch flows of the seed streams which cannot be determined in advance. It is observed in the figure that the line for the coarse seed volume increases more steeply than that for the fine seed valume, even though the slopes of both lines rise in the same steady manner. The steeper gradient of the former is explained by the larger volume of coarse seed used to fill one precipitator. The smooth, steady slopes indicate that on the average, there is about the same number of "special" precipitators being filled as there are "regulars". This is consistent with the 1/1 filling policy that was tested in the simulation run. Since one "regular" filling must be followed by one "special" filling and vice versa, as required by the policy, there can be no preferential filling of either type. In the initial data set, there are 26 "regulars" (called type 1 in the model) and 22 "specials" (type 2). This does not include those precipitators in file 3 which only contain pregnant liquor but no seed, those empty ones in file 2, and those that are either filling or drawing-off. After 200 minutes (3 hours, 20 minutes), the number of "regulars" and "specials" among the circulating precipitators are 25 and 20, respectively. At the end of the run, ie. after 3500 minutes (2 days, 10 hours and 20 minutes), the number of "regulars" and "specials" are 23 and 24. It can be seen that even though the number of "specials" and "regulars" are

FIGURE 7a NUMBER OF CIRCULATING PRECIPITATORS VS. SIMULATED TIME

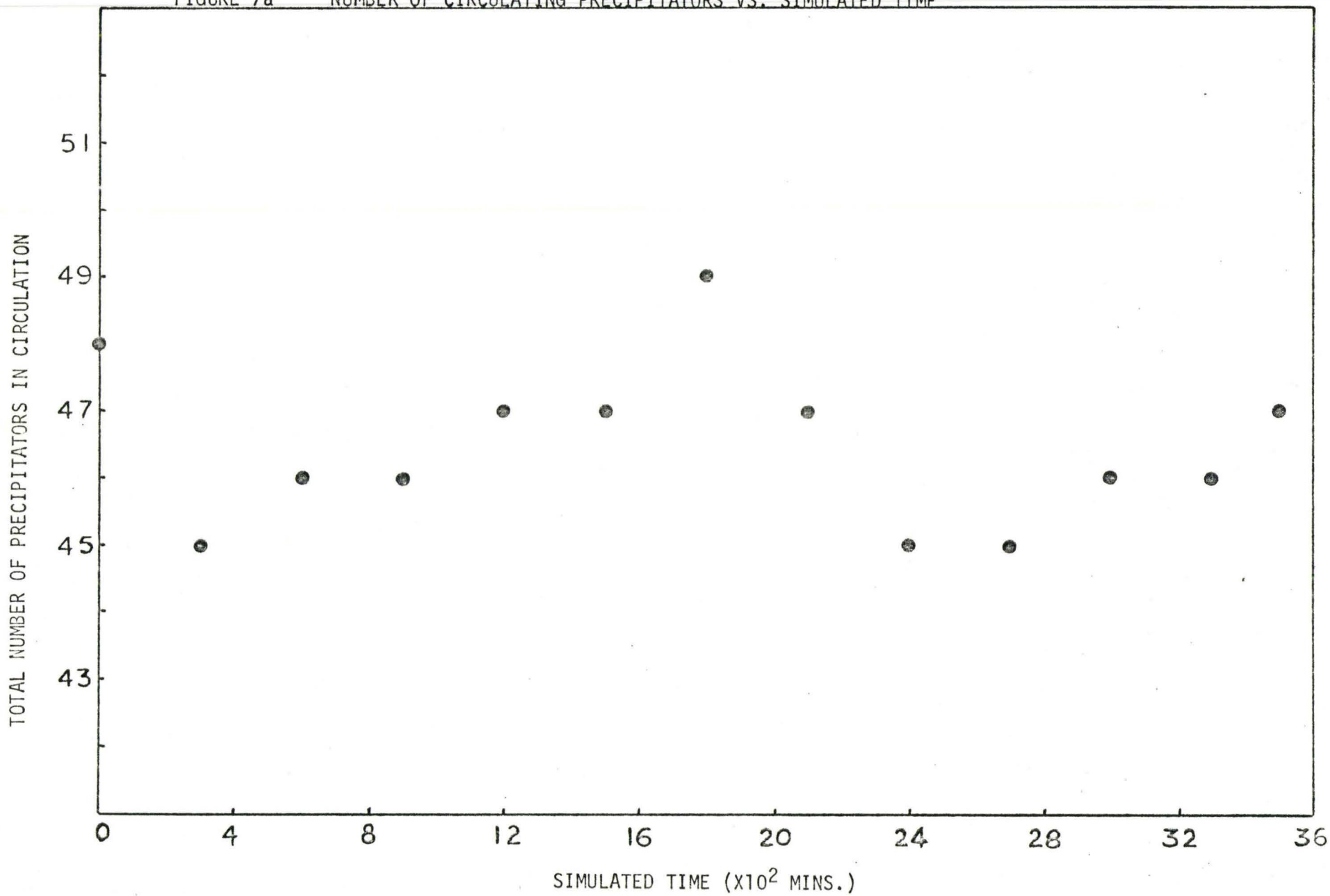
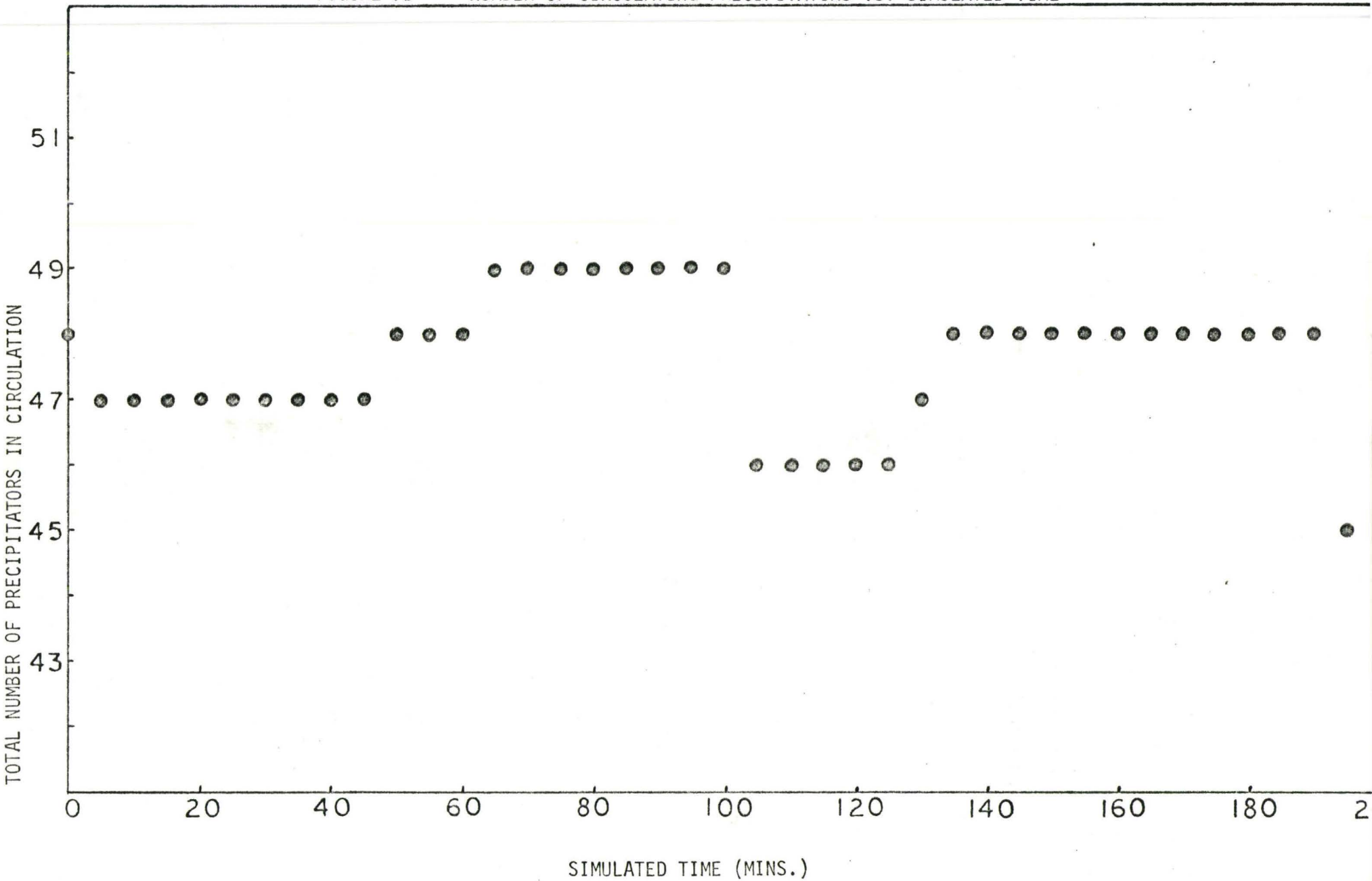


FIGURE 7b NUMBER OF CIRCULATING PRECIPITATORS VS. SIMULATED TIME



nearly in the 1:1 ratio, which could be expected with a consistent adherence to the 1/1 policy, the total number of circulating precipitators, ie. the sum of "specials" and "regulars", is not the same at different time points. Hence, the following assertions are made:

(1) the ratio of "regular" and "special" precipitators in circulation at any time may be predicted by a consistent filling policy;

(2) the actual number of "regular" precipitators and "special" precipitators in circulation follow a uniformly random distribution with an upper and a lower bound which could be estimated;

(3) the sum of "specials" and "regulars" in circulation is not a constant number but is shown to be variable from the data obtained, (Figure 7a);

(4) any average "steady state" model must account for precipitators that are filling or discharging; these precipitators carry contents that do not significantly change in crystal product concentration, and therefore are not "specials" or "regulars" in the same sense as those in circulation. However, the volume of their contents is not negligible. Consequently, these "non-reactive" precipitators must either be counted separately or be included in some fashion as equivalent "specials" or "regulars". It is thus apparent that, in order to solve the equations governing the average operating conditions in the circuit (First Case Study), variables other than the number of "specials" and "regulars" may be more easily pre-determined. This is further discussed in relation to the data obtained on the total number of precipitators in circulation and the circulation times resulting directly from the scheduling operations.

FIGURE 8 CIRCULATION TIME VS. TIME AT START OF DRAW-OFF

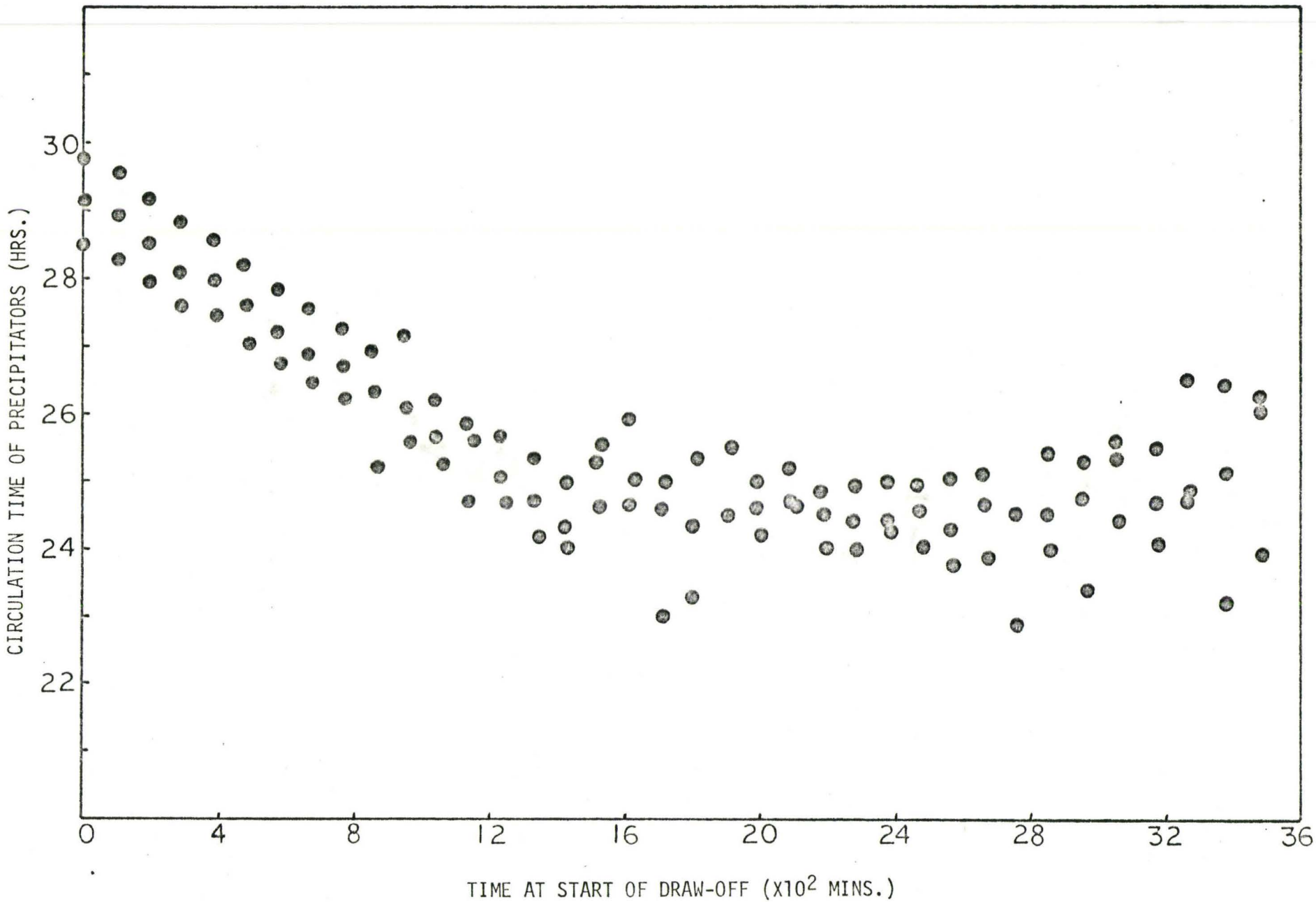


Figure 7a shows the number of circulating precipitators in the circuit at different time points during the simulation run. A magnification of the variations between time 0.0 minutes and 200.0 minutes is shown in Figure 7b. From the two graphs, a pattern of the changes of this discrete variable is discernable. Apparently, it does not change in a purely random manner and a repeated trend of the variations can be observed for the periods between 200 and 1200 minutes and between 2400 and 3500 minutes. Although this evidence is not conclusive, it points to the possibility of using time-series analysis or other forecasting techniques to formulate statistical models from plant data for predictions of this variable. Even the scarce simulation data available shows that the number of precipitators in circulation does have a predictable mean value and that the range of variations is not large. Since precipitators filling or drawing-off each has a total volume that vary between zero and the maximum capacity, and since there are usually six precipitations undergoing these operations, it is asserted that the average of their combined filled volume is equivalent to the volume of three precipitators in circulation. Hence, the sum of all precipitators that are effectively in operation can be estimated using the predicted number of circulating precipitators and adding three to it.

Figure 8 shows the circulation time of precipitators on the vertical axis and the time at which the precipitator with the corresponding circulation time is discharged on the horizontal axis. Between time 0.0 and time 1000.0 minutes a constant decrease of the circulation time is observed. The unusually large values at the beginning is due to the original ordering the precipitator times in file 4 in the data set. Hence, these values are artificial since they are not a true result of the scheduling operations programmed in the model. The

FIGURE 9a TOTAL FREE HEIGHT (1) AND HYDRATE AGITATOR LEVEL (2) VS. SIMULATED TIME

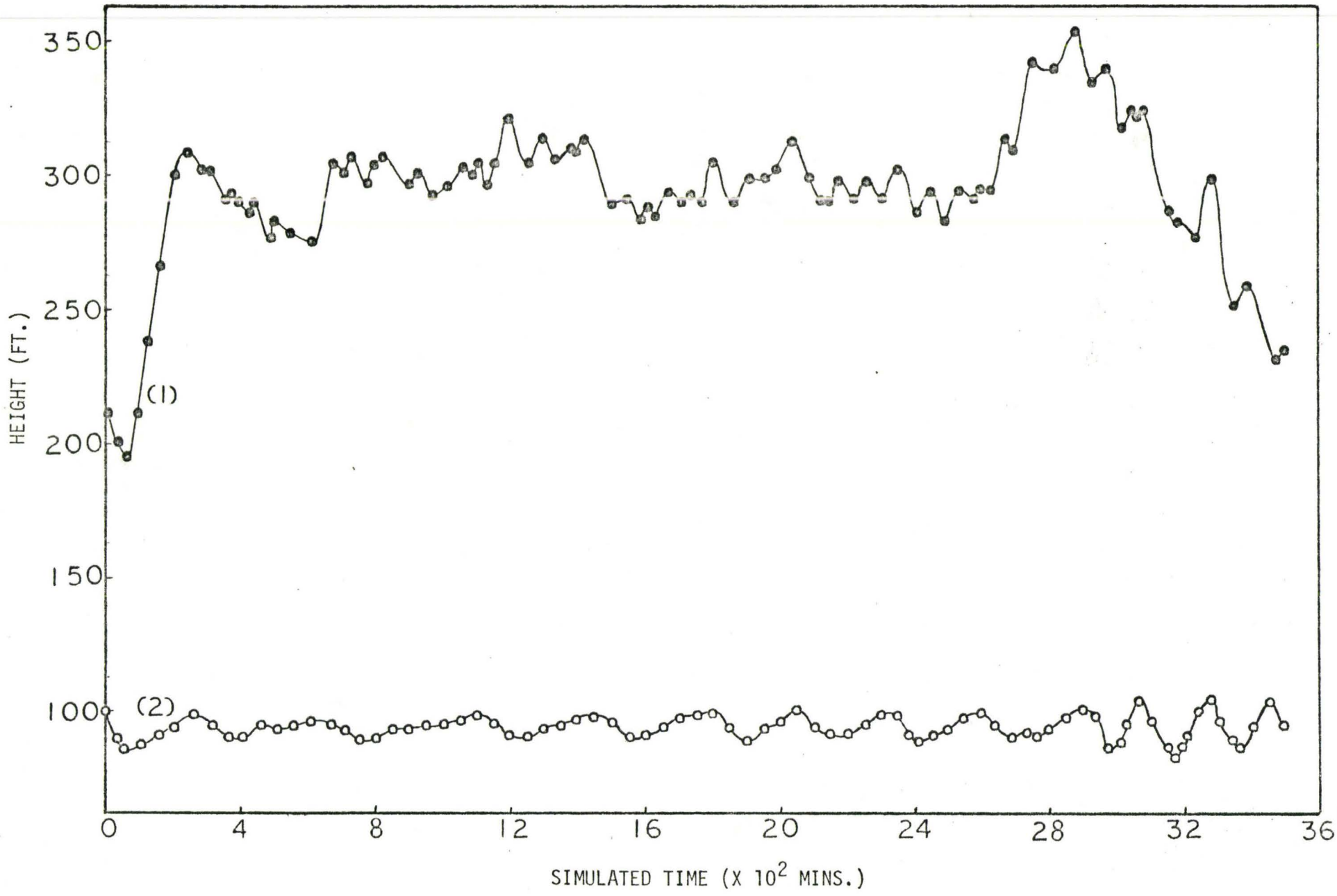
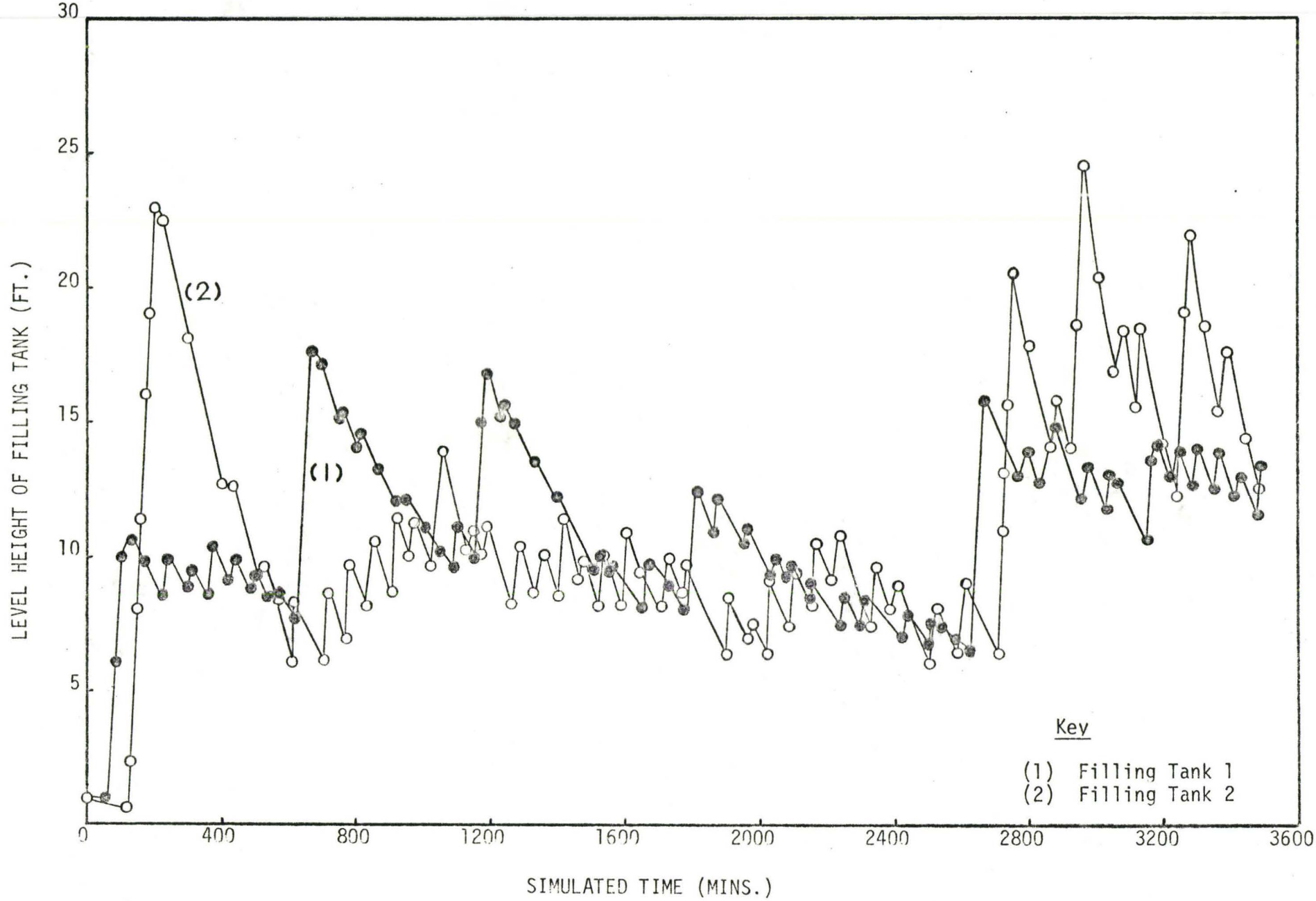


FIGURE 9b LEVEL HEIGHT OF FILLING TANKS VS. TIME



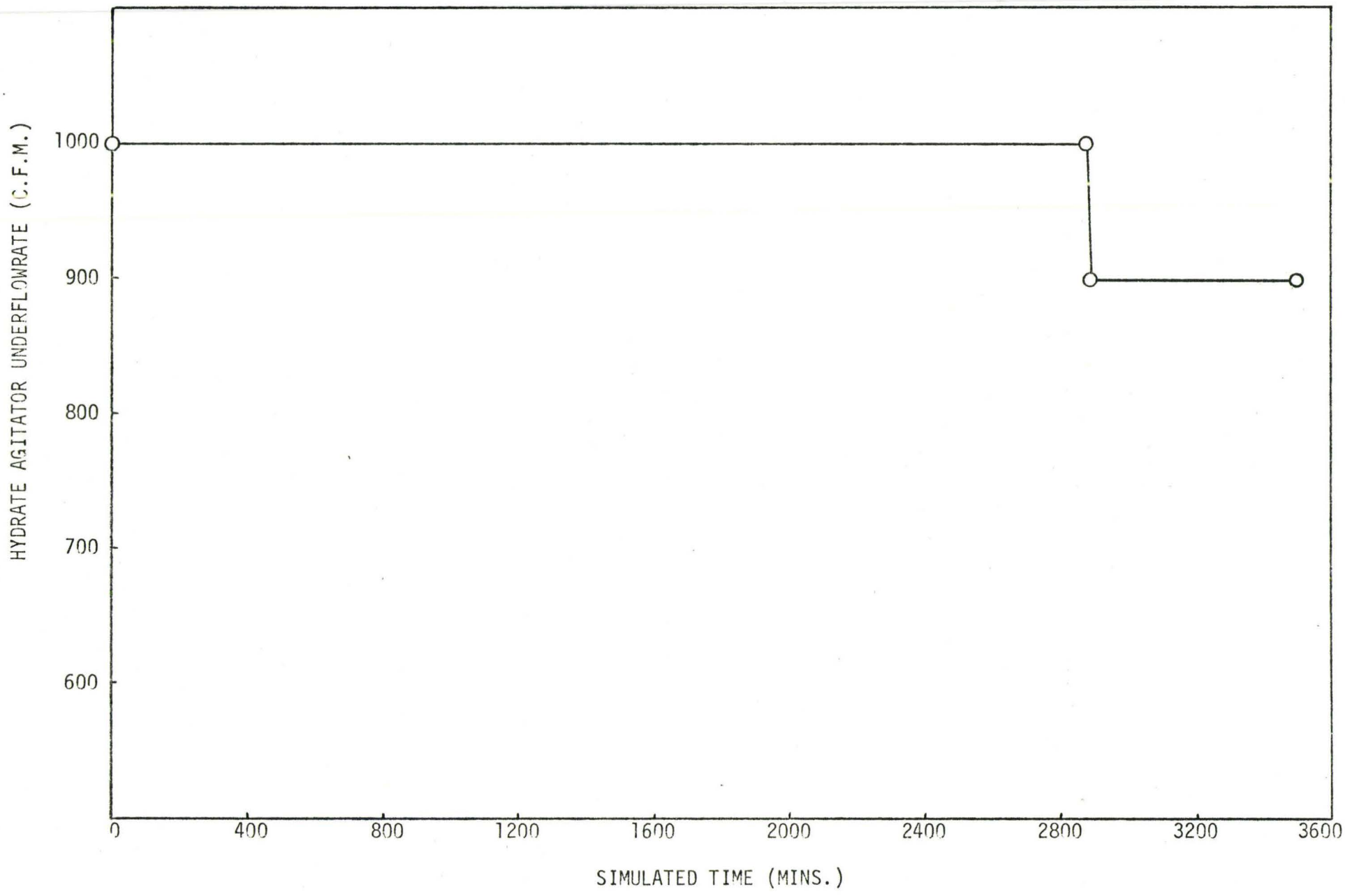
steadily declining slope, however, is evidence that the operations in the circuit are exerting a constraining effect on the length of time each precipitator is allowed for circulation. After time 1000.0 minutes, the slope levels off at around a circulation time value of 24.5 hours but the points begin to show a larger degree of scattering. There is no discernable trend or pattern and the points appear to be randomly distributed about some mean value. The conclusion from this is that the model predicts that the operations in the circuit will limit the circulation times of the precipitators to lie within a small range around the value of 24 hours. This is verified by communication from plant personnel indicating that the expected average circulation time in the plant is 24 hours. Therefore, a reliable estimate of the mean circulation time and its variation can be obtained from plant operating data.

With the information on the total seed volumes used up to a time, mean flowrates of continuous seed streams equivalent to the actual batch flows can be computed. Estimates of the effective number of operating precipitators and the mean circulation times can be made from plant data. The maximum volume of a full precipitator is a known constant and the flowrate of the continuous pregnant liquor stream is measurable. Hence, a method for solving the average "steady-state" conditions in the circuit, similar to that employed in the first case study, could be formulated.

(b) Total Free Height; Level Height of Hydrate Agitator; Level Height of Filling Tanks

Figures 9a and 9b show the changes in the total free height and the level height of the hydrate agitator and filling tanks during the simulation. The low values of the total free height at the start of the simulation are caused by the

FIGURE 10 HYDRATE AGITATOR UNDERFLOW VS. TIME



arbitrary ordering of the precipitators in the initial data set. The values increase steadily with time in response to the operations in the circuit. After time 200.0 minutes, the consistent upward gradient disappears and the graph levels off as the system begins to show the effects of the scheduling operations. At time 2880.0 minutes, the total free height reaches a maximum value of 354.4 ft. This is an indication that the available free volume in the circuit is near its allowable maximum. In order to bring the system back to normal operating conditions, the underflow of the hydrate agitator is decreased by a certain amount at this point. (See Figure 10. This aspect of control is artificial; in the real plant, factors downstream away from the precipitation circuit, such as the level of the spent liquor tank, must also be considered.) The system responds almost instantly with a reduction in the total free height. The damping effect is, however, noticeable as the line oscillatesto a new mean value. At the end of the simulation, the total free height is averaging around 230 ft. Wider oscillations are also observed with the level in the hydrate agitator. This is expected because its underflow has undergone a step-change and its input flow consisting of the precipitator draw-offs are always adjusted to maintain its level within a specified range. From this, it is seen that: (1) the extent of the fluctuations in the total free height and the hydrate agitator level caused by a change in the hydrate agitator underflow depends on the magnitude of the change; and (2) the new average total free height is related to the new underflowrate. Since the total free height should be constrained to lie between 200 and 350 ft., the step-change in the hydrate agitator underflow could have been made smaller to maintain a final average value of the total free height nearer to 275 ft.

The level of the two filling tanks average around 10 ft. after an initial rapid rise from the 1 foot level originally assigned in the data set. A plant objective is to minimize the two levels close to zero. Apparently, this requires more fillings of precipitators, ie. having a shorter idling time of the liquor inlets in between fillings. With more fillings, there must also be more empty precipitators available; hence, precipitators should be drawn-off sooner and the hydrate agitator underflow increased. This, however, will result in a larger total free height and a smaller value for the average circulation time. With less time allowed for the crystallization reaction, production would decrease. This reduction in product quantity is, however, in conjunction with a reduction in inventory costs tied to lower levels in the filling tanks. Hence, this area can be identified for an optimization study carried out to improve production benefits.

Section D: Model Validity

The simulation model is an abstraction of the real system, hence, its validity should be considered within the framework of the inherent simplifying assumptions. Of the five assumptions contained in the model, the third which involves the adjustment of the hydrate agitator underflow depending on the total free height in the circuit, is believed to be the one most seriously affecting the "correctness" of the simulation results. From discussions with plant personnel, it is apparent that this represents an important deviation from reality, thus making it difficult to estimate the bearing it has on the validity of the model and the results obtained. However, in reference to the previous discussion on system performance, it is noted that the assumption was actually used only once during the simulation run from time 0.0 to 3500.0 minutes. The hydrate

agitator underflow was decreased in a step-change when the total free height exceeded 350 ft. In spite of the questioned validity of this operation, some usefulness is shown by the information obtained on how the system would respond under that circumstance.

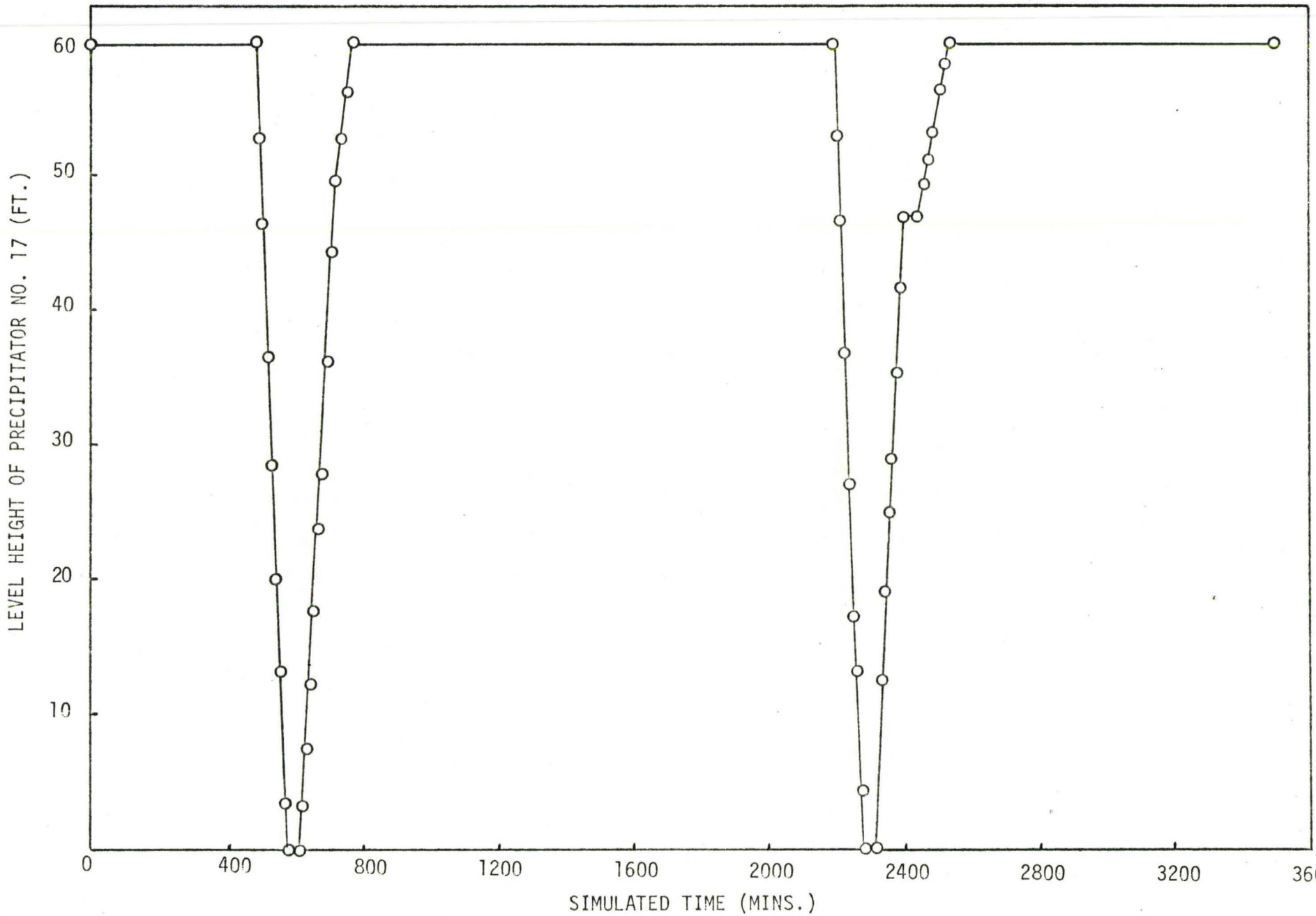
It is not possible to make a quantitative check on the model validity unless actual plant data are available for a comparison with the simulation data obtained from the model under similar operating conditions. A check on the material balance in the system should, however, reveal any programming error in the model. If the material balance is satisfactory, it proves that no significant quantitative error exists in the model, but it still does not logically follow that the quantitative results from the simulation are exact predictions of plant performance. In this study, the material balance in the system is checked at every time instance, and the residual term in the equation:

Input - Output - Accumulation = Residual

is computed in each instance. The largest value of 3 cu. ft. obtained at time 3500.0 minutes is less than 1.0×10^{-4} percent of the magnitude of the other three corresponding terms. The residual value is therefore negligible and the material balance satisfactory.

A criterion for verifying the validity of the model on a qualitative basis is that the circulation time of the precipitators should converge to 24 - 26 hours if the model truly reflects the actual plant conditions arising from the scheduling operations. The results obtained show that the data on the circulation time do lie within the expected range. Also, the total free height calculated during the simulation does not vary outside the range that is prescribed for the real plant. Data collected on the level height of one arbitrarily

FIGURE 11 LEVEL HEIGHT OF PRECIPITATOR NO. 17 VS. TIME



selected precipitator (No. 17) show in Figure 11 the same pattern of changes as that observed for precipitators in the circuit.

Although these observations and others which might be made show that the model is not invalid, it is not possible to provide conclusive evidence proving absolute validity. This is, however, not the objective of simulation, which is to yield more information on the problem rather than to provide its exact or specific solution. To this same end, recommendations for a further study of the scheduling operations and their effects in the precipitation circuit of the Bayer Process Plant are discussed next.

Section E: Recommendations of Strategies for Model Application and Improvement

The established simulation model can be used for perturbation studies to answer "what-if?" kind of questions. Examples of such studies include:

(1) Changing the number of precipitators in the circuit;

(2) Changing the average input flowrate of pregnant liquor;

(3) Changing the average underflowrate of the hydrate agitator;

(4) Implementing different filling policies over varying time periods;

(5) Changing the rates at which seed charges could be made at present.

and each time observing the effects of the changes on conditions in the circuit, such as, the resulting precipitator circulation times, the total free height, and the level heights of the filling tanks and the hydrate agitator. Since it is already recognised that beneficial effects (eg. increased circulation times) can be concurrent with undesirable effects (eg. increased filling tank levels), such studies of system disturbance should yeild a pattern of the feasible operating region for improved performance.

Modification of the model in various ways can be made to study other aspects of the operations in the circuit. The performance resulting from a plant re-design, such as the addition or removal of facilities for precipitator filling or discharging, can be simulated using a modified version of the same model. Since the physical constraints in the circuit limit operation conditions to lie within a definite range, if it is required to operate outside this range, the constraints must necessarily be changed. The importance of the circulation time as a measure of better plant performance has been mentioned. The reason for this importance is that product quantity and quality (size-distribution) is directly related to the circulation time, and also to the quantity and quality of the seed charge added. Thus, in order to obtain a more useful model, it is necessary to add modifications so as to account for the extra information on product quantity and quality. The way this could be done is to carry this information on seed quality in the system vectors describing the stream flows in the plant, and to pass on this information together with the information on the circulation time provided by the existing model to another model, programmed as a modular subroutine, that can predict the quantity and quanlity of the next batch of products. Mechanistic models in this category are being developed at McMaster University (Groeneweg[10]), and elsewhere (Misra and White[11]).

The present model requires the underflowrate of the hydrate agitator to be adjusted when the total free height is outside a prescribed range. In the plant, this adjustment is made according to the level changes in the spent liquor tank which is not in the precipitation circuit, but is further downstream. Therefore, to eliminate the artificial control of the hydrate agitator underflow programmed in the model, the definition of the scope of the system should be

broadened to include the downstream sections of the plant. This would, of course, entail an extension of the present simulation model. Such an extension would also allow a study of the effects of the changing quality of the seed material which is recycled from downstream to be used as seed charges for the precipitators. The DESS executive is fully capable of providing the information-handling framework needed in such simulation experiments.

CONCLUSION

A presentation of the new executive system, DESS, for discrete-event simulation, and two application case studies involving the precipitation circuit in the Bayer Process Plant has been made in this thesis. DESS was generated in answer to the need for an information-handling executive that anticipates the programming tasks involved in simulation studies of large, complex systems. Other similar executives, such as, GPSS, GASP, and SIMSCRIPT, have been considered and were rejected on such grounds as a lacking in flexibility and insufficiency. The ability of DESS to deal with complex systems involving large networks of information flow in an efficient manner makes it a realistic tool for simulation modeling and experimentation. (Some measure of this efficiency is realised in the relatively low core memory and computation time required per simulation run. In the second case study, the simulation required less than 37k words or about 2 million bits, and 42 seconds per run in the CDC 6400 computer.)

The application case studies have contributed more information on the scheduling operations and their effects on the flow conditions in the precipitation circuit of the Bayer Process Plant, and on the difficulties in representing the average conditions in the plant with a "steady-state" model. The results obtained do not purport to reflect exact plant behaviour but have served usefully in the identification of the relationships between the control operations and the response in different parts of the circuit. This knowledge should lead to more effective control of plant performance and more accurate predictions of changes and problem areas.

REFERENCES

(1) Bayer, K. J., Process of Making Alumina, U.S. Patent 515, 895, March 6, 1894.

(2) Study Reports on "The Bayer Process for Alumina Extraction", Chem. Engin. Dept., McMaster University, Canada, 1968-69.

(3) Johnson, A. I., and Associates, GEMCS Manual and Application Studies, Chem. Engin. Dept., McMaster University, July, 1970.

(4) Kiviat, P. J., Development of Discrete Digital Simulation Languages, Simulation, Vol. 8, No. 2, pp. 65 - 70, February, 1967.

(5) Kiviat, P. J., R. Villanueva, and H. M. Markowitz, The SIMSCRIPT II Programming Lanuage, Prentice-Hall, Inc., 1968.

(6) Markowitz, H. M., B. Hausner, and H. W. Karr, SIMSCRIPT - A Simulation Programming Language, The RAND Corp., Prentice-Hall, Inc., 1964.

(7) Peters, F. A., P. W. Johnson, and R. C. Kirby, A Cost Estimate of the Bayer Process For Producing Alumina, U.S. Dept. of the Interior, Bureau of Mines, Washington, 1966.

(8) Pritsker, A. A. B., and P. J. Kiviat, Simulation with GASP II, A FORTRAN based Simulation Language, Prentice-Hall, Inc., 1969.

(9) Wyman, F. P., Simulation Modeling: A Guide to Using SIMSCRIPT, J. Wiley and Sons, Inc., 1970.

(10) Groeneweg, P., Personal Communication, Chem. Engin. Dept., McMaster University, Canada, 1971.

(11) Misra, C., and E. T. White, A Mathematical Model of the Bayer Precipitation Process for Alumina Production, Paper submitted to CHEMECA 1970 Conference at Sydney and Melbourne, August, 1970.

APPENDIX 1

LISTINGS AND DESCRIPTIONS OF THE DESS PROGRAM

Subroutine GASP (NSET)

The subroutine is called by the user-supplied "main" program to start the simulation. It then selects events, sequences time, controls the monitoring of intermediate simulation results and initiates the printout of the final summary output when the simulation has been completed. It controls the simulation from start to end, and at the end, it transfers control back to the main program.

```
      SUBROUTINE GASP (NSET)
C
C     DESS/CDC6400
C
C     DISCRETE-EVENT SIMULATION SYSTEM
C
C                  CREATED BY --- RAY H. T. WONG
C                             --- MCMASTER UNIVERSITY
C                             --- HAMILTON, ONTARIO, CANADA
C
C                          IN --- 1971/1972
C
C                    BASED ON --- GASP II AND GEMCS/6400
C
C*****THE LENGTH OF THE ARRAYS AND VECTORS SHOULD BE AS FOLLOWS -
C
C*****ATRIB(IM),ENQ(NOQ),INN(NOQ),JCELS(NHIST,MXC),KRANK(NOQ),
C*****MAXNQ(NOQ),MFE(NOQ),MLC(NOQ),MLE(NOQ),NCELS(NHIST),NQ(NOQ),
C*****NSET(MXX,1),PARAM(NPRMS,NEPRM),QTIME(NOQ),SSUMA(NSTAT,NHIST),
C*****SUMA(NCLCT,NHIST),VNQ(NOQ)
C*****NOTE THAT NSET HAS TO BE DIMENSIONED TO NSET(MXX,ID) IN
C*****MAIN PROGRAMME (OR ERRORS IN CORE STORAGE WILL ARISE)
C
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      NOT=0
1     CALL DATAN (NSET)
C
C*****PRINT OUT FILING ARRAY
C
      JEVNT=101
      CALL MONTR (NSET)
      PRINT 17
C
C*****OBTAIN NEXT EVENT WHICH IS FIRST ENTRY IN FILE 1. ATRIB(1) IS
C*****EVENT TIME, ATRIB(2) IS EVENT CODE
C
2     CALL RMOVE (MFE(1),1,NSET)
      TNOW=ATRIB(1)
      JEVNT=ATRIB(2)
C
C*****TEST TO SEE IF THIS EVENT IS A MONITOR EVENT
C
      IF (JEVNT-100) 3,11,10
```

```
3       I=JEVNT
C
C*****CALL PROGRAMMERS EVENT ROUTINES
C
        CALL EVNTS (I,NSET)
C
C*****TEST METHOD FOR STOPPING
C
        IF (MSTOP) 4,14,5
4       MSTOP=0
C
C*****TEST FOR NO SUMMARY REPORT
C
        IF (NORPT) 9,6,7
5       IF (TNOW-TFIN) 14,6,6
6       CALL SUMRY (NSET)
        CALL OTPUT (NSET)
C
C*****TEST NUMBER OF RUNS REMAINING
C
7       IF (NRUNS-1) 9,16,8
8       NRUNS=NRUNS-1
        NRUN=NRUN+1
        GO TO 1
9       CALL ERROR (93,NSET)
10      CALL MONTR (NSET)
        GO TO 2
C
C*****RESET JMNIT
C
11      IF (JMNIT) 9,12,13
12      JMNIT=1
        GO TO 2
13      JMNIT=0
        GO TO 2
C
C*****TEST TO SEE IF EVENT INFORMATION IS TO BE PRINTED
C
14      IF (JMNIT) 9,2,15
15      ATRIB(2)=JEVNT
        JEVNT=100
        CALL MONTR (NSET)
        GO TO 2
C
C*****IF ALL RUNS ARE COMPLETED RETURN TO MAIN PROGRAM FOR INSTRUCTIONS
C
16      RETURN
C
C
C
17      FORMAT (1H1,38X,24H**INTERMEDIATE RESULTS**//)
        END
```

Subroutine DATAN (NSET)

The subroutine performs data input and initialization. System variables pertaining to the GASP subsystem are assigned starting values and entries describing the initial schedule of future events are made in the event-file (file no. 1). To allow sequential runs, the subroutine provides the option of reading in new values for the system variables and/or new initial simulation entries to alter the beginning situation. All data are read in as real numbers in a format of 5F12.0.

```
      SUBROUTINE DATAN (NSET)
      DIMENSION NSET(12,1)
C*****RVAR IS A TEMPORARY REAL VARIABLE STORAGE VECTOR - MAY NEED TO BE
C*****INCREASED IF OTHER VECTORS AND ARRAYS ARE INCREASED
      DIMENSION RVAR(20)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      IF (NOT) 2,3,1
C
C*****NEP IS A CONTROL VARIABLE FOR DETERMINING THE STARTING CARD
C*****TYPE FOR MULTIPLE RUN PROBLEMS.  THE VALUE OF NEP SPECIFIES THE
C*****STARTING CARD TYPE.
C
1     NT=NEP
      GO TO (3,5,6,8,10,12,14,22,26,38), NT
2     CALL ERROR (95,NSET)
3     NOT=1
      NRUN=1
C
C*****DATA CARD TYPE ONE
C
      READ 39, NAME
      READ (5,40) (RVAR(I),I=1,5)
      NPROJ=INT(RVAR(1))
      MON=INT(RVAR(2))
      NDAY=INT(RVAR(3))
      NYR=INT(RVAR(4))
      NRUNS=INT(RVAR(5))
      IF (NRUNS) 4,4,5
4     CALL EXIT
C
C*****DATA CARD TYPE TWO
C
5     READ (5,40) (RVAR(I),I=1,10)
      NPRMS=INT(RVAR(1))
      NHIST=INT(RVAR(2))
      NCLCT=INT(RVAR(3))
      NSTAT=INT(RVAR(4))
      ID=INT(RVAR(5))
      IM=INT(RVAR(6))
      NOQ=INT(RVAR(7))
      MXC=INT(RVAR(8))
C
C*****NEPRM IS A NEW VARIABLE INTRODUCED AS NUMBER OF ELEMENTS IN
C*****EACH PARAMETER SET
C
      NEPRM=INT(RVAR(9))
      SCALE=RVAR(10)
      IF (NHIST) 8,8,6
```

```
C
C*****DATA CARD TYPE THREE IS USED ONLY IF NHIST IS GREATER THAN ZERO
C*****SPECIFY NUMBER OF CELLS IN HISTOGRAMS NOT INCLUDING END CELLS
C
6     READ (5,40) (RVAR(I),I=1,NHIST)
      DO 7 I=1,NHIST
      NCELS(I)=INT(RVAR(I))
7     CONTINUE
C
C*****DATA CARD TYPE FOUR
C*****SPECIFY KRANK=RANKING ROW
C
8     READ (5,40) (RVAR(I),I=1,NOQ)
      DO 9 I=1,NOQ
      KRANK(I)=INT(RVAR(I))
9     CONTINUE
C
C*****DATA CARD TYPE FIVE
C*****SPECIFY INN=1 FOR LVF, INN=2 FOR HVF
C
10    READ (5,40) (RVAR(I),I=1,NOQ)
      DO 11 I=1,NOQ
      INN(I)=INT(RVAR(I))
11    CONTINUE
C
C*****PRINT OUT PROGRAM IDENTIFICATION INFORMATION
C
      WRITE (6,41) NAME,NPROJ,MON,NDAY,NYR,NRUN
      WRITE (6,44) NRUNS,NPRMS,NHIST,NCLCT,NSTAT,ID,IM,NOQ,MXC,NEPRM
      WRITE (6,43) SCALE,(NCELS(I),I=1,NHIST)
      WRITE (6,45) (KRANK(I),I=1,NOQ)
      WRITE (6,46) (INN(I),I=1,NOQ)
      IF (NPRMS) 2,14,12
12    DO 13 I=1,NPRMS
C
C*****DATA CARD TYPE SIX IS USED ONLY IF NPRMS IS GREATER THAN ZERO
C
      READ (5,40) (PARAM(I,J),J=1,NEPRM)
13    CONTINUE
C
C*****DATA CARD TYPE SEVEN.  THE NEP VALUE IS FOR THE NEXT RUN.  SET
C*****JSEED GREATER THAN ZERO TO SET TNOW EQUAL TO TBEG.
C
14    READ (5,40) (RVAR(I),I=1,7)
      MSTOP=INT(RVAR(1))
      JCLR=INT(RVAR(2))
      NORPT=INT(RVAR(3))
      NEP=INT(RVAR(4))
      TBEG=RVAR(5)
      TFIN=RVAR(6)
      JSEED=INT(RVAR(7))
      IF (JSEED) 15,18,15
15    ISEED=JSEED
      DO 16 K=1,30
      RNUM=DRAND(ISEED)
```

```
16    CONTINUE
      TNOW=TBEG
      DO 17 J=1,NOQ
      QTIME(J)=TNOW
17    CONTINUE
18    JMNIT=0
      MXX=IM+2
C
C*****PRINT PARAMETER VALUES AND SIMULATION RUN VARIABLES
C
      IF (NPRMS) 21,21,19
19    DO 20 I=1,NPRMS
      WRITE (6,42) I,(PARAM(I,J),J=1,NEPRM)
20    CONTINUE
21    CONTINUE
      WRITE (6,47) MSTOP,JCLR,NORPT,NEP,TBEG,TFIN,JSEED
C
C*****DATA CARD TYPE 8
C*****SPECIFY INPUTS FOR NEXT RUN
C*****READ IN AND PRINT OUT INITIAL EVENTS AND ENTITIES
C*****END INITIAL EVENTS AND ENTITIES WITH JQ EQUAL TO ZERO
```

```
C*****INITIALIZE NSET BY JQ EQUAL TO A NEGATIVE VALUE ON FIRST EVENT
C*****CARD
C
22    WRITE (6,48)
      DO 25 JS=1,ID
      READ (5,40) AJQ
      READ (5,40) (ATRIB(JK),JK=1,IM)
      JQ=INT(AJQ)
      WRITE (6,49) JQ,(ATRIB(JK),JK=1,IM)
      IF (JQ) 23,26,24
23    INIT=1
      CALL SET (1,NSET)
      GO TO 25
24    CALL FILEM (JQ,NSET)
25    CONTINUE
C
C*****JCLR BE POSITIVE FOR INITIALIZATION OF STATISTICAL STORAGE ARRAYS.
C
26    IF (JCLR) 38,38,27
27    IF (NCLCT) 2,31,28
28    DO 30 I=1,NCLCT
      DO 29 J=1,3
      SUMA(I,J)=0.
29    CONTINUE
      SUMA(I,4)=1.0E20
      SUMA(I,5)=-1.0E20
30    CONTINUE
31    IF (NSTAT) 2,35,32
32    DO 34 I=1,NSTAT
      SSUMA(I,1)=TNOW
      DO 33 J=2,3
      SSUMA(I,J)=0.
33    CONTINUE
      SSUMA(I,4)=1.0E20
      SSUMA(I,5)=-1.0E20
34    CONTINUE
35    IF (NHIST) 2,38,36
36    DO 37 K=1,NHIST
      DO 37 L=1,MXC
37    JCELS(K,L)=0
38    CONTINUE
      RETURN
C
C
C
39    FORMAT (20A4)
40    FORMAT (5F12.0)
41    FORMAT (1H0,20A4,/,*   SIMULATION PROJECT NO.*,I4,/10X,4HDATE,I3,1
     1H/,I3,1H/,I5,12X,10HRUN NUMBER,I5,/)
42    FORMAT (1H ,* PARAMETER SET NO. *,I5,* ELEMENTS ARE - *,5F12.4,/,(
     140X,5F12.4))
43    FORMAT (1H ,* SCALE = *,F10.4,/*  NCELS(I), I = 1,NHIST *,/20I5)
44    FORMAT (1H ,* NRUNS,NPRMS,NHIST,NCLCT,NSTAT,ID,IM,NOQ,MXC,NEPRM *,
     1/10I5)
45    FORMAT (1H ,* KRANK(I), I = 1,NOQ *,/20I5)
```

```
46    FORMAT (1H ,* INN(I), I = 1,NOQ *,/20I5)
47    FORMAT (1H ,* MSTOP,JCLR,NORPT,NEP,TBEG,TFIN,JSEED *,/4I5,2F12.4,
     15)
48    FORMAT (1H0,30X,* DESCRIPTION OF ENTRIES (EVENTS AND ENTITIES) *,
     110X,* FILE NUMBER *,50X,* ATTRIBUTES *)
49    FORMAT (1H ,15X,I5,20X,5F12.4,/,(40X,5F12.4))
      END
```

Subroutine SET

The subroutine attends to the information storage and retrieval operations performed in the dynamic filing array, NSET. It has three main functions: (1) initializes the filing array, NSET, (2) updates the pointer system that establishes the relationships between entries of files, and (3) maintains statistics on the number of entries in each file. Statistical values maintained for each file of NSET are: the current number of entries, NQ(JQ); the largest number of entries ever filed, MAXNQ(JQ); the time integrated number of entries, ENQ(JQ); the last time an entry was filed or removed, QTIME(JQ); and the square of the number in the file integrated over time, VNQ(JQ). ENQ(JQ) and VNQ(JQ) are used to obtain the average and standard deviation of the number of entries in File JQ. The relationships between the variables are as follows:

$$ENQ(JQ) = \sum_{i=0}^{j} NQ(JQ) * (t_i - QTIME(JQ))$$

$$VNQ(JQ) = \sum_{i=0}^{j} NQ(JQ) * NQ(JQ) * (t_i - QTIME(JQ))$$

where t_i is the time at which NQ changes, and Σ is taken over each i^{th} point when that occurs. Statistical estimates in the time interval 0.0 to t_j would be:

$$\text{The average number in file JQ} = \frac{ENQ(JQ)}{t_j}$$

Estimate of the standard deviation of number in file JQ = $\sqrt{\left[\frac{VNQ(JQ)}{t_j}\right] - XNQ^2}$

The pointer system maintained by SET in the filing array, NSET, consists of the use of the last two rows in the array (rows MX and MXX) to store the "successor" pointers and "predecessor" pointers. The "successor" pointer in row MX of an entry in NSET contains the column number of the next entry succeeding it in the same file. Similarly, the "predecessor" pointer in row MXX of the same entry, contains the column number of the previous entry preceding it in the file. If an entry is the first in a file, it would have no predecessor and the code number of 9999 would be entered by SET for its predecessor pointer. If it is the last entry in the file, it would have no successor; hence, its successor pointer would be coded as 7777. The number of available columns in NSET is equal to the value read in for ID. Therefore, the column ID is the last possible column for storing entries and its successor pointer is always given the code value of 8888.

```
      SUBROUTINE SET (JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
C
C*****INIT SHOULD BE ONE FOR INITIALIZATION OF FILE
C
      IF (INIT-1) 5,1,5
C
C*****INITIALIZE FILE TO ZERO.  SET UP POINTERS
C*****MUST INITIALIZE KRANK(JQ)
C*****MUST INITIALIZE INN(JQ)****INN(JQ)=1 IS FIFO**INN(JQ)=2 IS LIFO
C
1     KOL=7777
      KOF=8888
      KLE=9999
      MX=IM+1
      MXX=IM+2
C
C*****INITIALIZE POINTING CELLS OF NSET AND ZERO OTHER CELLS OF NSET
C
      DO 3 I=1,ID
      DO 2 J=1,IM
      NSET(J,I)=0
2     CONTINUE
      NSET(MXX,I)=I-1
      NSET(MX,I)=I+1
3     CONTINUE
      NSET(MX,ID)=KOF
      DO 4 K=1,NOQ
      NQ(K)=0
      MLC(K)=0
      MFE(K)=0
      MAXNQ(K)=0
      MLE(K)=0
      ENQ(K)=0.0
      VNQ(K)=0.
      QTIME(K)=TNOW
4     CONTINUE
C
C*****FIRST AVAILABLE COLUMN = 1
C
      MFA=1
      INIT=0
      OUT=0.0
      RETURN
C
C*****MFEX IS FIRST ENTRY IN FILE WHICH HAS NOT BEEN COMPARED WITH ITEM
C*****TO BE INSERTED
```

```
C
5       MFEX=MFE(JQ)
C
C*****KNT IS A CHECK CODE TO INDICATE THAT NO COMPARISONS HAVE BEEN MADE
C
        KNT=2
C
C*****KS IS THE ROW ON WHICH ITEMS OF FILE JQ ARE RANKED
        KS=KRANK(JQ)
C*****TEST FOR PUTTING VALUE IN OR OUT
C*****IF OUT EQUALS ONE AN ITEM IS TO BE REMOVED FROM FILE JQ.  IF OUT
C*****IS LESS THAN ONE AN ITEM IS TO BE INSERTED IN FILE JQ
C
        IF (OUT-1.0) 6,25,36
C
C*****PUTTING AN ENTRY IN FILE JQ
C*****NXFA IS THE SUCCESSOR COLUMN OF THE FIRST AVAILABLE COLUMN FOR
C*****STORING INFORMATION
C*****THE ITEM TO BE INSERTED WILL BE PUT IN COLUMN MFA
C
6       NXFA=NSET(MX,MFA)
C
C*****IF INN(JQ) EQUALS TWO THE FILE IS A HVF FILE. IF INN(JQ) IS
C*****ONE THE FILE IS A LVF FILE. FOR LVF FILES TRY TO INSERT
C*****STARTING AT END OF FILE.  MLEX IS LAST ENTRY IN FILE WHICH HAS
C*****NOT BEEN COMPARED WITH ITEMS TO BE INSERTED.
C
        IF (INN(JQ)-1) 36,7,19
7       MLEX=MLE(JQ)
C
C*****IF MLEX IS ZERO FILE IS EMPTY.  ITEM TO BE INSERTED WILL BE ONLY
C*****ITEM IN FILE.
C
        IF (MLEX) 36,8,13
8       NSET(MXX,MFA)=KLE
        MFE(JQ)=MFA
C
C*****THERE IS NO SUCCESSOR OF ITEM INSERTED.  SINCE ITEM WAS INSERTED
C*****IN COLUMN MFA THE LAST ENTRY OF FILE JQ IS IN COLUMN MFA.
C
9       NSET(MX,MFA)=KOL
        MLE(JQ)=MFA
C
C*****SET NEW MFA EQUAL TO SUCCESSOR OF OLD MFA.  THAT IS NXFA.  THE
C*****NEW MFA HAS NO PREDECESSOR SINCE IT IS THE FIRST AVAILABLE COLUMN
C*****FOR STORAGE.
C
10      MFA=NXFA
        IF (MFA-KOF) 11,12,12
11      NSET(MXX,MFA)=KLE
C
C*****UPDATE STATISTICS OF FILE JQ
C
12      XNQ=NQ(JQ)
        ENQ(JQ)=ENQ(JQ)+XNQ*(TNOW-QTIME(JQ))
```

```
      VNQ(JQ)=VNQ(JQ)+XNQ*XNQ*(TNOW-QTIME(JQ))
      QTIME(JQ)=TNOW
      NQ(JQ)=NQ(JQ)+1
      MAXNQ(JQ)=XMAX(MAXNQ(JQ),NQ(JQ))
      MLC(JQ)=MFE(JQ)
      RETURN
C
C*****TEST RANKING VALUE OF NEW ITEM AGAINST VALUE OF ITEM IN COLUMN
C*****MLEX
C
13    IF (NSET(KS,MFA)-NSET(KS,MLEX)) 16,14,14
C
C*****INSERT ITEM AFTER COLUMN MLEX.  LET SUCCESSOR OF MLEX BE MSU.
C
14    MSU=NSET(MX,MLEX)
      NSET(MX,MLEX)=MFA
      NSET(MXX,MFA)=MLEX
      GO TO (15,9), KNT
C
C*****SINCE KNT EQUALS ONE A COMPARISON WAS MADE AND THERE IS A
C*****SUCCESSOR TO MLEX, I.E., MSU IS NOT EQUAL TO KOL.  POINT COLUMN
C*****MFA TO MSU AND VICE VERSA.
C
15    NSET(MX,MFA)=MSU
      NSET(MXX,MSU)=MFA
      GO TO 10
C
C*****SET KNT TO ONE SINCE A COMPARISON WAS MADE.
C
16    KNT=1
C
C*****TEST MFA AGAINST PREDECESSOR OF MLEX BY LETTING MLEX EQUAL
C*****PREDECESSOR OF MLEX.
C
      MLEX=NSET(MXX,MLEX)
      IF (MLEX-KLE) 13,17,13
C
C*****IF MLEX HAD NO PREDECESSOR MFA IS FIRST IN FILE.
C
17    NSET(MXX,MFA)=KLE
      MFE(JQ)=MFA
C
C*****SUCCESSOR OF MFA IS MFEX AND PREDECESSOR OF MFEX IS MFA. (NOTE AT
C*****THIS POINT MLEX = MFEX IF LVF WAS USED).
C
18    NSET(MX,MFA)=MFEX
      NSET(MXX,MFEX)=MFA
      GO TO 10
C
C***** FOR HVF OPERATION TRY TO INSERT ITEM STARTING AT BEGINNING OF
C*****FILE JQ.
C*****IF MFEX IS 0, NO ENTRIES ARE IN FILE JQ.  THIS CASE WAS CONSIDERED
C*****PREVIOUSLY AT STATEMENT 10.
C
19    IF (MFEX) 36,8,20
```

```
C
C*****TEST RANKING VALUE OF NEW ITEM AGAINST VALUE OF ITEM IN COLUMN
C*****MFEX.
C
20    IF (NSET(KS,MFA)-NSET(KS,MFEX)) 21,22,22
C
C*****IF NEW VALUE IF LOWER, MFA MUST BE COMPARED AGAINST SUCCESSOR OF
C*****MFEX.
C
21    KNT=1
C
C*****LET MPRE = MFEX AND LET MFEX BE THE SUCCESSOR OF MFEX.
C
      MPRE=MFEX
      MFEX=NSET(MX,MFEX)
      IF (MFEX-KOL) 20,24,20
C
C*****IF NEW VALUE IS HIGHER, IT SHOULD BE INSERTED BETWEEN MFEX AND ITS
C*****PREDECESSOR.
C*****IF KNT = 2, MFEX HAS NO PREDECESSOR, GO TO STATEMENT 16.  IF KNT
C*****= 1, A COMPARISON WAS MADE AND A VALUE OF MPRE HAS ALREADY BEEN
C*****OBTAINED ON THE PREVIOUS ITERATION.  SET KNT = 2 TO INDICATE THIS.
C
22    GO TO (23,17), KNT
23    KNT=2
C
C*****MFA IS TO BE INSERTED AFTER MPRE.  MAKE MPRE THE PREDECESSOR OF
C*****MFA AND MFA THE SUCCESSOR OF MPRE.
C
24    NSET(MXX,MFA)=MPRE
      NSET(MX,MPRE)=MFA
C
C*****IF KNT WAS NOT RESET TO 2, THERE IS NO SUCCESSOR OF MFA.  POINTERS
C*****ARE UPDATED AT STATEMENT 17.  IF KNT = 2, IT WAS RESET AND THE
C*****SUCCESSOR OF MFA IS MFEX.
C
      GO TO (9,18), KNT
C
C*****REMOVAL OF AN ITEM FROM FILE JQ.
C
25    OUT=0.0
C
C*****UPDATE POINTING SYSTEM TO ACCOUNT FOR REMOVAL OF MLC (JQ).  COLUMN
C*****REMOVED IS ALWAYS SET TO MLC(JQ) BY SUBROUTINE RMOVE.
C
      MMLC=MLC(JQ)
C
C*****RESET OUT TO 0 AND CLEAR COLUMN REMOVED.  LET JL EQUAL SUCCESSOR
C*****OF COLUMN REMOVED AND JK EQUAL PREDECESSOR OF COLUMN REMOVED.
C*****IF JL = KOL, MLC WAS LAST ENTRY.  IF JK = KLE, MLC WAS FIRST ENTRY
C*****MLC WAS NOT FIRST OR LAST ENTRY.  UPDATE POINTERS SO THAT JL IS
C*****SUCCESSOR OF JK AND JK IS PREDECESSOR OF JL.
C
      DO 26 I=1,IM
      NSET(I,MMLC)=0
```

```
26      CONTINUE
        JL=NSET(MX,MMLC)
        JK=NSET(MXX,MMLC)
        IF (JL-KOL) 27,33,27
27      IF (JK-KLE) 28,32,28
28      NSET(MX,JK)=JL
        NSET(MXX,JL)=JK
C
C*****UPDATE POINTERS.
C
29      NSET(MX,MMLC)=MFA
        NSET(MXX,MMLC)=KLE
        IF (MFA-KOF) 30,31,31
30      NSET(MXX,MFA)=MMLC
31      MFA=MLC(JQ)
        MLC(JQ)=MFE(JQ)
C
C*****UPDATING FILE STATISTICS
C
        XNQ=NQ(JQ)
        ENQ(JQ)=ENQ(JQ)+XNQ*(TNOW-QTIME(JQ))
        VNQ(JQ)=VNQ(JQ)+XNQ*XNQ*(TNOW-QTIME(JQ))
        QTIME(JQ)=TNOW
        NQ(JQ)=NQ(JQ)-1
        RETURN
C
C*****MLC WAS FIRST ENTRY BUT NOT LAST ENTRY.  UPDATE POINTERS.
C
32      NSET(MXX,JL)=KLE
        MFE(JQ)=JL
        GO TO 29
33      IF (JK-KLE) 34,35,34
C
C*****MLC WAS LAST ENTRY BUT NOT FIRST ENTRY.  UPDATE POINTERS.
C
34      NSET(MX,JK)=KOL
        MLE(JQ)=JK
        GO TO 29
C
C*****MLC WAS BOTH THE LAST AND FIRST ENTRY, THEREFORE, IT IS THE ONLY
C*****ENTRY.
C
35      MFE(JQ)=0
        MLE(JQ)=0
        GO TO 29
36      CALL ERROR (88,NSET)
        CALL EXIT
        END
```

Subroutine GEMCS (NSET)

The subroutine maintains the information-handling scheme used in the GEMCS subsystem for the solution of complex networks describing real processes, with a technique that employs sequential, iterative calculations. It first causes the loading of a data set which describes the information flow diagram establishing the relationships between the various components and entities in the simulated process. It then carries out the execution of a series of computations contained in user-supplied modular subroutines each of which describes a set of related operations or activities in the process. This is performed according to the order prescribed in the data set and with the arrangement for information transfer into and out of the unit computations represented by the modular subroutines.

```
      SUBROUTINE GEMCS (NSET)
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      DIMENSION NSET(12,1)
C
C*****EEN --- DIMENSION EQUAL TO TOTAL NUMBER OF ELEMENTS IN ALL MODULES
C          --- IE. LESS THAN OR EQUAL TO DIMENSION ON EN * NCALC
C*****EN --- DIMENSION EQUAL TO LARGEST NUMBER OF ELEMENTS IN ANY MODULE
C*****LLST --- DIMENSION EQUAL TO TOTAL NUMBER OF MODULES INCLUDED
C*****NPOINT --- FIRST DIMENSION EQUAL TO OR GREATER THAN THE LARGEST
C             --- EQUIPMENT NUMBER (= NOE)
C*****NS --- STREAM CODES - DIMENSION EQUAL TO NUMBER OF STREAMS
C*****SI / SO --- FIRST DIMENSION INDICATES NO. OF INPUT/OUTPUT STREAMS
C              --- SECOND DIMENSION INDICATES NO. OF ELEMENTS PER STREAM
C*****SN --- FIRST DIMENSION EQUAL TO III, INDICATES NUMBER OF STREAMS
C         --- STORED ANY TIME (INCLUDING INPUT/OUTPUT STREAMS)
C
C*****NOTE THAT A CHANGE IN THE SN SIZE REQUIRES A CHANGE IN III TO
C*****MATCH THIS CHANGED SIZE
C
C*****NOTE THAT THE SIZE OF THE SN TABLE MUST BE THE NUMBER OF
C*****STREAMS WHICH HAVE A 1, 2, OR 3 STREAM TYPE LABEL, PLUS THE
C*****MAXIMUM NUMBER OF OUTPUT STREAMS FROM AN EQUIPMENT MODULE WHICH
C*****HAVE A 7 LABEL
C
      III=17
      ISP=0
      NCOUNT=0
      IF (ABS(TNOW-TBEG).GT.0.1) GO TO 1
      CALL DLOAD1
1     CONTINUE
C
C*****CALCULATING EQUIPMENT IN CALCULATION ORDER LIST
C
      NC=1
      IK=1
      KTEST=0
      IF (LOOP.EQ.999) KTEST=1
C
C*****PRINT SN TABLE
C
2     IF (KPRNT(4)) 5,5,3
3     WRITE (6,47) LOOP
      DO 4 IKE=1,III
      IF (SN(IKE,1).LE.0.0) GO TO 4
```

```
      WRITE (6,54) (SN(IKE,J),J=1,JJ)
4     CONTINUE
C
C*****READ EN VECTOR
C
5     NE=LLST(NC)
      MM=NE
      CALL DISKIO (1,MM)
      NN=EN(3)+.001
      NIN=EN(6)+.001
      NOUT=EN(11)+.001
      NTYPE=ABS(EN(2))+.001
      KSW=0
      IF (EN(2).LE.0.) KSW=1
      IF (KPRNT(4)) 7,7,6
6     WRITE (6,48) NE
      WRITE (6,54) (EN(I),I=1,NN)
C
C*****FINDING INPUT STREAMS
C
7     DO 14 I=1,NIN
      S=EN(I+6)
      CALL STREAM (-S)
      IF (IS-III) 10,10,8
8     CONTINUE
      SI(I,1)=S
C
C*****IF NO STREAM IS AVAILABLE, SET SI(I,-) = 0.0
C
      DO 9 J=2,JJ
      SI(I,J)=0.
9     CONTINUE
      GO TO 14
10    DO 11 J=1,JJ
      SI(I,J)=SN(IS,J)
11    CONTINUE
      IF (INT(S+.001)) 14,14,12
12    CONTINUE
      M=S+.001
      IF (NS(M)-6) 14,13,13
13    CONTINUE
      SN(IS,1)=0.
14    CONTINUE
      IF (KPRNT(4)) 17,17,15
15    WRITE (6,49) NE
      DO 16 IKE=1,NIN
      WRITE (6,54) (SI(IKE,J),J=1,JJ)
16    CONTINUE
17    CONTINUE
C
C*****FINDING OUTPUT STREAMS
C
      DO 20 I=1,NOUT
      S=EN(I+11)
      CALL STREAM (-S)
```

```
      IF (III-IS) 20,18,18
18    CONTINUE
      DO 19 J=2,JJ
      SO(I,J)=SN(IS,J)
19    CONTINUE
20    SO(I,1)=S
C
C*****CALLING MODULES
C
      CALL MODULE (NTYPE,NSET)
C
C*****STORING OUTPUT STREAMS AND PRINTING
C
      IF (NOUT) 29,29,21
21    CONTINUE
      DO 28 I=1,NOUT
      S=EN(I+11)
      M=S+.001
      IF (NS(M).NE.3) GO TO 22
      IF (KPRNT(6).EQ.2) GO TO 22
      WRITE (6,50) M
      WRITE (6,54) (SO(I,J),J=1,JJ)
22    IF (NS(M)-6) 23,28,23
23    CALL STREAM (-S)
      IF (IS-III) 26,26,24
C
C*****STORES IN NEXT LOCATION AVAILABLE IN SN
C
24    CALL STREAM (0.)
      IF (IS-III) 26,26,25
C
C*****IF NO SPACE IS AVAILABLE IN SN, WRITE - ERROR IN SN -
C
25    WRITE (6,53)
      GO TO 28
C
C*****IF NS(M) = 7, SO S STORED IN SN TABLE TEMPORARILY FOR IMMEDIATE
C*****USE IN NEXT CALCULATION
C
C*****IF NS(M) = 1, 2, OR 3, THEN STREAMS ARE STORED PERMANENTLY
C
26    DO 27 J=1,JJ
      SN(IS,J)=SO(I,J)
27    CONTINUE
28    CONTINUE
C
C*****STORE INPUT STREAMS IF DESIRED
C
29    CONTINUE
      IF (NIN.LE.0) GO TO 33
      DO 32 I=1,NIN
      S=EN(I+6)
      M=S+0.001
      IF (NS(M).GE.6) GO TO 32
      CALL STREAM (-S)
```

```
      IF (IS.LE.III) GO TO 30
      CALL STREAM (0.)
      IF (IS.LE.III) GO TO 30
      WRITE (6,53)
      GO TO 32
30    DO 31 J=1,JJ
      SN(IS,J)=SI(I,J)
31    CONTINUE
32    CONTINUE
33    CONTINUE
      IF (ISP) 36,36,34
34    CONTINUE
      IF (KPRNT(6).EQ.2) GO TO 36
      WRITE (6,51) NE
      DO 35 IKE=1,NOUT
      WRITE (6,54) (SO(IKE,J),J=1,JJ)
35    CONTINUE
36    IF (LOOP-999) 40,37,37
37    IF (KSW.NE.0) GO TO 40
      IF (NOUT) 40,40,38
38    CONTINUE
      IF (KPRNT(6).EQ.2) GO TO 40
      WRITE (6,52) NE
      DO 39 IKE=1,NOUT
      WRITE (6,54) (SO(IKE,J),J=1,JJ)
39    CONTINUE
40    NC=NC+1
      IF (NC-NCALC) 2,2,41
41    CONTINUE
      IF (LOOP-999) 42,43,43
42    CONTINUE
      NC=0
      GO TO 40
43    IF (KTEST) 44,44,46
44    NC=0
      DO 45 IG=1,5
      KPRNT(IG)=0
45    CONTINUE
      KTEST=1
      GO TO 40
46    IF (KPRNT(7).EQ.1) WRITE (6,55)
      RETURN
C
C
47    FORMAT (1H0,26H SN TABLE ON ENTERING LOOP,I5)
48    FORMAT (1H0,20H MODULE SET FOR UNIT,I5)
49    FORMAT (1H0,25H INPUT STREAMS FOR MODULE,I5)
50    FORMAT (1H0,26H OUTPUT FOR PRODUCT STREAM,I5)
51    FORMAT (1H0,26H OUTPUT STREAMS FOR MODULE,I5)
52    FORMAT (1H0,32H FINAL OUTPUT STREAMS FOR MODULE,I5)
53    FORMAT (1H0,12H ERROR IN SN)
54    FORMAT (1H0,5F15.5)
55    FORMAT (1H0,17H END OF EXECUTION/1H1)
      END
```

Subroutine STREAM (S)

The subroutine is for locating information in the system SN matrix which stores information that is transferred between unit computations.

```
      SUBROUTINE STREAM (S)
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      DO 1 IS=1,III
      KLK=IS
      IF (ABS(SN(IS,1)-ABS(S))-.010) 3,1,1
1     CONTINUE
      IS=KLK+1
      IF (S) 3,3,2
2     CONTINUE
      WRITE (6,6) S
3     IF (KPRNT(5)) 5,5,4
4     WRITE (6,7) IS
5     CONTINUE
      RETURN
C
C
6     FORMAT (1H0,15H ERROR...STREAM,F3.0,17H NOT IN SN MATRIX)
7     FORMAT (1H0,42H STREAM NUMBER OF STREAM BEING SEARCHED = ,I3)
      END
```

Subroutine DISKIO (IPNT, MM)

The subroutine transfers the appropriate part of the GEMCS data set used by each unit computation between the temporary vector, EN, and the main storage vector, EEN, during the simulation.

```
      SUBROUTINE DISKIO (IPNT,MM)
C
C*****THIS SUBROUTINE SIMULATES DISKIO ON360/30
C
C*****IF IPNT=1.  READS FROM MODULE SETS TABLE
C*****IF IPN=2. WRITES ONTO MODULE SETS TABLE
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      MQ=NPOINT(MM,1)
      ML=NPOINT(MM,2)
      IF (IPNT-1) 3,1,3
1     DO 2 I=1,ML
      EN(I)=EEN(MQ+I)
2     CONTINUE
      GO TO 7
3     IF (IPNT-2) 6,4,6
4     DO 5 I=1,ML
      EEN(MQ+I)=EN(I)
5     CONTINUE
      GO TO 7
6     WRITE (6,8) IPNT,MM
7     CONTINUE
      RETURN
C
C
8     FORMAT (1H0,22H DISKIO ERROR-IPNT,MM=,I5,1X,I5)
      END
```

Subroutine DLOAD1

The subroutine transfers data into the appropriate storage locations pertaining to the GEMCS subsystem in the system common data pool. This is done in a straightforward manner using a consistent data format of 5F12.5. Attention is directed to the limits on the various "DO LOOPS" which should correspond to the length of the vectors, EN, SI, SN and SO, as specified in the system "COMMON". It is possible by the correct setting of values of elements in the vector, KPRNT, to cause an "echo check" of the data as they are entered from cards, and to control the printout of simulation data on the conditions in the information flow during the run.

```
      SUBROUTINE DLOAD1
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      DIMENSION AKPRNT(10), ALLST(20), ANS(25), TITL1(18), TITL2(18)
C
C*****AKPRNT - DIMENSION SAME AS KPRNT
C*****ALLST - DIMENSION SAME AS LLST
C*****ANS - DIMENSION SAME AS NS
C
C*****  --  PRINTING CONTROL CHARACTERS  --
C
C*****KPRNT(1)=1  CAUSES PRINTING OF TOT. NO. OF MODULES, CALCULATION
C                 ORDER LIST, AND STREAM CODES
C*****KPRNT(1)=0  SUPPRESSES ABOVE PRINTING
C*****KPRNT(2)=1  CAUSES PRINTING OF INITIAL STREAMS
C*****KPRNT(2)=0  SUPPRESSES ABOVE PRINTING
C*****KPRNT(3)=1  CAUSES PRINTING OF MODULES SETS
C*****KPRNT(3)=0  SUPPRESSES ABOVE PRINTING
C*****KPRNT(4)=1  CAUSES PRINTING OF SN TABLE ON ENTERING LOOP
C*****KPRNT(4)=1  CAUSES PRINTING OF EN VECTOR FOR SPECIFIED MODULE SET
C*****KPRNT(4)=1  CAUSES PRINTING OF SI MATRIX FOR SPECIFIED MODULE SET
C*****KPRNT(4)=1  CAUSES PRINTING OF STREAM NUMBERS ENTERING AND LEAVING
C                 SPECIFIED MODULES (RE - PIECE OF EQUIPMENT)
C*****KPRNT(4)=0  SUPPRESSES ABOVE PRINTING
C*****KPRNT(5)=1  CAUSES PRINTING OF NCOUNT AND IS
C*****KPRNT(5)=0  SUPPRESSES ABOVE PRINTING
C*****KPRNT(6)=0  CAUSES PRINTING OF ALL OUTPUT FOR PRODUCT STREAMS
C                 AND FINAL OUTPUT STREAMS
C*****KPRNT(6)=2  SUPPRESSES ABOVE PRINTING
C*****KPRNT(7)=1  CAUSES PRINTING OF - END OF EXECUTION - , AND
C                 AND OTHER PRINTING FROM USER-SUPPLIED MODULES
C*****KPRNT(7)=0  SUPPRESSES ABOVE PRINTING
C
C     ------------------------------------------------------------------
C     -----
C
C*****NOTE THAT THE LIMIT ON THIS DO STATEMENT MUST CORRESPOND TO THE
C*****LENGTH OF THE EN VECTOR
C
      DO 1 I=1,25
      EN(I)=0.
1     CONTINUE
C
C     ------------------------------------------------------------------
C     -----
C
C*****NOTE THAT THE LIMIT ON THIS DO STATEMENT MUST CORRESPOND TO THE
C*****DIMENSIONS OF THE SI AND SO MATRICES
C
      DO 3 I=1,10
      DO 2 J=1,4
```

```
      SI(J,I)=0.
      SO(J,I)=0.
2     CONTINUE
      DO 3 IK=1,III
3     SN(IK,I)=0.
C
C     ------------------------------------------------------------------
C     -----
C
      READ (5,34) TITL1
      PRINT 34, TITL1
      READ (5,34) TITL2
      PRINT 34, TITL2
      READ (5,40) AKPRNT
      DO 4 I=1,10
      KPRNT(I)=AKPRNT(I)
4     CONTINUE
C
C*****READ NUMBER OF MODULES IN CALCULATION ORDER,
C*****AND NUMBER OF COMPONENTS
C*****(FOR ITERATIVE CALCULATIONS WITH LOOP GREATER THAN 1,
C*****NCALC IS READ IN WITH A NEGATIVE SIGN)
C
      READ (5,40) ANCALC,ANOCOM
      NCALC=ANCALC
      NOCOMP=ANOCOM
      IF (KPRNT(1)-1) 6,5,6
5     WRITE (6,36) NCALC,NOCOMP
6     CONTINUE
      IF (-NCALC) 8,7,7
7     NCALC=-NCALC
      LOOP=1
      GO TO 9
8     LOOP=999
C
C*****READ CALCULATION ORDER
C
9     READ (5,40) (ALLST(I),I=1,NCALC)
      DO 10 I=1,NCALC
      LLST(I)=ALLST(I)
10    CONTINUE
      IF (KPRNT(1)-1) 12,11,12
11    WRITE (6,37) (LLST(I),I=1,NCALC)
12    CONTINUE
C
C*****READ STREAM CODES
C*****MSN IS THE MAXIMUM STREAM NUMBER - IF NEGATIVE, OUTPUT STREAMS
C*****WILL BE PRINTED DURING EXECUTION
C
      READ (5,40) AMSN
      MSN=AMSN
      IF (-MSN) 14,13,13
13    MSN=-MSN
      ISP=1
14    IF (KPRNT(1)-1) 16,15,16
```

```
15    WRITE (6,38) MSN
16    READ (5,40) (ANS(I),I=1,MSN)
      DO 17 I=1,MSN
      NS(I)=ANS(I)
17    CONTINUE
      IF (KPRNT(1)-1) 19,18,19
18    WRITE (6,37) (NS(I),I=1,MSN)
19    JJ=NOCOMP+5
      READ (5,40) ANSR
      NSR=ANSR
      IF (KPRNT(2)-1) 21,20,21
20    WRITE (6,39) NSR
21    DO 24 I=1,NSR
      READ (5,40) (SN(I,J),J=1,JJ)
      IF (KPRNT(2)-1) 23,22,23
22    WRITE (6,41) (SN(I,J),J=1,JJ)
23    CONTINUE
24    CONTINUE
C
C*****READ MODULE SETS
C*****NOE IS THE NUMBER OF MODULE SETS TO BE READ
C
      READ (5,40) ANOE
      NOE=ANOE
      IF (KPRNT(3)-1) 26,25,26
25    WRITE (6,42) NOE
26    DO 30 I=1,NOE
C
C     ------------------------------------------------------------------
C     -----
C
C*****NOTE THAT THE LIMIT ON THIS DO STATEMENT MUST CORRESPOND TO THE
C*****LENGTH OF THE EN VECTOR
C
      DO 27 IZ=1,25
      EN(IZ)=0.
27    CONTINUE
C
C     ------------------------------------------------------------------
C     -----
C
      READ (5,40) (EN(N),N=1,5)
      NN=EN(3)
      NCOUNT=NCOUNT+NN
      READ (5,40) (EN(N),N=6,NN)
      IF (KPRNT(3)-1) 29,28,29
28    WRITE (6,41) (EN(N),N=1,NN)
29    MM=EN(1)+.001
      NPOINT(MM,1)=NCOUNT-NN
      NPOINT(MM,2)=NN
      CALL DISKIO (2,MM)
30    CONTINUE
      IF (KPRNT(5).GT.0) GO TO 31
      GO TO 33
31    DO 32 I=1,NOE
```

```
      WRITE (6,35) (NPOINT(I,J),J=1,2)
32    CONTINUE
33    CONTINUE
      RETURN
C
C
34    FORMAT (18A4)
35    FORMAT (2I5)
36    FORMAT (1H0,I5,12H MODULE SETS,I5,11H COMPONENTS//25H  CALCULATION
     1 ORDER LIST-)
37    FORMAT (1X,10I5)
38    FORMAT (1H0,I5,23H STREAM CODES ARE READ-)
39    FORMAT (1H0,I5,26H INITIAL STREAMS ARE READ-)
40    FORMAT (5F12.5)
41    FORMAT (1X,5F12.3)
42    FORMAT (1H0,I5,22H MODULE SETS ARE READ-)
      END
```

Subroutine FIND (XVAL, MCODE, JQ, JATT, KCOL, NSET)

The subroutine locates in File JQ of NSET a column, KCOL, which has an attribute value in JATT related to XVAL according to the code MCODE. KCOL is the unknown entry. Values must be specified for the first four of the following six arguments:

(1) XVAL - A value used to locate a column in NSET containing an attribute value having a specified relationship to XVAL.

(2) MCODE - A code specifying the relationship between XVAL and the attribute value. The five options available are:

(a) MCODE = 1: maximum value greater than XVAL.

(b) MCODE = 2: minimum value greater than XVAL.

(c) MCODE = 3: maximum value less than XVAL.

(d) MCODE = 4: minimum value less than XVAL.

(e) MCODE = 5: value equal to XVAL.

(3) JQ - The file containing the column to be located. JQ can be 1, 2, ... , NOQ.

(4) JATT - The row of NSET containing the attribute value to be used in finding KCOL. A value for JATT from 1 to IM+2 must be specified. JATT may be an attribute number, the successor row, or the predecessor row.

(5) KCOL - A column containing the located entry, defined as a result of the search conducted by FIND. If no column in File JQ meets the condition specified by MCODE, KCOL is set to zero.

(6) NSET - The location of NSET.

```
      SUBROUTINE FIND (XVAL,MCODE,JQ,JATT,KCOL,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
C
C*****CHANGE VALUE TO FIXED POINT WHEN SEARCHING NSET
C
      NVAL=XVAL*SCALE
C
C*****THE COLUMN WHICH IS THE BEST CANDIDATE IS KBEST
C
      KBEST=0
C
C*****THE NEXT COLUMN TO BE CONSIDERED AS A CANDIDATE IS NEXTK
C
      NEXTK=MFE(JQ)
      IF (NEXTK) 1,2,3
1     CALL ERROR (89,NSET)
2     KCOL=KBEST
      RETURN
C
C*****MGRNV IS +1 FOR GREATER THAN SEARCH AND -1 FOR LESS THAN SEARCH
C*****NMAMN IS +1 FOR MAXIMUM AND -1 FOR MINIMUM
C*****FOR SEARCH FOR EQUALITY THE SIGN OF MGRNV AND NMAMN ARE NOT USED
C
3     GO TO (4,5,6,7,4), MCODE
4     MGRNV=1
      NMAMN=1
      GO TO 8
5     MGRNV=1
      NMAMN=-1
      GO TO 8
6     MGRNV=-1
      NMAMN=1
      GO TO 8
7     MGRNV=-1
      NMAMN=-1
8     IF (MGRNV*(NSET(JATT,NEXTK)-NVAL)) 14,9,10
C
C*****WHEN EQUALITY IS OBTAINED TEST FOR MCODE=5, THE SEARCH FOR A
C*****SPECIFIED VALUE
C
9     IF (MCODE-5) 14,15,14
10    IF (MCODE-5) 11,14,11
11    IF (KBEST) 1,13,12
12    IF (NMAMN*(NSET(JATT,NEXTK)-NSET(JATT,KBEST))) 14,14,13
13    KBEST=NEXTK
14    NEXTK=NSET(MX,NEXTK)
      IF (NEXTK-7777) 8,2,2
```

```
15    KCOL=NEXTK
      RETURN
      END
```

Subroutine ERROR (J, NSET)

The subroutine causes the printing of the error code J, NSET and File 1 (event-file). It then calls the subroutines SUMRY and EXIT to terminate the run. The error codes are as follows:

Error Code	Subprogram in which Error Occurred
84	PRODQ
85	SUMQ
87	FILEM
88	SET
89	FIND
90	COLCT
91	TMST
93	GASP
94	PRNTQ
95	DATAN
97	RMOVE
98	SUMRY
99	MONTR

```
      SUBROUTINE ERROR (J,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      PRINT 3, J
      JEVNT=101
C
C*****PRINT FILING ARRAY NSET
C
      CALL MONTR (NSET)
      PRINT 4
C
C*****PRINT NEXT EVENT FILE
C
      CALL PRNTQ (1,NSET)
C
C*****PRINT SUMMARY REPORT UP TO PRESENT
C
      CALL SUMRY (NSET)
      IF (JEVNT-101) 1,2,1
1     RETURN
2     CALL EXIT
C
C
3     FORMAT (///36X,16HERROR EXIT, TYPE,I3,7H ERROR.)
4     FORMAT (1H1,41X,16HSCHEDULED EVENTS//)
      END
```

Other Subprograms

Name	Function
FILEM (JQ, NSET)	Stores the vector ATRIB in File JQ of NSET.
CHECK (KCOL, JQ, NSET)	Checks and copies column KCOL from File JQ of NSET and stores it in the vector ATRIB. Values of KCOL and JQ must be specified.
RMOVE (KCOL, JQ, NSET)	Removes column KCOL from File JQ of NSET and stores it in the vector ATRIB. Values of KCOL and JQ must be specified.
MONTR (NSET)	Enables a programmer to print NSET at any time during a simulation or to trace the events during a portion of a simulation.
DRAND (ISEED)	Generates a random number, DRAND. The computer system's random number routine provided here, RANF(X), is in the library of the CDC 6400 at McMaster. ISEED is the random number generator seed.
UNFRM (A, B)	Generates a random deviate from a uniform distribution between A and B.
RNORM (J)	Generates a random deviate from a normal distribution with parameters PARAM (J, I).

Name	Function
RLOGN (J)	Generates a random deviate from a lognormal distribution with parameters PARAM (J, I).
ERLNG (J)	Generates a random deviate from an Erlang distribution with parameters in PARAM (J, I).
NPOSN (J, NPSSN)	Generates a random deviate from a Poisson distribution with parameters PARAM (J, I).
SUMQ (JATT, JQ, NSET)	Calculates the sum of the values of the attributes stored in row JATT of File JQ.
PRODQ (JATT, JQ, NSET)	Calculates the product of the values of the attributes stored in row JATT of File JQ.
AMIN (ARG1, ARG2)	Sets AMIN to the minimum of ARG1 and ARG2.
AMAX (ARG1, ARG2)	Sets AMAX to the maximum of ARG1 and ARG2.
XMAX (IARG1, IARG2)	Sets XMAX to the maximum of IARG1 and IARG2.
COLCT (X, N, NSET)	Sets X_i = X and collects: $\sum_i X_i$ in SUMA (N, 1) $\sum_i X_i^2$ in SUMA (N, 2) Number of observations of type N in SUMA (N, 3): $\min_i(X_i)$ in SUMA (N, 4) $\max_i(X_i)$ in SUMA (N, 5)

Name	Function
TMST (X, T, N, NSET)	Sets X_i = X, computes the time since the last change in variable N, TT_i, and collects: $\Sigma_i TT_i$ in SSUMA (N, 1); $\Sigma_i X_i * TT_i$ in SSUMA (N, 2); $\Sigma_i X_i^2 * TT_i$ in SSUMA (N, 3); $\min_i (X_i)$ in SSUMA (N, 4); $\max_i (X_i)$ in SSUMA (N, 5)
HISTO (X1, A, W, N)	X1 is a value to be used to form histogram N. A is the lower limit. W is the cell width.
PRNTQ (JQ, NSET)	Prints average, standard deviation, maximum number of entries and contents of File JQ.
SUMRY (NSET)	Prints a final GASP summary report.
OTPUT (NSET)	User-supplied subroutine to provide additional output at the end of a simulation run.
EVNTS (I, NSET)	User-supplied subroutine provided to call the subroutine defining event I.
MODULE (NT, NSET)	User-supplied subroutine provided to call the subroutine defining unit computation NT.

```
      SUBROUTINE FILEM (JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
C
C*****TEST TO SEE IF THERE IS AN AVAILABLE COLUMN FOR STORAGE
C
      IF (MFA-ID) 2,2,1
1     PRINT 5
      CALL ERROR (87,NSET)
C
C*****PUT ATTRIBUTE VALUES IN FILE
C
2     DO 4 I=1,IM
      DEL=.000001
      IF (ATRIB(I)) 3,4,4
3     DEL=-.000001
4     NSET(I,MFA)=SCALE*(ATRIB(I)+DEL)
C
C*****CALL SET TO PUT NEW ENTRY IN PROPER PLACE IN NSET
C
      CALL SET (JQ,NSET)
      RETURN
C
C
5     FORMAT (//24H OVERLAP SET GIVEN BELOW/)
      END
```

```
      SUBROUTINE CHECK (KCOL,JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      IF (KCOL) 1,1,2
1     WRITE (6,4)
      RETURN
2     MLC(JQ)=KCOL
C
C*****PUT VALUES OF KCOL IN ATRIB
C
      DO 3 I=1,IM
      ATRIB(I)=NSET(I,KCOL)
      ATRIB(I)=ATRIB(I)/SCALE
3     CONTINUE
      RETURN
C
4     FORMAT (1H0,* ERROR IN SUBROUTINE CHECK - COLUMN KCOL NOT IN NSET
     1*)
      END
```

```
      SUBROUTINE RMOVE (KCOL,JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      IF (KCOL) 1,1,2
1     CALL ERROR (97,NSET)
2     MLC(JQ)=KCOL
C
C*****PUT VALUES OF KCOL IN ATTRIB
C
      DO 3 I=1,IM
      ATRIB(I)=NSET(I,KCOL)
      ATRIB(I)=ATRIB(I)/SCALE
3     CONTINUE
C
C*****SET OUT=1 AND CALL SET TO REMOVE ENTRY FROM NSET
C
      OUT=1.
      CALL SET (JQ,NSET)
      RETURN
      END
```

```
      SUBROUTINE MONTR (NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
C
C*****IF JEVNT .GE. 101, PRINT NSET
C
      IF (JEVNT-101) 3,1,3
1     PRINT 9, TNOW
      DO 2 I=1,ID
      PRINT 10, I,(NSET(J,I),J=1,MXX)
2     CONTINUE
      RETURN
3     IF (MFE(1)) 5,8,4
C
C*****IF JMNIT = 1,PRINT TNOQ,CURRENT EVENT CODE, AND ALL ATTRIBUTES OF
C*****THE NEXT EVENT
C
4     IF (JMNIT-1) 7,6,5
5     PRINT 11
      CALL EXIT
6     MMFE=MFE(1)
      PRINT 12, TNOW,ATRIB(2),(NSET(I,MMFE),I=1,MXX)
7     RETURN
8     PRINT 13, TNOW
      GO TO 7
C
C
9     FORMAT (1H1,10X31H**GASP JOB STORAGE AREA DUMP AT,F10.4,2X,12HTIME
     1 UNITS**//)
10    FORMAT (12I10)
11    FORMAT (///36X,26H ERROR EXIT,TYPE 99 ERROR.)
12    FORMAT (/10X,23HCURRENT EVENT....TIME =,F8.2,5X,7HEVENT =,F7.2,/10
     1X,17HNEXT EVENT.......,(6I8))
13    FORMAT (10X,19H FILE 1 IS EMPTY AT,F10.2)
      END
```

```
      FUNCTION DRAND (ISEED)
      IF (ISEED) 1,2,3
1     X=-ISEED
      DRAND=RANF(X)
      ISEED=0
      RETURN
2     X=ISEED
      DRAND=RANF(X)
      RETURN
3     X=ISEED
      DRAND=RANF(X)
      ISEED=0
      RETURN
      END
```

```
      FUNCTION UNFRM (A,B)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
C*****THIS CARD IS TO MAINTAIN THE PROPER SEQUENCING
      UNFRM=A+(B-A)*DRAND(ISEED)
      RETURN
      END
```

```
      FUNCTION RNORM (J)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      RA=DRAND(ISEED)
      RB=DRAND(ISEED)
      V=(-2.0*ALOG(RA))**0.5*COS(6.283*RB)
      RNORM=V*PARAM(J,4)+PARAM(J,1)
      IF (RNORM-PARAM(J,2)) 1,2,3
1     RNORM=PARAM(J,2)
2     RETURN
3     IF (RNORM-PARAM(J,3)) 2,2,4
4     RNORM=PARAM(J,3)
      RETURN
      END
```

```
      FUNCTION RLOGN (J)
C
C*****THE PARAMETERS USED WITH RLOGN ARE THE MEAN AND STANDARD DEVIATION
C*****OF A NORMAL DISTRIBUTION
C
      VA=RNORM(J)
      RLOGN=EXP(VA)
      RETURN
      END
```

```
      FUNCTION ERLNG (J)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      K=PARAM(J,4)
      IF (K-1) 1,2,2
1     PRINT 8, J
      CALL EXIT
2     R=1
      DO 3 I=1,K
      R=R*DRAND(ISEED)
3     CONTINUE
      ERLNG=-PARAM(J,1)*ALOG(R)
      IF (ERLNG-PARAM(J,2)) 4,5,6
4     ERLNG=PARAM(J,2)
5     RETURN
6     IF (ERLNG-PARAM(J,3)) 5,5,7
7     ERLNG=PARAM(J,3)
      RETURN
C
C
8     FORMAT (/16HK = 0 FOR ERLNG,I7)
      END
```

```
      SUBROUTINE NPOSN (J,NPSSN)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      NPSSN=0
      P=PARAM(J,1)
      IF (P-6.0) 1,1,4
1     Y=EXP(-P)
      X=1.0
2     X=X*DRAND(ISEED)
      IF (X-Y) 5,3,3
3     NPSSN=NPSSN+1
      GO TO 2
4     TEMP=PARAM(J,4)
      PARAM(J,4)=(PARAM(J,1))**.5
      NPSSN=RNORM(J)
      PARAM(J,4)=TEMP
      IF (NPSSN) 4,5,5
5     KK=PARAM(J,2)
      KKK=PARAM(J,3)
      NPSSN=KK+NPSSN
      IF (NPSSN-KKK) 7,7,6
6     NPSSN=PARAM(J,3)
7     RETURN
      END
```

```
      FUNCTION SUMQ (JATT,JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      SUMQ=0
      IF (JQ-NOQ) 2,2,1
1     CALL ERROR (85,NSET)
2     IF (NQ(JQ)) 3,3,4
3     RETURN
4     MTEM=MFE(JQ)
5     VSET=NSET(JATT,MTEM)
      SUMQ=SUMQ+VSET/SCALE
      IF (NSET(MX,MTEM)-7777) 6,7,6
6     MTEM=NSET(MX,MTEM)
      GO TO 5
7     RETURN
      END
```

```
      FUNCTION PRODQ (JATT,JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      PRODQ=1.
      IF (JQ-NOQ) 2,2,1
1     CALL ERROR (84,NSET)
2     IF (NQ(JQ)) 3,3,4
3     PRODQ=0.
      RETURN
4     MTEM=MFE(JQ)
5     VSET=NSET(JATT,MTEM)
      PRODQ=PRODQ*VSET/SCALE
      IF (NSET(MX,MTEM)-7777) 6,7,6
6     MTEM=NSET(MX,MTEM)
      GO TO 5
7     RETURN
      END
```

```
      FUNCTION AMIN (ARG1,ARG2)
      IF (ARG1-ARG2) 1,1,2
1     AMIN=ARG1
      RETURN
2     AMIN=ARG2
      RETURN
      END
```

```
      FUNCTION AMAX (ARG1,ARG2)
      IF (ARG1-ARG2) 2,1,1
1     AMAX=ARG1
      RETURN
2     AMAX=ARG2
      RETURN
      END
```

```
      FUNCTION XMAX (IARG1,IARG2)
      IF (IARG1-IARG2) 2,2,1
1     XMAX=IARG1
      RETURN
2     XMAX=IARG2
      RETURN
      END
```

```
      SUBROUTINE COLCT (X,N,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      IF (N) 1,1,2
1     CALL ERROR (90,NSET)
2     IF (N-NCLCT) 3,3,1
3     SUMA(N,1)=SUMA(N,1)+X
      SUMA(N,2)=SUMA(N,2)+X*X
      SUMA(N,3)=SUMA(N,3)+1.0
      SUMA(N,4)=AMIN(SUMA(N,4),X)
      SUMA(N,5)=AMAX(SUMA(N,5),X)
      RETURN
      END
```

```
      SUBROUTINE TMST (X,T,N,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      IF (N) 1,1,2
1     CALL ERROR (91,NSET)
2     IF (N-NSTAT) 3,3,1
3     TT=T-SSUMA(N,1)
      SSUMA(N,1)=SSUMA(N,1)+TT
      SSUMA(N,2)=SSUMA(N,2)+X*TT
      SSUMA(N,3)=SSUMA(N,3)+X*X*TT
      SSUMA(N,4)=AMIN(SSUMA(N,4),X)
      SSUMA(N,5)=AMAX(SSUMA(N,5),X)
      RETURN
      END
```

```
      SUBROUTINE HISTO (X1,A,W,N)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
1     IF (N-NHIST) 3,3,2
2     PRINT 9, N
      CALL EXIT
3     IF (N) 2,2,4
C
C*****TRANSLATE X1 BY SUBTRACTING A IF X.LE.A THEN ADD 1 TO FIRST CELL
C
4     X=X1-A
      IF (X) 5,6,6
5     IC=1
      GO TO 8
C
C*****DETERMINE CELL NUMBER IC. ADD 1 FOR LOWER LIMIT CELL AND 1 FOR
C*****TRUNCATION
C
6     IC=X/W+2.
      IF (IC-NCELS(N)-1) 8,8,7
7     IC=NCELS(N)+2
8     JCELS(N,IC)=JCELS(N,IC)+1
      RETURN
C
C
9     FORMAT (19H ERROR IN HISTOGRAM,I4//)
      END
```

```
      SUBROUTINE PRNTQ (JQ,NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      PRINT 11, JQ
      IF (TNOW-TBEG) 1,1,2
1     PRINT 9
      GO TO 4
C
C*****COMPUTE EXPECT NO. IN FILE JQ UP TO PRESENT THIS MAY BE USEFUL
C*****IN SETTING THE VALUE OF ID
C
2     XNQ=NQ(JQ)
      X=(ENQ(JQ)+XNQ*(TNOW-QTIME(JQ)))/(TNOW-TBEG)
      DENOM = (TNOW - TBEG) - X*X
      IF (DENOM .LT. 0.00000001 .AND. DENOM .GT. -0.00000001) GO TO 16
      STD=((VNQ(JQ)+XNQ*XNQ*(TNOW-QTIME(JQ)))/(TNOW-TBEG)-X*X)
      IF (STD .LT. 0.0) PRINT 19, STD
      IF (STD .LT. 0.0) STD = - STD
      STD = STD**0.5
      PRINT 15, X,STD,MAXNQ(JQ)
      GO TO 17
16    CONTINUE
      PRINT 18, X,MAXNQ(JQ)
17    CONTINUE
C
C*****PRINT FILE IN PROPER ORDER REQUIRES TRACING THROUGH THE POINTERS
C*****OF THE FILE
C
      LINE=MFE(JQ)
      IF (LINE-1) 3,5,5
3     PRINT 13
4     RETURN
5     PRINT 12
6     DO 7 I=1,IM
      ATRIB(I)=NSET(I,LINE)
      ATRIB(I)=ATRIB(I)/SCALE
7     CONTINUE
      PRINT 14, (ATRIB(I),I=1,IM)
      LINE=NSET(MX,LINE)
      IF (LINE-7777) 6,4,8
8     PRINT 10
      CALL EXIT
C
9     FORMAT (/35X,25H NO PRINTOUT TNOW = TBEG //)
10    FORMAT (///36X,26HERROR EXIT, TYPE 94 ERROR.)
11    FORMAT (//39X,25H FILE PRINTOUT, FILE  NO.,I3)
12    FORMAT (/45X,14H FILE CONTENTS/)
13    FORMAT (/43X,18HTHE FILE  IS EMPTY)
```

```
14    FORMAT (30X,8F10.4)
15    FORMAT (/35X,27HAVERAGE NUMBER IN FILE  WAS,F10.4,/35X,9HSTD. DEV
     1,18X,F10.4,/35X,7HMAXIMUM,24X,I4)
18    FORMAT (/35X,27HAVERAGE NUMBER IN FILE  WAS,F10.4,/35X,
     1*STD. DEV. IS INFINITELY LARGE*,/35X,7HMAXIMUM,24X,I4)
19    FORMAT (1H0,* ERROR IN STANDARD DEVIATION (- PRNTQ -) *,F12.4)
      END
```

```
      SUBROUTINE SUMRY (NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      PRINT 19
      PRINT 31, NAME
      PRINT 20, NPROJ,MON,NDAY,NYR,NRUN
      IF (NPRMS) 3,3,1
1     DO 2 I=1,NPRMS
      PRINT 21, I,(PARAM(I,J),J=1,NEPRM)
2     CONTINUE
3     IF (NCLCT) 4,9,5
4     PRINT 22
      CALL EXIT
5     PRINT 23
C
C*****COMPUTE AND PRINT STATISTICS GATHERED BY CLCT
C
      DO 8 I=1,NCLCT
      IF (SUMA(I,3)) 4,6,7
6     PRINT 24, I
      GO TO 8
7     XS=SUMA(I,1)
      XSS=SUMA(I,2)
      XN=SUMA(I,3)
      AVG=XS/XN
      STD=(((XN*XSS)-(XS*XS))/(XN*(XN-1.0)))**.5
      N=XN
      PRINT 25, I,AVG,STD,SUMA(I,4),SUMA(I,5),N
8     CONTINUE
9     IF (NSTAT) 4,14,10
10    PRINT 26
C
C*****COMPUTE AND PRINT STATISTICS GATHERED BY TMST
C
      DO 13 I=1,NSTAT
      IF (SSUMA(I,1)) 4,11,12
11    PRINT 24, I
      GO TO 13
12    XT=SSUMA(I,1)
      XS=SSUMA(I,2)
      XSS=SSUMA(I,3)
      AVG=XS/XT
      STD=(XSS/XT-AVG*AVG)**.5
      PRINT 27, I,AVG,STD,SSUMA(I,4),SSUMA(I,5),XT
13    CONTINUE
14    IF (NHIST) 4,17,15
15    PRINT 28
C
```

```
C*****PRINT HISTOGRAMS
C
      DO 16 I=1,NHIST
      NCL=NCELS(I)+2
      PRINT 29, I,(JCELS(I,J),J=1,NCL)
16    CONTINUE
C
C*****PRINT FILES AND FILE STATISTICS
C
17    DO 18 I=1,NOQ
      CALL PRNTQ (I,NSET)
18    CONTINUE
      RETURN
C
19    FORMAT (1H1,39X,23H**GASP SUMMARY REPORT**/)
20    FORMAT (30X,22HSIMULATION PROJECT NO.,I4//,                    30X,4HD
     1TE,I3,1H/,I3,1H/,I5,12X,10HRUN NUMBER,I5/)
21    FORMAT (20X,14H PARAMETER NO.,I5,4F12.4)
22    FORMAT (///36X,26HERROR EXIT, TYPE 98 ERROR.)
23    FORMAT (//44X,18H**GENERATED DATA**/27X,4HCODE,4X,4HMEAN,6X,8HSTD
     1DEV.,5X,4HMIN.,7X,4HMAX.,5X,4HOBS./)
24    FORMAT (27X,I3,10X,18HNO VALUES RECORDED)
25    FORMAT (27X,I3,4F11.4,I7)
26    FORMAT (/44X,23H**TIME GENERATED DATA**/27X,4HCODE,4X,4HMEAN,6X,8
     1STD.DEV.,5X,4HMIN.,7X,4HMAX.,3X,10HTOTAL TIME/)
27    FORMAT (27X,I3,5F11.4)
28    FORMAT (/37X,37H**GENERATED FREQUENCY DISTRIBUTIONS**/27X,4HCODE,
     10X,10HHISTOGRAMS/)
29    FORMAT (27X,I3,5X,11I4/35X,11I4/)
31    FORMAT (20X,20A4)
      END
```

```
      SUBROUTINE OTPUT (NSET)
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      RETURN
      END
```

```
      SUBROUTINE EVNTS(I,NSET)
C
C*****THIS SUBROUTINE SELECTS THE NEXT EVENT
C
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20), I
1     CONTINUE
      CALL VALV02(1)
      GO TO 100
2     CONTINUE
      CALL VALV02(2)
      GO TO 100
3     CONTINUE
      CALL VALV02(3)
      GO TO 100
4     CONTINUE
      CALL VALV02(4)
      GO TO 100
5     CONTINUE
      CALL VALV03(1)
      GO TO 100
6     CONTINUE
      CALL VALV03(2)
      GO TO 100
7     CONTINUE
      CALL VALV03(3)
      GO TO 100
8     CONTINUE
      CALL VALV03(4)
      GO TO 100
9     CONTINUE
      CALL VALV03(5)
      GO TO 100
10    CONTINUE
      CALL GEMCS(NSET)
      GO TO 100
11    CONTINUE
C
      GO TO 100
12    CONTINUE
      CALL VALV02(5)
      GO TO 100
13    CONTINUE
      CALL VALV02(6)
      GO TO 100
14    CONTINUE
C
```

```
      GO TO 100
15    CONTINUE
C
      GO TO 100
16    CONTINUE
C
      GO TO 100
17    CONTINUE
C
      GO TO 100
18    CONTINUE
C
      GO TO 100
19    CONTINUE
C
      GO TO 100
20    CONTINUE
C
      GO TO 100
100   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MODULE (NT,NSET)
      DIMENSION NSET(12,1)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
     1,24,25,26,27,28,29,30), NT
1     CONTINUE
      CALL VARI01
      GO TO 100
2     CONTINUE
      CALL NODEJ
      GO TO 100
3     CONTINUE
      CALL TANK01(1)
      GO TO 100
4     CONTINUE
      CALL RECD01(NSET)
      GO TO 100
5     CONTINUE
      CALL VLBL (NSET)
      GO TO 100
6     CONTINUE
      CALL PRGP01(NSET)
      GO TO 100
7     CONTINUE
      CALL TANK02(NSET)
      GO TO 100
8     CONTINUE
      CALL SCHED01(NSET)
      GO TO 100
9     CONTINUE
C
      GO TO 100
10    CONTINUE
C
      GO TO 100
11    CONTINUE
C
      GO TO 100
12    CONTINUE
C
      GO TO 100
13    CONTINUE
C
      GO TO 100
14    CONTINUE
C
      GO TO 100
15    CONTINUE
C
      GO TO 100
16    CONTINUE
C
      GO TO 100
17    CONTINUE
C
      GO TO 100
```

```
18    CONTINUE
C
      GO TO 100
19    CONTINUE
C
      GO TO 100
20    CONTINUE
C
      GO TO 100
21    CONTINUE
C
      GO TO 100
22    CONTINUE
C
      GO TO 100
23    CONTINUE
C
      GO TO 100
24    CONTINUE
C
      GO TO 100
25    CONTINUE
C
      GO TO 100
26    CONTINUE
C
      GO TO 100
27    CONTINUE
C
      GO TO 100
28    CONTINUE
C
      GO TO 100
29    CONTINUE
C
      GO TO 100
30    CONTINUE
C
      GO TO 100
100   CONTINUE
      RETURN
      END
```

APPENDIX 2

PROGRAM LISTINGS AND DESCRIPTION OF THE MODEL IN CASE STUDY NO. 1

Program DESS

This "main" routine is the shortest, but performs three very important functions. First, it limits the number of columns available for the array NSET to 30. All the subroutines (system as well as non-system) having access to NSET via their argument list are compiled with NSET dimensioned for 12 rows and 1 column. Owing to the method of storage used in the computer core-memory involving aligning all columns of an array into one single-file vector, the next column beginning from where the last ends, access to NSET by the sub-routines is accomplished simply by passing on the starting location of the array in the first twelve elements (of the first column) via the argument. Since the information pertaining to any one subroutine is relocatable in the core, NSET can be assigned all available storage elsewhere in the core-memory not required by other portions of the simulation program.

The next function performed in the main routine is to start off the simulation run by calling GASP (NSET). When the simulation experiment is completed, and if it is not terminated before control is returned to the main routine, execution will be stopped with the command STOP 1.

```
      PROGRAM DESS(INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=PUN
     1CH)
C
C*****THIS MAIN PROGRAMME DETERMINES THE NUMBER OF COLUMNS IN NSET.
C
      DIMENSION NSET(12,30)
      CALL GASP (NSET)
      STOP 1
      END
```

Subroutine OPERAØ1

This subroutine simulates the controlling elements in the operations of the precipitation circuit. It sets the number of "special" precipitators and the number of "regular" precipitators using information on the present seed charge available and information on past operating conditions. (XØ represents the amount of fine seed presently available; YØ represents the amount of coarse seed available; RSM1 represents the number of "special" precipitators used previously; RRM1 represents the number of "regular" precipitators used previously; XM1 is the fine seed charge available; and YM1 is the coarse seed charge available, all at the previous time instance.) The present number of precipitators is calculated using the equations:

RSØ = XØ * RSM1/XM1 ("specials")

RRØ = YØ * RRM1/YM1 ("regulars")

The above equations were chosen arbitrarily, but it shows that if the amount of fine seed has increased, the number of "special" precipitators would be set proportionately larger. Since "specials" tend to use up more fine seed material, the overall effect should tend to stabilize the process. The total number of precipitators used at any given time is arbitrarily fixed at 60. Therefore, the values of RSØ and RRØ are "normalised" to add up to that number. These are then converted into the nearest integral values and stored in the information stream vector, SO, which allows the system to communicate between modules.

```
      SUBROUTINE OPERA01
C
C*****THIS SUBROUTINE SIMULATES THE DECISION-MAKING OPERATIONS
C
C*****SYSTEM COMMON CARDS
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,ISS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      X0=SI(1,6)
      XM1=SI(1,7)
      RSM1=SI(1,8)
      Y0=SI(2,6)
      YM1=SI(2,7)
      RRM1=SI(2,8)
      RS0=X0*RSM1/XM1
      RR0=Y0*RRM1/YM1
      EN(4)=60.0/(RS0+RR0)
      RS0=RS0*EN(4)
      RR0=RR0*EN(4)
      SO(1,6)=NEN=RS0+0.5
      SO(1,7)=NEN=RR0+0.5
      RETURN
      END
```

Subroutine SCHEDØ1 (NSET)

This subroutine creates new entries which it stores in the event-file. Thus, the first event it scheduled occurs at time 124.0 hours. The subroutine has been included here to illustrate the ability of the DESS executive to internally cause any event to occur at any particular time.

```
      SUBROUTINE SCHED01 (NSET)
C
C*****THIS SUBROUTINE SCHEDULES FUTURE EVENTS
C
C*****SYSTEM COMMON CARDS
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
C
      DIMENSION NSET(12,1)
      ATRIB(1)=TNOW+124.0
      ATRIB(2)=1.0
      CALL FTLEM (1,NSET)
      RETURN
      END
```

Subroutine VARIAØ1

If MODE is zero or negative, this subroutine behaves as module no. 2 on the information flow diagram (Appendix 3b) and imparts an "uncertainty" to the number of "special" precipitators and to the number of "regular" precipitators as set in OPERAØ1. The set value is taken as the mean of a normally distributed variate; the maximum and minimum limits are set at 175% and 25% of the mean, respectively, and the standard deviation is 25% of the mean value. The numbers returned are samplings from the distribution. These are transferred to the output stream vector after having been "normalised" to a sum of 60.

If MODE is any positive non-zero integer, the subroutine behaves as module no. 3 in the information flow diagram. In this case, it adds a variance to the liquor flowrate Z, and the fine and coarse seed charge flowrates, X and Y, respectively. The maximum absolute variation is set at 10% of the mean value and the standard deviation is one-third of that. Again, values returned are samplings from normal distributions.

```
      SUBROUTINE VARIA01
C
C*****THIS SUBROUTINE IMPARTS A VARIANCE TO THE STREAM VARIABLES.
C
C*****SYSTEM COMMON CARDS
C
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      MODE=EN(16)+0.001
      IF (MODE) 1,1,2
1     CONTINUE
      RNS=SI(1,6)
      RNR=SI(1,7)
      PARAM(2,1)=RNS
      PARAM(2,2)=1.75*RNS
      PARAM(2,3)=0.25*RNS
      PARAM(2,4)=0.25*RNS
      PARAM(3,1)=RNR
      PARAM(3,2)=1.75*RNR
      PARAM(2,3)=0.25*RNR
      PARAM(3,4)=0.25*RNR
      SO(1,6)=VS=RNORM(1)
      RNS=RNORM(2)
      RNR=RNORM(3)
      EN(4)=60.0/(RNS+RNR)
      SO(1,7)=NSS=RNS*EN(4)+0.5
      SO(1,8)=NRR=RNR*EN(4)+0.5
      GO TO 3
2     X=SI(1,6)
      Y=SI(1,7)
      Z=SI(1,8)
      PARAM(4,1)=X
      PARAM(4,2)=1.1*X
      PARAM(4,3)=0.9*X
      PARAM(4,4)=0.0333*X
      PARAM(5,1)=Y
      PARAM(5,2)=1.1*Y
      PARAM(5,3)=0.9*Y
      PARAM(5,4)=0.0333*Y
      PARAM(6,1)=Z
      PARAM(6,2)=1.1*Z
      PARAM(6,3)=0.9*Z
      PARAM(6,4)=0.0333*Z
      SO(1,6)=X=RNORM(4)
      SO(1,7)=Y=RNORM(5)
      SO(1,8)=Z=RNORM(6)
3     CONTINUE
      RETURN
      END
```

Subroutine PRECIØ1

This subroutine yields values for the variables describing average conditions in the precipitation circuit, given values for some of the other normally known or set variables pertaining to the system. Consider the equations relating these variables as outlined in the Simulation Study Report of 1969[2].

(1) $VSEEDS * NS = X * THETAS$

(2) $VSEEDR * NR = Y * THETAR$

(3) $\frac{VFILTS * NS}{THETAS} + \frac{VFILTR * NR}{THETAR} = Z$

(4) $VFILTS + VSEEDS = VS$

(5) $VFILTR + VSEEDR = VR$

(6) $THETAS = THETAR$

(7) $VS = VR$

(8) $NS + NR = NT$

(9) $\frac{NR}{NS} = Q$

The notation used is as follows:

X = fine seed flowrate ($ft.^3/hr.$)

Y = coarse seed flowrate ($ft.^3/hr.$)

Z = flowrate of filtrate from digestion circuit ($ft.^3/hr.$)

NS = number of "special" precipitators

NR = number of "regular" precipitators

VS = total volume in a "special" precipitator ($ft.^3$)

VR = total volume in a "regular" precipitator ($ft.^3$)

THETAS = residence time in "specials" (hrs.)

THETAR = residence time in "regulars" (hrs.)

VFILTS = volume of filtrate charge in "specials" ($ft.^3$)

VFILTR = volume of filtrate charge in "regulars" ($ft.^3$)

VSEEDS = volume of seed charge in "specials" including dilution liquors ($ft.^3$)

VSEEDR = volume of seed charge in "regulars" including dilution liquors ($ft.^3$)

NT = total number of precipitators

Q = ratio of "regulars" to "specials"

With nine equations and fifteen variables, at least six variables must be known before a complete solution can be obtained. However, not all the variables are independent of each other. The six mutually independent variables that are set in the model and therefore determine the values of the others are X, Y, Z, VS, NS, and NR. (Instead of the last three chosen, it would be the same with choosing VR, NT and Q or VR, NT and NR, etc., of course.) With six known variables, the set of equations can then simply be solved. From equations (7), (8) and (9), values may be obtained for Q, NR and VR, given values for X, Y, Z, VS, NS and NR. Let,

THETAS = THETAR = θ, and VR = VS = V.

From equations (1) and (3),

$$\frac{\text{VSEEDS} * \text{NS}}{X} = \frac{(\text{VFILTS} * \text{NS} + \text{VFILTR} * \text{NR})}{Z} = \theta.$$

Substituting VSEEDS from equation (4),

$$Z * NS * (V-VFILTS) = X * (VFILTS * NS + VFILTR * NR) \quad (10)$$

From equations (1) and (2), eliminating θ,

$$\frac{VSEEDS * NS}{X} = \frac{VSEEDR * NR}{Y} = \theta$$

Therefore,

$$\frac{VSEEDS}{VSEEDR} = \frac{NR}{NS} * \frac{X}{Y} \quad (11)$$

Similarly, from equations (4) and (5),

$$\frac{VSEEDS}{VSEEDR} = \frac{V - VFILTS}{V - VFILTR} \quad (12)$$

Hence, from equations (11) and (12),

$$\frac{V - VFILTS}{V - VFILTR} = \frac{NR}{NS} * \frac{X}{Y}$$

$$V - VFILTS = (V - VFILTR) * \frac{NR}{NS} * \frac{X}{Y}$$

$$VFILTS = V - \frac{X}{Y} * \frac{NR}{NS} * (V - VFILTR)$$

Substituting VFILTS in equation (10),

$$Z * NS\ (V - V + \frac{X}{Y} * \frac{NR}{NS} * [V - VFILTR]) =$$

$$X * (NS * [V - \frac{X}{Y} * \frac{NR}{NS} * (V - VFILTR)] + NR * VFILTR)$$

$$\frac{Z * X}{Y} * NR * [V - VFILTR] = X * [NS * V - \frac{X}{Y} * NR * (V - VFILTR) + NR * VFILTR]$$

$$[V - VFILTR] * NR * \frac{X}{Y} * [Z + X] = X * (NS * V + NR * VFILTR)$$

$$[V - VFILTR] * NR * [X + Z] = Y * NS * V + Y * NR * VFILTR$$

$$VFILTR * NR * [X + Y + Z] = [X + Z] * NR * V - Y * NS * V$$

Therefore,

$$VFILTR = V * [(X + Z) - Y * (\frac{NS}{NR})]/[X + Y + Z]$$

By symmetry, it can be shown that

$$VFILTS = V * [(Y + Z) - X * (\frac{NR}{NS})]/[X + Y + Z]$$

Thus VSEEDS and VSEEDR can be computed:

$$VSEEDS = V - VFILTS$$

$$VSEEDR = V - VFILTR$$

Also, THETAS = THETAR = θ

$$= \frac{NS}{X} * VSEEDS$$

$$= \frac{NS}{X} * (V - VFILTS)$$

To check the solution, THETAS and THETAR are computed again using,

$$THETAS = THETAR = \frac{NR}{Y} * VSEEDR$$

$$= \frac{NR}{Y} * (V - VFILTR)$$

In the subroutine, RT, RS, and RR are used in lieu of NT, NS and NR, respectively, in order to avoid integer arithmetic. X2 stores the value for the fine-seed charge available at a future time. It is calculated using an arbitrarily assumed equation that predicts the larger amount of fine seed that will be available if the number of "regular" precipitators is greater than the number of "special" precipitators used in the present operation. The amount of coarse seed available is larger or smaller according to whether the number of "specials" is more or less than the number of "regulars". The incremental change in either case is determined roughly in order of magnitude by the factors FACT1 and FACT2. The flowrate of seed charge available in the future, and which is associated with the present production are stored in X2 and Y2, for fine and coarse seed, respectively. RS and RR are the number of "special" and "regular" precipitators actually used in the operations at the "present" time instance.

```
      SUBROUTINE PRECI01
C
C*****THIS SUBROUTINE SOLVES THE PRECIPITATOR EQUATIONS GIVEN
C*****VALUES FOR CERTAIN VARIABLES.
C
C*****SYSTEM COMMON CARDS
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      FACT1=EN(16)
      FACT2=EN(17)
      X=SI(1,6)
      Y=SI(1,7)
      Z=SI(1,8)
      VS=VR=V=SI(2,6)
      RS=SI(2,7)
      RR=SI(2,8)
      VFILTS=V*(Z+Y-X*RR/RS)/(X+Y+Z)
      VFILTR=V*(Z+X-Y*RS/RR)/(X+Y+Z)
      VSEEDS=V-VFILTS
      VSEEDR=V-VFILTR
      THETAS=THETAR=(V-VFILTS)*RS/X
      CHECK=(V-VFILTR)*RR/Y
      IF (ABS(CHECK-THETAS).GT.0.0001) WRITE (6,2) CHECK
      RT=RS+RR
      RS=60.0*RS/RT
      RR=60.0*RR/RT
      RT=RS+RR
      Q=RR/RS
      X2=X+X*FACT1*(RR-RS)/RT
      Y2=Y+Y*FACT2*(RS-RR)/RT
      L1=RS+1.0E-10
      L2=RR+1.0E-10
      WRITE (6,3) VFILTS,VFILTR,VSEEDS,VSEEDR,THETAS,THETAR,RT
      WRITE (6,4) Q,VS,VR,X2,Y2,L1,L2
      DO 1 K=6,10
      SO(1,K)=0.0
1     CONTINUE
      SO(2,6)=X2
      SO(2,7)=Y2
      SO(2,8)=RS
      SO(2,9)=RR
      RETURN
C
2     FORMAT (* CHECK THETAS *,F12.4)
3     FORMAT (1H0,100(1H-),//,* VOL. OF LIQUOR IN TYPE S REACTORS = *,F1
     112.4,* CU. FT. *,/,* VOL. OF LIQUOR IN TYPE R REACTORS = *,F12.4,*
     2CU. FT. *,/,* VOL. OF SEED CHARGE IN TYPE S REACTORS = *,F12.4,* C
     3U. FT. *,/,* VOL. OF SEED CHARGE IN TYPE R REACTORS = *,F12.4,* CU
     4. FT. *,/,* RESIDENCE TIME IN TYPE S REACTORS = *,F12.4,* HOURS *,
     5/,* RESIDENCE TIME IN TYPE R REACTORS = *,F12.4,* HOURS *,/,* TOTA
     6L NUMBER OF REACTORS = *,F5.0)
4     FORMAT (1H ,*RATIO OF TYPE R TO TYPE S = *,F12.4,/,* TOT. VOL. FIL
```

```
1LED IN A TYPE S REACTOR = *,F12.4,* CU. FT. *,/,* TOT. VOL. FILLED
2 IN A TYPE R REACTOR = *,F12.4,* CU. FT. *,/,* EXPECTED FINE SEED
3FLOWRATE IN FUTURE = *,F12.4,* CFM *,/,* EXPECTED COARSE SEED FLOW
4RATE IN FUTURE = *,F12.4,* CFM *,/,* NUMBER OF TYPE S REACTORS USE
5D = *,I3,/,* NUMBER OF TYPE R REACTORS USED = *,I3,/,100(1H-)/)
 END
```

Subroutine QUEØ1 (NSET)

This subroutine retrieves the next entry from each of the two queues representing the coarse and fine seed charge available to the precipitation circuit at the "present" time. The values are equal to the flowrate of seed charge and are ranked in the queues according to the times of production of the seed material associated with the particular flowrate. (To explain it in another way, the seed that is produced at time, t, will cause a certain flowrate of seed charge, s, to enter the precipitation circuit at time ($t + \Delta t$). Therefore, s is associated with t and is inserted into the file for all s values and ranked according to the value of t). The flowrate value associated with the lowest value for time of production is withdrawn before all others. This value is then transferred to the appropriate element of the SN vector, so that "input stream" information is available for the next series of sequential computation within the GEMCS framework.

When the subroutine is first called, the sixth and eighth elements in streams 1 and 8 (refer to the information flow diagram in Appendix 3b) are stored in variables X, XNS, Y and YNR. These values represent the fine seed flowrate, number of "specials", coarse seed flowrate, and number of "regulars", respectively, associated with a previous time instance. Therefore, when execution enters QUEØ1 again, subsequently, these are transferred to the appropriate elements in steams 1 and 8 and are replaced with the present flowrate values to be used for the next time step. The past values for the number of precipitators in the variables XNS, YNR are also transferred into streams 1 and 8 and are replaced with the last set of values for the actual number of precipitators used in the process over the previous time period. (These are obtained from the

eighth and ninth elements of stream 7.) Hence, X, Y, XNS and YNR all yield values that are associated with the state of the process at the previous time-step. The flow diagram in Appendix 3a should illustrate this more clearly.

```
      SUBROUTINE QUE01 (NSET)
C
C*****THIS SUBROUTINE OBTAINS THE SEED-CHARGE INPUT VALUES
C*****AND THE NUMBER OF PRECIPITATORS AT A PREVIOUS TIME FROM THE
C
C*****SYSTEM COMMON CARDS
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      DIMENSION NSET(12,1)
      IF (TNOW.LT.0.1) GO TO 1
      IF (MFE(2).LE.0) WRITE (6,3) MFE(2)
      CALL RMOVE (MFE(2),2,NSET)
      CALL STREAM (4.0)
      SN(IS,6)=ATRIB(2)
      CALL STREAM (1.0)
      SN(IS,6)=ATRIB(2)
      SN(IS,7)=X
      X=SN(IS,6)
      SN(IS,8)=XNS
      CALL STREAM (7.0)
      XNS=SN(IS,8)
      IF (MFE(3).LE.0) WRITE (6,4) MFE(3)
      CALL RMOVE (MFE(3),3,NSET)
      CALL STREAM (4.0)
      SN(IS,7)=ATRIB(2)
      CALL STREAM (8.0)
      SN(IS,6)=ATRIB(2)
      SN(IS,7)=Y
      Y=SN(IS,6)
      SN(IS,8)=YNR
      CALL STREAM (7.0)
      YNR=SN(IS,9)
      GO TO 2
1     CONTINUE
      CALL STREAM (1.0)
      X=SN(IS,6)
      XNS=SN(IS,8)
      CALL STREAM (8.0)
      Y=SN(IS,6)
      YNR=SN(IS,8)
2     CONTINUE
      RETURN
C
```

```
3       FORMAT (1H0,* MFE(2) = *,I3)
4       FORMAT (1H0,* MFE(3) = *,I3)
        END
```

Subroutine QUEØ2 (NSET)

This subroutine has access to NSET and the system variables. Its essential function is to transfer the values of the seed charge flowrate associated with the present seed production (but which is only available for future operations) into the fine seed queue (file 2) and the coarse seed queue (file 3) from the SN stream vector. The buffer vector ATRIB contains the flowrate value in the second element and the value of the "present" time (with which the queues are ranked) in the first element. The values are then stored in NSET by calling FILEM and become part of the appropriate queue.

```
      SUBROUTINE QUE02 (NSET)
C
C*****THIS SUBROUTINE STORES THE SEED-FLOW AVAILABLE IN THE FUTURE
C
C*****SYSTEM COMMON CARDS
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      DIMENSION NSET(12,1)
      CALL STREAM (7,0)
      ATRIB(1)=TNOW
      ATRIB(2)=SN(IS,6)
      CALL FILEM (2,NSET)
      ATRIB(1)=TNOW
      ATRIB(2)=SN(IS,7)
      CALL FILEM (3,NSET)
      RETURN
      END
```

Subroutine EVNTS (I, NSET)

This subroutine is called by the subroutine GASP (NSET) for selection of the next event to be simulated. The event code, I, is transferred in the argument. In this particular simulation run, the first five events scheduled in the data set have the code value of 1, and the sixth event, the code value 2. The first five events involve calling subroutine GEMCS (NSET) to handle the information stream flow used in the solution of the equations describing conditions in the precipitation circuit. The sixth event scheduled at time 101.0 hours consists of simply a command to print "The following events are scheduled during the simulation run". Events after that instance in simulated time are scheduled in an "endogenous" fashion, ie. they are planned to occur, internally, at a future time, during the execution of events scheduled originally in the data set (so-called "exogenous" events). Hence, the system is shown to be able to handle both "exogenous" (externally caused or pre-scheduled) events and "endogenous" events (internally caused, predicted, or triggered off by some relationship being satisfied at some point in time). In this example, the endogenous events are also assigned the event code of 1.

```
      SUBROUTINE EVNTS (I,NSET)
C
C*****THIS SUBROUTINE SELECTS THE NEXT EVENT.
C
C*****SYSTEM COMMON CARDS
C
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
C
      DIMENSION NSET(12,1)
      GO TO (1,2,3,4,5,6), I
1     CONTINUE
      WRITE (6,8) TNOW
      CALL GEMCS (NSET)
      GO TO 7
2     CONTINUE
      WRITE (6,9)
      GO TO 7
3     CONTINUE
C
      GO TO 7
4     CONTINUE
C
      GO TO 7
5     CONTINUE
C
      GO TO 7
6     CONTINUE
C
      GO TO 7
7     CONTINUE
      RETURN
C
8     FORMAT (///,*  STATE OF SIMULATED PROCESS AT TIME  *,F12.4,* HOURS
     1 *,///)
9     FORMAT (1H0,* THE FOLLOWING EVENTS ARE SCHEDULED DURING THE SIMULA
     1TION RUN  *)
      END
```

Subroutine MODULE (NT, NSET)

This is a user-supplied subroutine required in the GEMCS subsystem. It is called from the subroutine GEMCS (NSET) to select the appropriate subroutine module from the information network for calculation of output information given certain input data. The subroutines called by the subroutine MODULE essentially represent the nodes where information streams diverage or converge in an information flowchart representing the state of the simulated system at a discrete time period. The control of the information-handling and the manipulation of data in and out of each of these nodes in the flowchart is completely under the control of the GEMCS subsystem which includes the subroutine GEMCS (NSET), DLOAD1, STREAM(S) and DISKIO (IPNT, MM) besides MODULE (NT, NSET). It is noted here that in the GEMCS subsystem, communication with the GASP subsystem and access to the array, NSET, is only allowed in the subroutines GEMCS (NSET) and MODULE (NT, NSET).

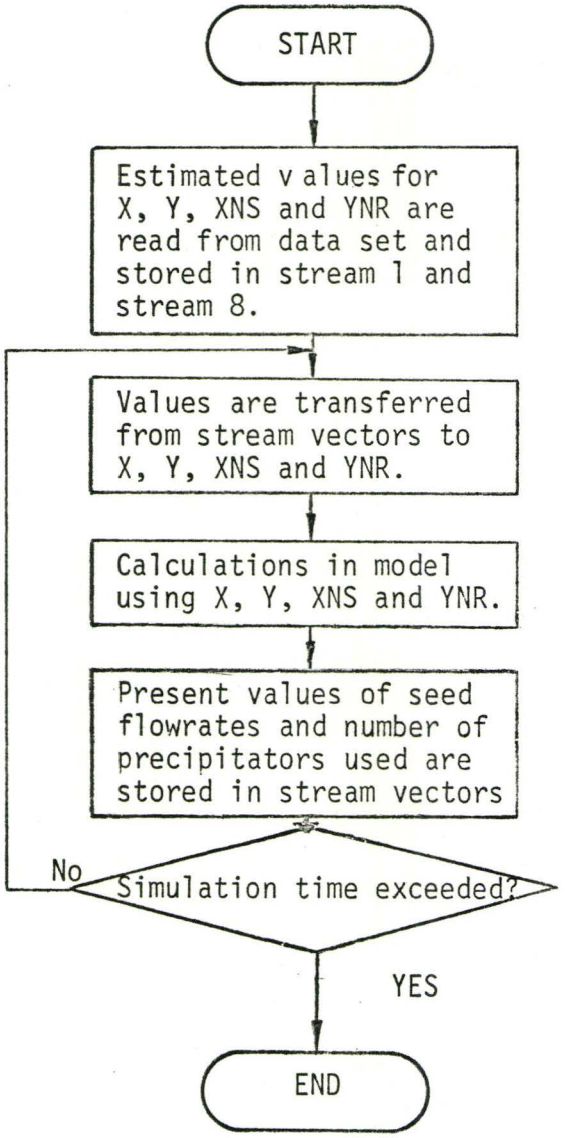
The dummy variable, NT, in the argument of MODULE carries the code number for the type of module to be called. The subroutine MODULE is the GEMCS analogous version of the subroutine EVNTS in the GASP subsystem and must be rewritten and supplied by the user for each different simulation experiment. However, whereas the subroutine EVNTS calls the next simulated event at the "present" simulated time, the subroutine MODULE calls the next GEMCS module to solve the information network, in a sequential manner until all elements of the information streams in the network have been evaluated or set; control is then transferred out of the GEMCS subsystem.

```
      SUBROUTINE MODULE (NT,NSET)
      DIMENSION NSET(12,1)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
     1,24,25,26,27,28,29,30), NT
1     CONTINUE
C
      GO TO 31
2     CONTINUE
C
      GO TO 31
3     CONTINUE
C
      GO TO 31
4     CONTINUE
C
      GO TO 31
5     CONTINUE
C
      GO TO 31
6     CONTINUE
C
      GO TO 31
7     CONTINUE
C
      GO TO 31
8     CONTINUE
C
      GO TO 31
9     CONTINUE
C
      GO TO 31
10    CONTINUE
      CALL PRECI01
      GO TO 31
11    CONTINUE
      CALL OPERA01
      GO TO 31
12    CONTINUE
      CALL VARIA01
      GO TO 31
13    CONTINUE
      CALL QUE01 (NSET)
      GO TO 31
14    CONTINUE
      CALL QUE02 (NSET)
      GO TO 31
15    CONTINUE
      CALL SCHED01 (NSET)
      GO TO 31
16    CONTINUE
C
      GO TO 31
17    CONTINUE
C
      GO TO 31
```

```
18    CONTINUE
C
      GO TO 31
19    CONTINUE
C
      GO TO 31
20    CONTINUE
C
      GO TO 31
21    CONTINUE
C
      GO TO 31
22    CONTINUE
C
      GO TO 31
23    CONTINUE
C
      GO TO 31
24    CONTINUE
C
      GO TO 31
25    CONTINUE
C
      GO TO 31
26    CONTINUE
C
      GO TO 31
27    CONTINUE
C
      GO TO 31
28    CONTINUE
C
      GO TO 31
29    CONTINUE
C
      GO TO 31
30    CONTINUE
C
      GO TO 31
31    CONTINUE
      RETURN
      END
```

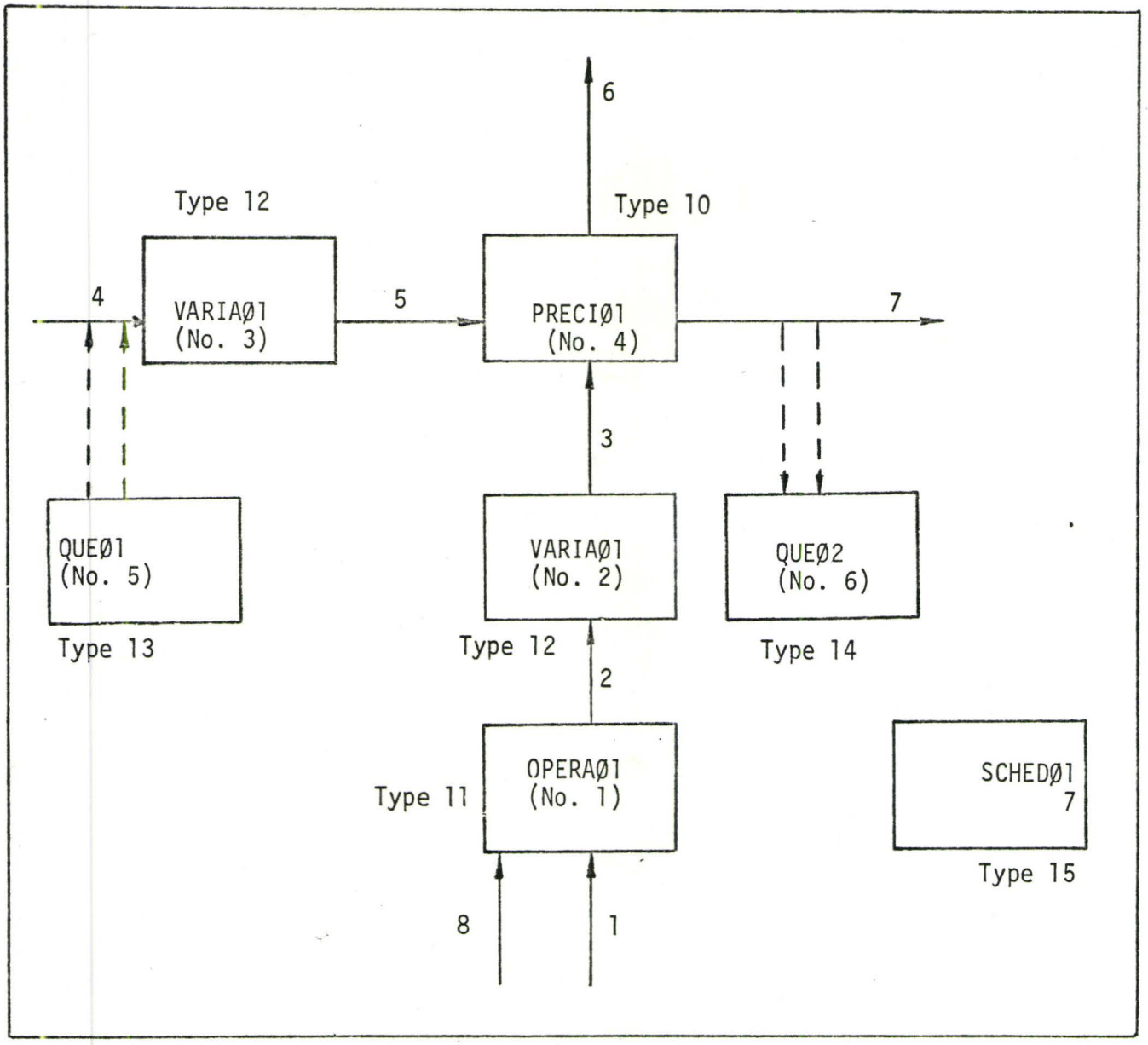
APPENDIX 3a

DIAGRAM SHOWING TRANSFER OF INFORMATION BETWEEN SYSTEM DATA POOL AND SUBROUTINES QUEØ1 AND QUEØ2



APPENDIX 3b

THE INFORMATION FLOW DIAGRAM OF SIMULATED SYSTEM



APPENDIX 4

DATA SET FOR CASE STUDY NO. 1

```
      6400 END OF RECORD
 TEST OF DESS (DISCRETE-EVENT SIMULATION SYSTEM)
1.0          11.0          25.0          1971.0        1.0
6.0          0.0           0.0           0.0           30.0
2.0          3.0           0.0           4.0           10000.0
1.0          1.0           1.0
1.0          1.0           1.0
37200.0      30000.0       44000.0       2300.0
20.0         5.0           35.0          5.0
40.0         25.0          55.0          5.0
6200.0       5600.0        6800.0        200.0
12000.0      11000.0       13000.0       400.0
48000.0      43000.0       53000.0       1000.0
1.0          0.0           0.0           1.0           0.0
400.0        66.0
-1.0
0.0          0.0
1.0
0.0          1.0
1.0
24.0         1.0
1.0
44.0         1.0
1.0
67.0         1.0
1.0
93.0         1.0
1.0
101.0        2.0
2.0
0.0          6200.0
3.0
0.0          12000.0
0.0
0.0          0.0
 DISCRETE-EVENT SIMULATION DATA-SET FOR FIRST EVENT AT TIME 0.0 HOURS
-------------------------------------------------------------------------
1.0          1.0           1.0
```

```
0.0         1.0         0.0         0.0         0.0
7.0         5.0
5.0         1.0         2.0         3.0         4.0
6.0         7.0
8.0
1.0         2.0         2.0         1.0         2.0
3.0         3.0         1.0
3.0
1.0         0.0         0.0         0.0         0.0
6200.0      6200.0      20.0
4.0         0.0         0.0         0.0         0.0
6200.0      12000.0     48000.0
8.0         0.0         0.0         0.0         0.0
12000.0     12000.0     40.0
7.0
1.0         11.0        15.0
2.0         1.0         8.0
1.0         2.0
2.0         12.0        16.0
1.0         2.0
1.0         3.0
0.0
3.0         12.0        16.0
1.0         4.0
1.0         5.0
1.0
4.0         10.0        17.0        0.0         0.09
2.0         5.0         3.0
2.0         6.0         7.0
0.1         0.1
5.0         13.0        15.0
0.0
0.0
6.0         14.0        15.0
0.0
0.0
```

```
7.0          15.0          15.0
0.0
0.0
'          END OF FILE
```

APPENDIX 5

PROGRAM LISTINGS AND DESCRIPTION OF THE MODEL IN CASE STUDY NO. 2

Program TST

This is the "main" program which sets the number of columns in NSET to 100. It calls subroutine GASP to start the simulation and terminates it when the run is complete.

```
      PROGRAM TST(INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT,TAPE7=PUNC
     1H)
C
C*****THIS MAIN PROGRAMME DETERMINES THE NUMBER OF COLUMNS IN NSET.
C
      DIMENSION NSET(12,100)
      CALL GASP (NSET)
      STOP 1
      END
```

Subroutine CLOCK (DAYS, HRS, ZMIN, TMINS)

A time value in minutes is supplied to the subroutine via the argument TMINS. This value is converted into units of days, hours and minutes and the information is returned to the calling subprogram via the remaining arguments.

```
      SUBROUTINE CLOCK (DAYS,HRS,ZMINS,TMINS)
C
C     THIS SUBROUTINE CONVERTS TMINS (IN MINUTES) TO DAYS, HOURS,
C     AND MINUTES
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      LHRS=(TMINS/60.0)+1.0E-10
      HRS=LHRS
      ZMINS=TMINS-(HRS*60.0)
      LDAYS=(HRS/24.0)+1.0E-10
      DAYS=LDAYS
      HRS=HRS-(DAYS*24.0)
      IF (KPRNT(7).EQ.1) WRITE (6,1) DAYS,HRS,ZMINS
      RETURN
C
1     FORMAT (1H ,F10.4,* DAYS *,F10.4,* HRS. *,F10.4,* MINS. *)
      END
```

Subroutine VARIØ1

The subroutine supplies a flowrate value of 800.0 ± 50.0 cfm. to stream no. 1 in the information flow diagram (shown in Appendix 6b). The variations are represented by random deviates from a uniform distribution. The third column of the system stream vectors, SN, SI and SO, is used consistently throughout this simulation to represent flowrate values. (The first column is used for storing the stream numbers.)

```
      SUBROUTINE VARI01
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      CALL STREAM (1.0)
      SN(IS,3)=UNFRM(-50.0,50.0)+800.0
      RETURN
      END
```

Subroutine NODEJ

The subroutine acts as a node in the information flow diagram (Appendix 6b) where stream 1.0 is split into streams 3.0 and 4.0.

```
      SUBROUTINE NODEJ
C
C     A SIMPLE JUNCTION MODULE (WHERE STREAMS CONVERGE AND DIVERGE)
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      SO(1,3)=0.5*SI(1,3)
      SO(2,3)=SO(1,3)
      RETURN
      END
```

Subroutine SCHEDØ1 (NSET)

The subroutine prints the time during the simulation run. It also schedules the next event when the GEMCS subsystem will be called again to handle the computation of the activities in the process network. The time interval in between each scheduled call of GEMCS is stored in EN(4). The time interval in between each printing of the simulation data is stored in EN(5).

```
      SUBROUTINE SCHED01 (NSET)
C
C*****THIS SUBROUTINE SCHEDULES FUTURE EVENTS
C
      DIMENSION NSET(12,1)
C
C*****GEMCS COMMON CARDS
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
C*****GASP COMMON CARDS
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
C
C     EN(4) --- TIME INTERVAL (MINS.) IN BETWEEN EACH CALL OF GEMCS
C     EN(5) --- TIME INTERVAL (MINS.) IN BETWEEN EACH PRINTING
C
      IF (TNOW.GT.0.1) GO TO 1
      LSD=KPRNT(7)
      ANOW=TNOW-0.01
      TIME=TNOW-0.01
1     CONTINUE
      IF (LSD.EQ.0) GO TO 2
      IF (TNOW.LT.0.1) GO TO 2
      KPRNT(7)=0
      IF ((TNOW-TIME).LT.EN(5)) GO TO 2
      KPRNT(7)=1
      TIME=TNOW-0.01
2     CONTINUE
      IF ((TNOW-TFIN).GE.-EN(4).AND.LSD.NE.0) KPRNT(7)=1
      IF (KPRNT(7).NE.1) GO TO 3
      IF (TNOW.GE.0.1) WRITE (6,5)
      IF (TNOW.LT.0.1) WRITE (6,6)
      CALL CLOCK (DAYS,HRS,ZMINS,TNOW)
3     CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,7)
C
C     SCHEDULING ANOTHER GEMCS CALL IN 15.0 MINUTES TIME
C
      IF (TNOW.LT.0.1) GO TO 4
      IF ((TNOW-ANOW).LT.EN(4)) RETURN
4     ATRIB(1)=TNOW+EN(4)
      ATRIB(2)=10.0
      CALL FILEM (1,NSET)
      ANOW=TNOW-0.01
      RETURN
C
5     FORMAT (1H ,* THE TIME IS *)
6     FORMAT (1H1,* THE TIME IS *)
7     FORMAT (1H0,100(1H-),/)
      END
```

Subroutine POLYØ1 (M, N, L)

The subroutine yields information on the type of precipitator to be filled next. If L is 0, only a check on the type to be filled is made; no actual filling is assumed. If L is 1, subsequent filling is assumed to be carried out, and information stored in the subroutine is updated. M supplies the policy code: the 1/1 policy (1 "regular"/1 "special") has code 1, and the 2/1 policy (2 "regulars"/1 "special") has code 2. The information returned is stored in N. For a "regular", N is 1, and for a "special", N is 2.

```
      SUBROUTINE POLY01 (M,N,L)
C
C     SUBROUTINE DETERMINES THE TYPE OF THE NEXT PRECIPITATOR TO BE
C     FILLED ACCORDING TO OPERATING POLICY M
C     IF M = 1, POLICY IS 1/1 (I.E. 1 REGULAR/1 SPECIAL)
C     IF M = 2, POLICY IS 2/1 (I.E. 2 REGULARS/1 SPECIAL)
C     FOR A REGULAR, N = 1
C     FOR A SPECIAL, N = 2
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      DIMENSION LINE(4)
C
C     IF L = 0, ONLY A CHECK ON THE TYPE OF PRECIPITATOR TO BE FILLED
C     NEXT IS MADE, NO ACTUAL FILLING IS ASSUMED
C     IF L = 1, SUBSEQUENT FILLING IS ASSUMED TO BE CARRIED OUT
C
      IF (L.LT.0.OR.L.GT.1) GO TO 15
      IF (M.LT.1.OR.M.GT.2) GO TO 2
      IF (KPRNT(7).NE.1) GO TO 1
      IF (LINE(2).EQ.0.OR.LINE(4).EQ.0) GO TO 1
      IF (L.NE.1) GO TO 1
      WRITE (6,17) M
1     CONTINUE
      GO TO (3,9), M
2     CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,16) M
      M=1
      GO TO (3,9), M
3     IF (LINE(2).EQ.0) GO TO 4
      LINE(1)=0
      LINE(2)=0
      IF (KPRNT(7).NE.1) GO TO 4
      IF (L.EQ.1) WRITE (6,18)
4     CONTINUE
      IF (LINE(1).EQ.0.OR.LINE(1).EQ.1) GO TO 5
      IF (LINE(1).EQ.2) GO TO 7
      IF (KPRNT(7).EQ.1) WRITE (6,19)
      RETURN
5     N=1
      IF (L.EQ.0) GO TO 6
      LINE(1)=2
6     CONTINUE
      IF (KPRNT(7).NE.1) RETURN
      IF (L.EQ.1) WRITE (6,20)
      RETURN
7     N=2
      IF (L.EQ.0) GO TO 8
      LINE(1)=1
8     CONTINUE
      IF (KPRNT(7).NE.1) RETURN
      IF (L.EQ.1) WRITE (6,21)
      RETURN
9     IF (LINE(4).EQ.0) GO TO 10
      LINE(3)=1
      LINE(4)=0
```

```
      IF (KPRNT(7).NE.1) GO TO 10
      IF (L.EQ.1) WRITE (6,22)
10    CONTINUE
      IF (LINE(3).LT.4) GO TO 11
      IF (KPRNT(7).EQ.1) WRITE (6,23)
      RETURN
11    IF (LINE(3).EQ.3) GO TO 13
      N=1
      IF (L.EQ.0) GO TO 12
      LINE(3)=LINE(3)+1
12    CONTINUE
      IF (KPRNT(7).NE.1) RETURN
      IF (L.EQ.1) WRITE (6,20)
      RETURN
13    N=2
      IF (L.EQ.0) GO TO 14
      LINE(3)=1
14    CONTINUE
      IF (KPRNT(7).NE.1) RETURN
      IF (L.EQ.1) WRITE (6,21)
      RETURN
15    IF (KPRNT(7).NE.1) RETURN
      WRITE (6,24)
      RETURN
C
16    FORMAT (1H0,* ERROR IN POLY01 -- M = *,I5/*  M IS RESET TO 1 *)
17    FORMAT (1H ,* FILLING POLICY IS *,I2)
18    FORMAT (1H ,* POLICY 1/1 IS IN OPERATION *)
19    FORMAT (1H0,* ERROR EXIT 1 *)
20    FORMAT (1H ,* FILLING AS REGULAR *)
21    FORMAT (1H ,* FILLING AS SPECIAL *)
22    FORMAT (1H ,* POLICY 2/1 IS IN OPERATION *)
23    FORMAT (1H0,* ERROR EXIT 2 *)
24    FORMAT (1H0,* ERROR EXIT 3 - POLY01 - *)
      END
```

Subroutine VALVØ2 (J)

The subroutine controls the availability of the precipitator inlets for seed changing and liquor filling. It is first called at the beginning of the simulation to initialize some non-DESS variables used in the model. All inlets and discharge outlets to the precipitator circuit are set at the "available" condition.

```
      SUBROUTINE VALV02 (J)
C
C     VALV02 CONTROLS THE AVAILABILITY OF PRECIPITATOR INLET VALVES
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /PREC/ AVAIL(3),MAVA(2,1)
      COMMON /TANK/ TTNOW(2),VOLB(2)
      COMMON /DIS/ JDIS(5),TTTN(5),LLCOU,VOLD
      COMMON /PREC2/ TCS,TFS
      COMMON /SEED/ COARS,FINES
      COMMON /PRG/ JDD,JSD
C
C     AVAIL(M) --- SETTINGS OF PRECIPITATOR INLET VALVES (M = 3)
C     JDIS(I) --- AVAILABILITY CODE OF DISCHARGE OUTLET OF GROUP I
C     TTTN(I) --- TIME AT WHICH OUTLET BECOMES AVAILABLE
C     TTNOW(I) --- TIME AT WHICH OUTLET VALVE OF TANK I IS CLOSED
C     NTANK --- TANK NUMBER
C     TCS --- TIME AT WHICH COARSE SEED INLET BECOMES AVAILABLE
C     TFS --- TIME AT WHICH FINE SEED INLET BECOMES AVAILABLE
C
      GO TO (1,2,4,5,8,9), J
1     CONTINUE
      GO TO 10
C
C     CONTROLS AVAILABILITY OF 2 COARSE SEED INLET VALVES
C     THERE ARE ONLY 2 COARSE SEED INLETS AVAILABLE, THEREFORE
C     NO MORE THAN 2 MAY BE AVAILABLE AT ANY ONE TIME
C     AVAIL(2) = 0.0 --- BOTH INLETS AVAILABLE
C     AVAIL(2) = 1.0 --- ONLY ONE INLET AVAILABLE
C     AVAIL(2) = 2.0 --- NO COARSE SEED INLET IS AVAILABLE
C
2     CONTINUE
      IF (AVAIL(2).LE.0.8.AND.KPRNT(7).EQ.1) WRITE (6,11) AVAIL(2)
      IF (AVAIL(2).GT.0.8) AVAIL(2)=AVAIL(2)-1.0
      IF (KPRNT(7).NE.1) GO TO 3
      WRITE (6,12)
      CALL CLOCK (DAYS,HRS,ZMINS,TNOW)
3     CONTINUE
      JSD=JSD-1
      IF (AVAIL(2).LT.1.11.AND.AVAIL(2).GT.0.99) TCS=TNOW
      NAV=AVAIL(2)+0.01
      IF (KPRNT(7).NE.1) GO TO 10
      IF (NAV.EQ.0) WRITE (6,13)
      IF (NAV.EQ.1) WRITE (6,14)
      IF (NAV.EQ.2) WRITE (6,15)
      WRITE (6,16)
      GO TO 10
C
C     CONTROLS AVAILABILITY OF FINE SEED INLET VALVE
C     WHEN SUBROUTINE IS CALLED, FINE SEED CHARGING IS COMPLETED,
C     AND THE VALVE IS THEN SHUT --- THEREFORE, SET AVAIL(3) = 0.0
C
4     CONTINUE
      JSD=JSD-1
```

```
      TFS=TNOW
      AVAIL(3)=0.0
      IF (KPRNT(7).NE.1) GO TO 10
      WRITE (6,17)
      CALL CLOCK (DAYS,HRS,ZMINS,TNOW)
      WRITE (6,16)
      GO TO 10
C
C     SUBROUTINE IS FIRST CALLED TO CLOSE ALL VALVES INITIALLY
C     OTHER VARIABLES ARE ALSO INITIALISED
C
5     CONTINUE
C
C     VALUES MUST BE SUPPLIED TO TTNOW(1), TTNOW(2), TCS, TFS,
C     AVAIL(I), I = 1,3, TTTN(I), I = 1,5, JDIS(I), I = 1,5
C
      TTNOW(1)=0.0
      TTNOW(2)=0.0
      TCS=0.0
      TFS=0.0
C
C     MAKE ALL INLETS INITIALLY AVAILABLE
C
      JSD=0
      JDD=0
      MAVA(1,1)=0
      MAVA(2,1)=0
      DO 6 I=1,3
      AVAIL(I)=0.0
6     CONTINUE
      COARS=0.0
      FINES=0.0
C
C     MAKE ALL DISCHARGE OUTLETS AVAILABLE
C
      DO 7 I=1,5
      TTTN(I)=0.0
      JDIS(I)=0
7     CONTINUE
      LLCOU=0
      IF (KPRNT(7).NE.1) GO TO 10
      WRITE (6,16)
      WRITE (6,18)
      WRITE (6,16)
      GO TO 10
8     CONTINUE
9     CONTINUE
10    CONTINUE
      RETURN
C
11    FORMAT (1H0,* ERROR IN AVAIL(2) - VALV02 - AVAIL(2) = *,F12.4)
12    FORMAT (1H0,* COARSE SEED INLET IS NOW AVAILABLE AT TIME - *)
13    FORMAT (1H0,* TWO COARSE SEED INLETS ARE NOW AVAILABLE *)
14    FORMAT (1H0,* ONE COARSE SEED INLET IS NOW AVAILABLE *)
15    FORMAT (1H0,* ERROR IN AVAIL(2) AND NAV *)
```

```
16    FORMAT (1H0,100(1H-),/)
17    FORMAT (1H0,* FINE SEED INLET IS NOW AVAILABLE AT TIME - *)
18    FORMAT (1H ,* ALL VALVES ARE CLOSED AT START OF SIMULATION *)
      END
```

Subroutine TANKØ1 (K)

The subroutine is a model of the filling tanks in the precipitation circuit. There are two parts in the subroutine. The first consists of computing the new level height in the tank and storing it in the EEN vector with a call of the subroutine DISKIO. The second involves the setting of the flowrate of the liquor flows to the precipitators.

```
      SUBROUTINE TANK01 (K)
C
C     A SIMPLE FILLING TANK MODULE
C
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /PREC/ AVAIL(3),MAVA(2,1)
      COMMON /TANK/ TTNOW(2),VOLB(2)
      COMMON /TANK1/ HGT(2),TUT(2)
C
      DIMENSION FLOW(2)
C
C     AVAIL(M) --- SETTINGS OF PRECIPITATOR INLET VALVES (M = 3)
C     TTNOW(I) --- TIME AT WHICH OUTLET VALVE OF TANK I IS CLOSED
C     NTANK --- TANK NUMBER
C
C     EN(16) --- PREVIOUS HEIGHT OR INITIAL HEIGHT
C     EN(17) --- PREVIOUS TIME INSTANCE OR STARTING TIME INSTANCE
C     EN(18) --- FILLING TANK AREA
C     EN(19) --- NOT USED
C
      GO TO (1,2), K
1     CONTINUE
      SO(1,3)=FLOW(NE)
      H1=EN(16)+(SI(1,3)*(TNOW-EN(17))-VOLB(NE))/EN(18)
      EN(16)=H1
      EN(17)=TNOW
      HGT(NE)=H1
      CALL DISKIO (2,NE)
      IF (KPRNT(7).NE.1) RETURN
      WRITE (6,7) NE,H1
      CALL CLOCK (DAYS,HRS,ZMINS,TNOW)
      RETURN
2     CONTINUE
      H1=EN(16)
      NTA=EN(1)+0.01
      FLOW(NTA)=SI(1,3)
      IF (H1.GT.0.1) FLOW(NTA)=SI(1,3)+5.0*H1
      IF (H1.LT.0.0) TFIN=TNOW
      IF (MAVA(NTA,1).EQ.0) GO TO 3
      IF (MAVA(NTA,1).EQ.1) GO TO 4
      IF (MAVA(NTA,1).EQ.2) GO TO 5
3     CONTINUE
      TUT(NTA)=0.0
      FLOW(NTA)=0.0
      GO TO 6
4     CONTINUE
      TUT(NTA)=FLOW(NTA)
      GO TO 6
5     CONTINUE
      TUT(NTA)=FLOW(NTA)/2.0
```

```
6     RETURN
C
7     FORMAT (1H ,* L. HT. OF TK. *,I2,* = *,F10.4,* FT., AT *)
      END
```

Subroutine TANKØ2 (NSET)

The subroutine computes the level height in the hydrate agitator and stores it in the EEN vector. It also seaches through the precipitator files (files 2, 3 and 4) and reports the number of precipitators in each. (File 2 contains empty precipitators; file 3 contains precipitators waiting for seed charging, and file 4 contains precipitators "in circulation".) The number of precipitators that are being filled or drawn-off at the same time point are similarly recorded. The "total free height" measuring the free volume in the circuit available for material charging is computed in the final part of the subprogram with the equation:

$$T.F.H. = N \times 59.7 - \sum_{i=1}^{N} H_i$$

where, T.F.H. = Total free height,

N = Number of precipitators in the circuit,

H_i = Present height of the i^{th} precipitator.

If the value for T.F.H. is found to exceed 350 ft. and if the level in the hydrate agitator is below 110 ft., the hydrate agitator underflow will be decreased by 10% of its present value. If T.F.H. is below 200 ft., the underflow will be increased by the same amount.

```
      SUBROUTINE TANK02 (NSET)
C
C     A SIMPLE MODEL FOR THE HYDRATE AGITATOR
C
      DIMENSION NSET(12,1)
      DIMENSION NPREC(4)
C
C     GASP AND GEMCS COMMON CARDS
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      COMMON /TK2/ HHH
      COMMON /DIS/ JDIS(5),TTTN(5),LLCOU,VOLD
      COMMON /FH/ XHT(5)
      COMMON /PRG/ JDD,JSD
      COMMON /FILL/ XLIQ(3,8)
      COMMON /FH2/ FREEH
C
C     EN(16) --- OUTLET FLOW
C     EN(17) --- PAST TIME VALUE (STORAGE)
C     EN(18) --- INITIAL HEIGHT AND STORAGE FOR SUBSEQUENT VALUES
C                OF LIQUOR HEIGHT
C
      IF (TNOW.LT.0.1) TIMM=0.0
      SO(1,3)=EN(16)
      DUR=TNOW-EN(17)
      HHH=EN(18)+(VOLD-(SO(1,3)*DUR))/531.0
      EN(17)=TNOW
      EN(18)=HHH
      NTA=EN(1)+0.01
      CALL DISKIO (2,NTA)
      IF (KPRNT(7).NE.1) GO TO 1
      WRITE (6,12) HHH
      CALL CLOCK (DAYS,HRS,ZMINS,TNOW)
      IF (HHH.GT.125.4) WRITE (6,13)
      WRITE (6,11)
1     CONTINUE
      FREEH=0.0
      DO 4 I=2,4
      NPREC(I)=0
      NFREE=0
      MF=MFE(I)
      IF (MF.LE.0) GO TO 3
2     CALL CHECK (MF,I,NSET)
```

```
      NFREE=NFREE+1
      NPREC(I)=NFREE
      FREEH=FREEH+ATRIB(5)
      MF=NSET(MX,MF)
      IF (MF.EQ.7777) GO TO 3
      GO TO 2
3     CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,14) I,NPREC(I)
4     CONTINUE
      DO 5 J=1,5
      FREEH=FREEH+XHT(J)
5     CONTINUE
      DO 6 J=1,3
      FREEH=FREEH+XLIQ(J,5)
6     CONTINUE
      IF (KPRNT(7).NE.1) GO TO 7
      WRITE (6,15) JDD
      WRITE (6,16) LLCOU
7     CONTINUE
      NFREE=NPREC(2)+NPREC(3)+NPREC(4)+LLCOU+JDD
      X=NFREE
      FREEH=(X*59.7)-FREEH
      IF (KPRNT(7).NE.1) GO TO 8
      WRITE (6,17) NFREE
      WRITE (6,11)
8     CONTINUE
      TIMM=TIMM+DUR
      IF (TIMM.LT.120.0) GO TO 10
      IF (KPRNT(7).NE.1) GO TO 9
      WRITE (6,18) FREEH
      IF (FREEH.LT.200.0) WRITE (6,19)
      IF (FREEH.GT.350.0) WRITE (6,20)
      WRITE (6,11)
9     CONTINUE
      TIMM=0.0
      IF (FREEH.LT.200.0) EN(16)=EN(16)+0.1*EN(16)
      IF (FREEH.GT.350.0.AND.HHH.LT.110.0) EN(16)=EN(16)-0.1*EN(16)
10    CONTINUE
      IF (HHH.GE.115.0) EN(16)=EN(16)+0.1*EN(16)
      CALL DISKIO (2,NTA)
      RETURN
C
11    FORMAT (1H0,100(1H-),/)
12    FORMAT (1H ,* L. HT. OF HY. AG. = *,F10.4,* FT., AT *)
13    FORMAT (1H ,* WARNING -- L. HT. OF HY. AG. EXCEEDS 125.4 FT. -- OV
     1ERFLOW BEGINS *)
14    FORMAT (1H ,* NO. IN FILE *,I2,* = *,I5)
15    FORMAT (1H ,* PPTS. FILLING = *,I5)
16    FORMAT (1H ,* PPTS. DRAWING-OFF = *,I5)
17    FORMAT (1H ,* TOT. NO. OF PPTS. = *,I5)
18    FORMAT (1H ,* TOT. FREE HT. = *,F10.4,* FT. *)
19    FORMAT (1H ,* TOT. FREE HT. TOO LOW -- UNDERFL. OF HY. AG. IS INCR
     1EASED *)
20    FORMAT (1H ,* TOT. FREE HT. TOO HIGH -- CIRC. HRS. EXPECTED TO DEC
     1REASE -- DECREASE UNDERFLW. OF HY. AG. *)
```

```
END
```

Subroutine PRGPØ1 (NSET)

The subroutine simulates the scheduling of the precipitators for filling, seeding and discharging (ie. "draw-off"). The sections of the subroutine where each of these operations have been coded are labelled accordingly.

In the section on seeding operations, the precipitators which are being presently seeded (and are therefore "in circulation" in file 4) are first checked, and their new level computed. The temporary storage matrix DEMP stores the precipitator number, the final height and time at which seeding should be stopped for the maximum three precipitators that could be seeded at a time. Following this, the seed inlets are checked for availability. If one or more seeding has been completed, "new" empty precipitators are withdrawn from file 3 and placed in the matrix REACT. If no new seeding can be started, the program proceeds to filling operations. For actual seeding operations, the precipitator information is removed from REACT, processed, and then stored again in file 4 and in the matrix DEMP. Events are also scheduled for the future time instance when any one of the three possible precipitator seedings is completed. These events cause the seed inlet valves to shut and also notify seed inlet availability.

In the section on filling operations, the program checks which filling tank is presently being used, and if empty precipitators are available, and at what time these are available. If liquor inlets are available and there are empty precipitators in file 2, and if the filling restrictions are not violated, new fillings will be started. The choice of filling tanks for the supply of liquor to the precipitators depends on their level height. The one with the higher level is used first if possible. If not, the other tank is used (as

would be the case if the only empty precipitator is in group 1; then, only filling tank 1 would or can be used even if filling tank 2 has a higher level.) The subroutine POLYØ1 is called to yield information on the type of precipitator to be filled next. The matrix XLIQ stores temporary information on the maximum three precipitators being filled at any time. When each filling is complete, the precipitator is transferred to file 3 (precipitator queue waiting for seeding).

The last section of the subroutine concerns the discharging operations. First, the draw-off rate of the discharging precipitators is adjusted depending on the level height in the hydrate agitator. If the level exceeds 100 ft., the rate is decreased by 10%. If it falls below 90 ft., the rate is increased by 10%. The maximum and minimum draw-off rates are 350 and 200 cfm., respectively. A maximum of three draw-offs is allowed and only one precipitator in a group can be drawn-off at a time. The circulation time (COOKT) is computed for each precipitator at the start of its discharge. When discharging is complete, the precipitators are entered into file 2 again. This then completes the cycle of operations. (Shut-down and cleaning are not considered to be short-term scheduling operations.)

```
      SUBROUTINE PRGP01 (NSET)
C
C     MODEL SIMULATING THE SCHEDULING OPERATIONS IN THE PRECIPITATION
C     CIRCUIT
C
C     --- IMPORTANT NOTE ---
C     MODULES 1 AND 2 MUST BE THE FILLING TANKS
C     MODULE 3 MUST BE THE HYDRATE AGITATOR
C
      DIMENSION NSET(12,1)
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      COMMON /PREC2/ TCS,TFS
      COMMON /PREC/ AVAIL(3),MAVA(2,1)
      COMMON /TANK/ TTNOW(2),VOLB(2)
      COMMON /DIS/ JDIS(5),TTTN(5),LLCOU,VOLD
      COMMON /PRG/ JDD,JSD
      COMMON /SEED/ COARS,FINES
      COMMON /TK2/ HHH
      COMMON /TANK1/ HGT(2),TUT(2)
      COMMON /FH/ XHT(5)
      COMMON /DIS2/ ATR1(5),ATR2(5),XTIM(5)
      COMMON /FILL/ XLIQ(3,8)
      DIMENSION MDP(10), NIL(10), REACT(10,10), SFLOW(5)
      DIMENSION DEMP(3,3), VOLA(2)
C
C     TCS --- TIME AT WHICH COARSE SEED INLET BECOMES AVAILABLE
C     TFS --- TIME AT WHICH FINE SEED INLET BECOMES AVAILABLE
C     AVAIL(M) --- SETTINGS OF PRECIPITATOR INLET VALVES (M = 3)
C     AVAIL(1) --- NOT USED
C     AVAIL(2) --- COARSE SEED INLET VALVE
C     AVAIL(3) --- FINE SEED INLET VALVE
C     MAVA(1,1) --- INDICATOR FOR AVAILABILITY OF LIQUOR INLET NO. 1
C     MAVA(2,1) --- INDICATOR FOR AVAILABILITY OF LIQUOR INLET NO. 2
C     TTNOW(I) --- TIME AT WHICH OUTLET VALVE OF TANK I IS CLOSED
C     V(I) --- SETTING OF OUTLET VALVE OF FILLING TANK I
C     NTANK --- TANK NUMBER
C     JDIS(I) --- AVAILABILITY CODE OF DISCHARGE OUTLET OF GROUP I
C     TTTN(I) --- TIME AT WHICH OUTLET BECOMES AVAILABLE
C     JDD --- THE NUMBER OF INLETS AVAILABLE FOR LIQUOR CHARGING
C     COARS --- TOTAL AMOUNT OF COARSE SEED ALREADY USED
C     FINES --- TOTAL AMOUNT OF FINE SEED ALREADY USED
C
C     MDP(I) --- CONTAINS THE COLUMN NUMBERS OF THE FIRST TEN ENTRIES
C                IN FILE NO. 3
```

```
C     NIL(I) --- CONTAINS THE COLUMN NUMBERS OF THE FIRST TEN ENTRIES
C                IN FILE NO. 2
C     REACT(I,J) --- TEMPORARY STORAGE SPACE FOR PRECIPITATORS WAITING
C                    FOR SEEDING OR LIQUOR FILLING
C     SFLOW(I) --- THE TOTAL FLOW FROM THE PRECIPITATION CIRCUIT AT
C                  A GIVEN TIME INSTANCE
C
C     SI(1,3) --- COARSE SEED STREAM (CFM)
C     SI(2,3) --- FINE SEED STREAM (CFM)
C     EN(16) --- LIQUOR FILLING POLICY CODE
C
C     ------------------------------------------------------------------
C
C     SEEDING OPERATIONS
C
C     ------------------------------------------------------------------
C
C     SECTION TO CHECK WHETHER PRECIPITATORS ARE READY FOR
C     SEEDING AND TO OBTAIN THE TIMES WHEN THEY ARE AVAILABLE
C
C     ------------------------------------------------------------------
C
      IF (TNOW.GT.0.1) GO TO 2
      DO 1 I=1,3
      DEMP(I,3)=-100.0
1     CONTINUE
2     CONTINUE
      DO 3 I=1,3
      IF (DEMP(I,3).LT.0.0) GO TO 3
      CALL FIND (DEMP(I,1),5,4,2,KCOL,NSET)
      CALL RMOVE (KCOL,4,NSET)
      IF (I.EQ.1.OR.I.EQ.2) ATRIB(9)=SI(1,3)
      IF (I.EQ.3) ATRIB(9)=SI(2,3)
      ATRIB(5)=DEMP(I,3)-(DEMP(I,2)-TNOW)*ATRIB(9)/531.0
      IF ((DEMP(I,2)-TNOW).LT.0.01) DEMP(I,3)=-100.0
      CALL FILEM (4,NSET)
3     CONTINUE
C
C     IF JSD = 3, ALL INLETS ARE STILL BEING USED
C     THERE NO SEEDING IS POSSIBLE
C
      IF (JSD.EQ.3) GO TO 24
C
C     STORING A NEGATIVE VALUE IN THE THIRD ELEMENT OF THE TEMPORARY
C     STORAGE VECTOR FOR PRECIPITATORS WAITING FOR SEEDING
C
      DO 4 I=1,10
      REACT(I,3)=-100.0
4     CONTINUE
C
C     CHECKING FOR ANY PRECIPITATORS THAT ARE READY FOR SEEDING
C
C     FILE 3 CONTAINS ALL PRECIPITATORS WAITING FOR SEEDING
C
      IF (MFE(3).GT.0) GO TO 6
```

```
5     CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,75)
      IF (KPRNT(7).EQ.1) WRITE (6,91)
      NOM=0
C
C     IF NOM = 0, NO PRECIPITATOR IS READY FOR SEEDING
C
      GO TO 10
6     CONTINUE
      CALL CHECK (MFE(3),3,NSET)
      IF (ATRIB(3).GT.TNOW) GO TO 5
      MDP(1)=MFE(3)
      DO 7 J=1,5
      REACT(1,J)=ATRIB(J)
7     CONTINUE
      NOM=1
      DO 9 J=1,9
      JKE=MDP(J)
      MDP(J+1)=NSET(MX,JKE)
      IF (MDP(J+1).EQ.7777) GO TO 10
      CALL CHECK (MDP(J+1),3,NSET)
      IF (ATRIB(3).GT.TNOW) GO TO 10
      DO 8 I=1,5
      REACT(J+1,I)=ATRIB(I)
8     CONTINUE
      NOM=NOM+1
      IF (NOM.GE.10) GO TO 10
9     CONTINUE
C
C     ------------------------------------------------------------------
C
C     SECTION WHERE ACTUAL SEEDING OPERATIONS ARE PERFORMED
C
C     ------------------------------------------------------------------
C
10    CONTINUE
C
C     NOM = NUMBER OF PRECIPITATORS READY FOR SEEDING
C     IF NOM = 0, THERE IS NO PRECIPITATOR AVAILABLE FOR SEEDING
C
      IF (AVAIL(2).GT.1.5) TCS=TNOW
      IF (AVAIL(3).GT.0.5) TFS=TNOW
      IF (NOM.EQ.0) GO TO 24
      JSH=NOM
C
C     JSH IS SET EQUAL TO NOM BECAUSE NOM PRECIPITATORS IN QUEUE NO. 3
C     ARE CHECKED TO SEE IF THEY COULD BE SEEDED WITH AVAILABLE SEED
C     STREAMS (I.E. TO CHECK TYPE OF PRECIPITATORS AND TO TO CHECK
C     AVAIL(2) AND AVAIL(3))
C
C
C     SEEDING IS LIMITED BY ---
C     (1) AVAILABILITY OF ONLY 2 COARSE SEED INLETS,
C     (2) AVAILABILITY OF ONLY ONE FINE SEED INLET,
C     (3) AVAILABILITY OF ONLY A MAXIMUM OF 3 SEED INLETS,
```

```
C     (4) AVAILABILITY OF ANY PRECIPITATOR READY FOR SEEDING
C
      DO 23 J=1,JSH
      IF (KPRNT(7).NE.1) GO TO 11
      IF (REACT(J,3).LT.0.0) WRITE (6,76) REACT(J,3),J
      IF (REACT(J,3).GT.TNOW) WRITE (6,76) REACT(J,3),J
C
C     NREAC STORES THE CODE OF THE PRECIPITATOR TYPE --- 1 FOR REGULAR
C                                                --- 2 FOR SPECIAL
C
11    CONTINUE
      NREAC=REACT(J,4)+0.01
      IF (NREAC.EQ.1) GO TO 12
      IF (NREAC.EQ.2) GO TO 18
      IF (KPRNT(7).EQ.1) WRITE (6,77) NREAC
      RETURN
12    CONTINUE
      IF (JSD.EQ.3) GO TO 24
C
C     REGULAR --- COARSE SEED CHARGE
C     TEST TO SEE IF ANY OF THE TWO COARSE SEED CHARGE INLETS ARE
C     AVAILABLE
C     IF AVAIL(2) = 0.0 OR 1.0, AT LEAST ONE COARSE SEED INLET IS
C     AVAILABLE
C
      IF (AVAIL(2).LT.1.5) GO TO 13
      GO TO 23
13    CONTINUE
      WAIT2=TCS-REACT(J,3)
      CALL RMOVE (MDP(J),3,NSET)
      HT1=13.0
      DUR=531.0*HT1/SI(1,3)
      IF (WAIT2.LT.0.0) WAIT2=0.0
      IF (KPRNT(7).EQ.1) WRITE (6,90) ATRIB(1),ATRIB(2)
      IF (KPRNT(7).EQ.1) CALL CLOCK (DAYS,HRS,ZMINS,WAIT2)
      ATRIB(3)=TNOW+DUR
      AVAIL(2)=AVAIL(2)+1.0
      JSD=JSD+1
      REACT(J,3)=ATRIB(3)
      ATRIB(5)=ATRIB(5)+HT1
      DO 14 I=1,2
      IF (DEMP(I,3).LT.0.0) GO TO 15
14    CONTINUE
15    CONTINUE
      DEMP(I,1)=ATRIB(2)
      DEMP(I,2)=ATRIB(3)
      DEMP(I,3)=ATRIB(5)
      COARS=COARS+DUR*SI(1,3)
      IF (KPRNT(7).NE.1) GO TO 16
      WRITE (6,78)
      CALL CLOCK (DAYS,HRS,ZMINS,ATRIB(3))
      WRITE (6,79) COARS
      WRITE (6,91)
C
C     MAKING AN ENTRY INTO FILE NO. 4 - QUEUE WAITING FOR DISCHARGE
```

```
C
C
C     CIRCULATION HOURS INCLUDE SEEDING TIME
C
16    CONTINUE
      ATRIB(3)=ATRIB(3)-DUR
      ATRIB(5)=ATRIB(5)-HT1
      CALL FILEM (4,NSET)
      DO 17 I=1,5
      ATRIB(I)=0.0
17    CONTINUE
C
C     SCHEDULING EVENT TO SHUT COARSE SEED VALVE WHEN SEEDING IS DONE
C          --- I.E. SET AVAIL(2) = AVAIL(2) - 1.0
C
      ATRIB(1)=REACT(J,3)
      ATRIB(2)=2.0
      CALL FILEM (1,NSET)
      GO TO 22
18    CONTINUE
      IF (JSD.EQ.3) GO TO 24
C
C     SPECIAL --- FINE SEED CHARGE
C     TEST TO SEE IF THE FINE SEED CHARGE INLET IS AVAILABLE
C
      IF (AVAIL(3).LT.0.5) GO TO 19
C
C     IF AVAIL(3) = 0.0, THE FINE SEED INLET IS AVAILABLE
C
      GO TO 23
19    CONTINUE
      WAIT3=TFS-REACT(J,3)
      CALL RMOVE (MDP(J),3,NSET)
      HT1=10.0
      DUR=531.0*HT1/SI(2,3)
      IF (WAIT3.LT.0.0) WAIT3=0.0
      IF (KPRNT(7).EQ.1) WRITE (6,90) ATRIB(1),ATRIB(2)
      IF (KPRNT(7).EQ.1) CALL CLOCK (DAYS,HRS,ZMINS,WAIT3)
      ATRIB(3)=TNOW+DUR
      JSD=JSD+1
      REACT(J,3)=ATRIB(3)
      ATRIB(5)=ATRIB(5)+HT1
      DEMP(3,1)=ATRIB(2)
      DEMP(3,2)=ATRIB(3)
      DEMP(3,3)=ATRIB(5)
      FINES=FINES+DUR*SI(2,3)
      IF (KPRNT(7).NE.1) GO TO 20
      WRITE (6,80)
      CALL CLOCK (DAYS,HRS,ZMINS,ATRIB(3))
      WRITE (6,81) FINES
      WRITE (6,91)
C
C     MAKING AN ENTRY INTO FILE NO. 4 - QUEUE WAITING FOR DISCHARGE
C
C     CIRCULATION HOURS INCLUDE SEEDING TIME
```

```
C
20    CONTINUE
      ATRIB(3)=ATRIB(3)-DUR
      ATRIB(5)=ATRIB(5)-HT1
      CALL FILEM (4,NSET)
      DO 21 I=1,5
      ATRIB(I)=0.0
21    CONTINUE
      AVAIL(3)=1.0
C
C     SCHEDULING EVENT TO SHUT FINE SEED VALVE WHEN SEEDING IS DONE
C          -- I.E. SET AVAIL(3) = 0.0
C
      ATRIB(1)=REACT(J,3)
      ATRIB(2)=3.0
      CALL FILEM (1,NSET)
22    CONTINUE
C
C     MAKING AN ENTRY IN EVENT FILE
C
      ATRIB(1)=REACT(J,3)
      ATRIB(2)=10.0
      CALL FILEM (1,NSET)
      REACT(J,3)=-100.0
23    CONTINUE
C
C     ------------------------------------------------------------------
C
C     FILLING OPERATIONS
C
C     ------------------------------------------------------------------
C
C     SECTION TO CHECK WHICH FILLING TANK IS BEING USED -- CHECK
C     WHETHER EMPTY PRECIPITATORS ARE AVAILABLE AND THE TIMES WHEN
C     THEY ARE AVAILABLE
C
C     ------------------------------------------------------------------
C
24    CONTINUE
      IF (TNOW.GT.0.1) GO TO 26
      DO 25 IP=1,3
      XLIQ(IP,3)=0.0
      XLIQ(IP,5)=0.0
      XLIQ(IP,6)=0.0
25    CONTINUE
26    CONTINUE
      INDD=0
      VOLB(1)=0.0
      VOLB(2)=0.0
      VOLD=0.0
27    CONTINUE
      INDD=INDD+1
C
C     STORING A NEGATIVE VALUE IN THE THIRD ELEMENT OF THE TEMPORARY
C     STORAGE VECTOR FOR PRECIPITATORS WAITING FOR LIQUOR CHARGING
```

```
C
      DO 28 I=1,10
      REACT(I,3)=-100.0
28    CONTINUE
C
C     IF JDD = 3, ALL INLETS ARE STILL BEING USED
C     THEREFORE NO SEEDING OR LIQUOR CHARGING IS POSSIBLE
C
      IF (JDD.GE.3) GO TO 51
C
C
C     *****  LIMITATIONS  *****
C
C     HERE ARE THE 5 POSSIBLE CASES WHERE WE MUST CUT THE FILLING OF
C     PRECIPITATORS. IF AT THE END OF THE FILLING OF A PRECIPITATOR -
C
C     (1)  THERE ARE NO EMPTY PRECIPITATORS IN ANY GROUP.
C     (2)  THE NEXT PRECIPITATOR TO FILL IS A REGULAR AND THERE ARE NO
C          EMPTY PRECIPITATORS IN GROUPS 2, 3, 4, AND 5.
C     (3)  THE NEXT PRECIPITATOR TO FILL IS A SPECIAL AND THERE ARE NO
C          EMPTY PRECIPITATORS IN GROUPS 1, 2, 3, AND 4.
C     (4)  THE NEXT PRECIPITATOR TO FILL IS A REGULAR, THERE IS AN
C          EMPTY PRECIPITATOR ONLY IN GROUP 5 AND THE FILLING TANK NO.
C          ALREADY FILLS TWO PRECIPITATORS (THE FILLING TANK NO. 2
C          CANNOT FILL MORE THAN 2 PRECIPITATORS AT THE SAME TIME AND
C          THE FILLING TANK NO. 1 CANNOT FILL IN GROUP 5).
C     (5)  THE NEXT PRECIPITATOR TO FILL IS A SPECIAL, THERE IS AN
C          PRECIPITATOR ONLY IN GROUP 1 AND THE FILLING TANK NO. 1
C          ALREADY FILLS TWO PRECIPITATORS (THE FILLING TANK NO. 1
C          CANNOT FILL MORE THAN 2 PRECIPITATORS AT THE SAME TIME AND
C          THE FILLING TANK NO. 2 CANNOT FILL IN GROUP 1).
C
C
C     FILE NO. 2 CONTAINS ALL EMPTY PRECIPITATORS
C
      IF (MFE(2).GT.0) GO TO 29
      IF (KPRNT(7).EQ.1) WRITE (6,82)
      IF (KPRNT(7).EQ.1) WRITE (6,91)
      GO TO 51
29    CONTINUE
C
C     READING FROM FILE NUMBER 2
C
      CALL CHECK (MFE(2),2,NSET)
C
C     NSET COLUMN NUMBER OF FIRST ENTRY IN FILE 2 (FIRST IN FIRST OUT)
C
      NIL(1)=MFE(2)
      NDP=MFE(2)
      IF (ATRIB(3).GT.TNOW) GO TO 51
C
C     ATTRIBUTE (1) CONTAINS GROUP NUMBER
C     ATTRIBUTE (2) CONTAINS PRECIPITATOR NUMBER
C     ATTRIBUTE (3) CONTAINS TIME WHEN FIRST AVAILABLE
C     ATTRIBUTE (4) CONTAINS PRECIPITATOR TYPE
```

```
C           REGULAR = 1 --- SPECIAL = 2
C      ATTRIBUTE (5) CONTAINS LEVEL HEIGHT (FEET)
C
       DO 30 K=1,5
       REACT(1,K)=ATRIB(K)
30     CONTINUE
       DO 32 I=2,10
C
C      COLUMN NUMBER OF NEXT ENTRY IN MX ROW OF PREVIOUS ENTRY
C
       IAC=I
       NIL(IAC)=NSET(MX,NDP)
       NDP=NSET(MX,NDP)
       IF (NDP.EQ.7777) GO TO 33
       CALL CHECK (NDP,2,NSET)
       IF (ATRIB(3).GT.TNOW) GO TO 33
       DO 31 K=1,5
       REACT(I,K)=ATRIB(K)
31     CONTINUE
32     CONTINUE
33     CONTINUE
       INS=3-JDD
       IF (JDD.GE.3.AND.KPRNT(7).EQ.1) WRITE (6,83) JDD
       IF (JDD.GE.3.OR.INS.LE.0) GO TO 51
       DO 50 I=1,INS
       IF (TNOW.GE.0.1) GO TO 34
C
C      STORING EQUIPMENT NUMBER OF PRESENT MODULE
C
       NN29=EN(1)+0.01
C
C      READ EN(16) --- LEVEL HEIGHT (FEET) OF TANKS 1 AND 2
C
       CALL DISKIO (1,1)
       HGT(1)=EN(16)
       CALL DISKIO (1,2)
       HGT(2)=EN(16)
C
C      RETRIEVING EN VECTOR OF PRESENT MODULE
C
       CALL DISKIO (1,NN29)
34     CONTINUE
C
C      CHECKING FILLING POLICY
C
       M123=EN(16)+0.01
       CALL POLY01 (M123,N,0)
       JACK=0
C
C      IF LEVEL IN TANK 1 IS HIGHER, FIRST TRY TO FILL WITH TANK 1
C
       IF (HGT(1).GT.HGT(2)) GO TO 35
C
C      AND VICE VERSA ...
C
```

```
      IF (HGT(2).GE.HGT(1)) GO TO 38
      IF (KPRNT(7).EQ.1) WRITE (6,84)
      RETURN
35    CONTINUE
C
C     ATTEMPTING TO FILL WITH TANK 1
C
      DO 36 J=1,10
      IF (REACT(J,3).LT.0.0) GO TO 36
      IF (REACT(J,3).GT.TNOW) GO TO 51
      IF (REACT(J,1).GT.4.11) GO TO 36
      IF (N.EQ.1.AND.REACT(J,1).LT.1.1) GO TO 36
      IF (N.EQ.2.AND.REACT(J,1).LT.1.1.AND.MAVA(1,1).GE.2) GO TO 36
      IF (REACT(J,1).LT.4.11.AND.MAVA(1,1).LE.1) GO TO 37
36    CONTINUE
      JACK=JACK+1
      IF (JACK.LE.1) GO TO 38
      GO TO 51
37    CONTINUE
      NTA=1
      GO TO 41
38    CONTINUE
C
C     ATTEMPTING TO FILL WITH TANK 2
C
      DO 39 J=1,10
      IF (REACT(J,3).LT.0.0) GO TO 39
      IF (REACT(J,3).GT.TNOW) GO TO 51
      IF (REACT(J,1).LT.1.11) GO TO 39
      IF (N.EQ.2.AND.REACT(J,1).GT.4.11) GO TO 39
      IF (N.EQ.1.AND.REACT(J,1).GT.4.11.AND.MAVA(2,1).GE.2) GO TO 39
      IF (REACT(J,1).GT.1.11.AND.MAVA(2,1).LE.1) GO TO 40
39    CONTINUE
      JACK=JACK+1
      IF (JACK.LE.1) GO TO 35
      GO TO 51
40    CONTINUE
      NTA=2
41    CONTINUE
      REACT(J,3)=-100.0
C
C     ACTUAL LIQUOR FILLING OPERATIONS
C
C     CHECKING TO SEE IF THE LIQUOR INLET IS AVAILABLE
C     IF INLET IS AVAILABLE, MAVA(NTA,1) = 0 OR 1
C     IF INLET IS NOT AVAILABLE, MAVA(NTA,1) = 2
C     NTA DENOTES THE FILLING TANK NUMBER
C
      IF (MAVA(NTA,1).EQ.0.OR.MAVA(NTA,1).EQ.1) GO TO 42
      IF (KPRNT(7).EQ.1) WRITE (6,85) MAVA(NTA,1),NTA
      GO TO 51
42    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,86) NTA
      MAVA(NTA,1)=MAVA(NTA,1)+1
      IAC=J
```

```
      CALL RMOVE (NIL(IAC),2,NSET)
      JDD=JDD+1
      IF (JDD.GT.3.AND.KPRNT(7).EQ.1) WRITE (6,87)
      DO 43 JI=1,3
      IF (XLIQ(JI,6).LT.0.5) GO TO 44
43    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,88) JI,XLIQ(JI,6)
      RETURN
44    CONTINUE
      DO 45 IP=1,2
      XLIQ(JI,IP)=ATRIB(IP)
45    CONTINUE
      IF (XLIQ(JI,3).LT.TNOW) XLIQ(JI,3)=TNOW
      XLIQ(JI,5)=0.0
      XLIQ(JI,6)=1.0
      XLIQ(JI,8)=NTA
C
C     OBTAINING FILLING POLICY
C     IF M = 1 -- 1/1 POLICY, IF M = 2 -- 2/1 POLICY
C     IF N = 1 -- REGULAR, IF N = 2 -- SPECIAL
C
      M123=EN(16)+0.001
      CALL POLY01 (M123,N,1)
      IF (N.EQ.1) GO TO 46
      IF (N.EQ.2) GO TO 47
      IF (KPRNT(7).EQ.1) WRITE (6,89)
      RETURN
46    CONTINUE
      XLIQ(JI,4)=46.7
      GO TO 48
47    CONTINUE
      XLIQ(JI,4)=49.7
48    CONTINUE
C
C     WAIT1 = EMPTY PRECIPITATOR WAITING TIME
C
      WAIT1=XLIQ(JI,3)-ATRIB(3)
      IF (WAIT1.GE.0.0) GO TO 49
      WAIT1=0.0
      XLIQ(JI,3)=ATRIB(3)
49    CONTINUE
      IF (KPRNT(7).NE.1) GO TO 50
      WRITE (6,90) ATRIB(1),ATRIB(2)
      CALL CLOCK (DAYS,HRS,ZMINS,WAIT1)
      WRITE (6,91)
50    CONTINUE
      IF (JDD.LT.3.AND.INDD.EQ.1) GO TO 27
51    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,92)
      DUM=SI(1,3)
      DO 52 JI=1,2
      VOLA(JI)=0.0
      S=JI+2
      CALL STREAM (S)
      SI(1,3)=SN(IS,3)
```

```
      CALL DISKIO (1,JI)
      CALL TANK01 (2)
52    CONTINUE
      SI(1,3)=DUM
      CALL DISKIO (1,NE)
      DO 57 JI=1,3
      IF (XLIQ(JI,6).LT.0.5) GO TO 57
      NTA=XLIQ(JI,8)+0.01
      XLIQ(JI,7)=TUT(NTA)
      DUM=XLIQ(JI,5)
      XLIQ(JI,5)=XLIQ(JI,5)+(XLIQ(JI,7)/531.0)*(TNOW-XLIQ(JI,3))
      IF (XLIQ(JI,5).LE.XLIQ(JI,4)) GO TO 54
      DUM=XLIQ(JI,4)-DUM
      VOLA(NTA)=VOLA(NTA)+DUM*531.0
      XLIQ(JI,5)=XLIQ(JI,4)
      XLIQ(JI,3)=XLIQ(JI,3)+531.0*DUM/XLIQ(JI,7)
      GO TO 55
53    CONTINUE
      IF (KPRNT(7).NE.1) GO TO 57
      WRITE (6,93) XLIQ(JI,1),XLIQ(JI,2),XLIQ(JI,5)
      CALL CLOCK (DAYS,HRS,ZMINS,XLIQ(JI,3))
      GO TO 57
54    CONTINUE
      VOLA(NTA)=VOLA(NTA)+XLIQ(JI,7)*(TNOW-XLIQ(JI,3))
      XLIQ(JI,3)=TNOW
      IF (XLIQ(JI,5).LT.XLIQ(JI,4)) GO TO 53
55    CONTINUE
      XLIQ(JI,4)=(XLIQ(JI,4)-46.7)/3.0+1.0
      XLIQ(JI,6)=0.0
      IF (KPRNT(7).NE.1) GO TO 56
      WRITE (6,93) XLIQ(JI,1),XLIQ(JI,2),XLIQ(JI,5)
      CALL CLOCK (DAYS,HRS,ZMINS,XLIQ(JI,3))
C
C     TRANSFERRING PRECIPITATORS INTO FILE 3 (QUEUE WAITING FOR SEEDING
C
56    CONTINUE
      ATRIB(1)=XLIQ(JI,1)
      ATRIB(2)=XLIQ(JI,2)
      ATRIB(3)=XLIQ(JI,3)
      ATRIB(4)=XLIQ(JI,4)
      ATRIB(5)=XLIQ(JI,5)
      CALL FILEM (3,NSET)
      NTA=XLIQ(JI,8)+0.01
      XLIQ(JI,5)=0.0
      MAVA(NTA,1)=MAVA(NTA,1)-1
      JDD=JDD-1
      TTNOW(NTA)=TNOW
57    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,91)
      VOLB(1)=VOLB(1)+VOLA(1)
      VOLB(2)=VOLB(2)+VOLA(2)
C
C     ------------------------------------------------------------
C
C     DISCHARGING OPERATIONS
```

```
C
C     ------------------------------------------------------------------
C
      IF (JDD.GT.3.AND.KPRNT(7).EQ.1) WRITE (6,94) JDD
      IF (JDD.GT.3) RETURN
      IF (TNOW.GE.0.1) GO TO 59
C
C     RATE IS THE RATE OF DRAW-OFF FROM THE PRECIPITATORS IN CFM UNITS
C
      RATE=275.0
      STOR=TNOW
      NN29=EN(1)+0.01
C
C     MODULE 3 IS THE HYDRATE AGITATOR
C
      CALL DISKIO (1,3)
      HHH=EN(18)
      CALL DISKIO (1,NN29)
      DO 58 I=1,5
      ATR1(I)=0.0
      ATR2(I)=0.0
      XTIM(I)=0.0
      XHT(I)=0.0
      SFLOW(I)=0.0
58    CONTINUE
59    CONTINUE
      DO 60 I=1,5
      IF (JDTS(I).LT.1) TTTN(I)=TNOW
60    CONTINUE
      SO(1,3)=0.0
      IF ((TNOW-STOR).LT.15.0) GO TO 62
      IF (HHH.GE.90.0.AND.HHH.LE.100.0) GO TO 61
      IF (HHH.LT.90.0) RATE=RATE+0.1*RATE
      IF (HHH.GT.100.0) RATE=RATE-0.1*RATE
61    CONTINUE
      IF (RATE.LT.200.0) RATE=200.0
      IF (RATE.GT.350.0) RATE=350.0
      STOR=TNOW
62    CONTINUE
      IF (HHH.GT.110.0) GO TO 68
      IF (LLCOU.GE.3) GO TO 70
63    CONTINUE
      MF=MFE(4)
      M123=1
      IF (MF.LE.0.AND.KPRNT(7).EQ.1) WRITE (6,95)
      IF (MF.LE.0) RETURN
64    CALL CHECK (MF,4,NSET)
      TIME=ATRIB(3)
      IF (TIME.GT.TNOW) GO TO 69
      NGP=ATRIB(1)+0.01
C
C     NGP CONTAINS THE PRECIPITATOR GROUP
C
      COOKT=TTTN(NGP)-TIME
      COOKT=COOKT/60.0
```

```
      IF (COOKT.LT.0.0) GO TO 69
      IF (JDIS(NGP).NE.0) GO TO 66
C
C     IF JDIS(NGP) IS NOT EQUAL TO 0, THE DISCHARGE OUTLET OF GROUP NGP
C     IS NOT AVAILABLE
C
      CALL RMOVE (MF,4,NSET)
      IF (KPRNT(7).NE.1) GO TO 65
      WRITE (6,96) ATRIB(1),ATRIB(2),COOKT
      CALL CLOCK (DAYS,HRS,ZMINS,TTTN(NGP))
65    CONTINUE
      ATR1(NGP)=ATRIB(1)
      ATR2(NGP)=ATRIB(2)
      XTIM(NGP)=TTTN(NGP)
      XHT(NGP)=ATRIB(5)
      SFLOW(NGP)=RATE
C
C     DISCHARGE OUTLET NGP IS NOT AVAILABLE--- THEREFORE JDIS(NGP) = 1
C
      JDIS(NGP)=1
      LLCOU=LLCOU+1
      MF=MFE(4)
      GO TO 67
66    MF=NSET(MX,MF)
      IF (MF.EQ.7777) GO TO 69
67    CONTINUE
      IF (LLCOU.GE.3) GO TO 69
      IF (MF.LE.0) GO TO 69
      GO TO 64
68    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,97) HHH
      IF (KPRNT(7).EQ.1.AND.HHH.GE.125.4) WRITE (6,98)
69    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,91)
70    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,99)
      VOLC=0.0
      DO 73 K=1,5
      IF (ATR1(K).LT.0.1) GO TO 73
      SFLOW(K)=RATE
      XX=XHT(K)
      XHT(K)=XHT(K)-(TNOW-XTIM(K))*SFLOW(K)/531.0
      IF (XHT(K).GT.0.0) GO TO 71
      XHT(K)=0.0
      XTIM(K)=XTIM(K)+XX*531.0/SFLOW(K)
      VOLC=VOLC+XX*531.0
      IF (KPRNT(7).NE.1) GO TO 72
      WRITE (6,93) ATR1(K),ATR2(K),XHT(K)
      CALL CLOCK (DAYS,HRS,ZMINS,XTIM(K))
      GO TO 72
71    CONTINUE
      VOLC=VOLC+(TNOW-XTIM(K))*SFLOW(K)
      XTIM(K)=TNOW
      IF (KPRNT(7).NE.1) GO TO 73
      WRITE (6,93) ATR1(K),ATR2(K),XHT(K)
```

```
      CALL CLOCK (DAYS,HRS,ZMINS,XTIM(K))
      GO TO 73
72    CONTINUE
      ATRIB(1)=ATR1(K)
      ATRIB(2)=ATR2(K)
      ATRIB(3)=XTIM(K)
      ATRIB(4)=0.0
      ATRIB(5)=XHT(K)
C
C     FILE NO. 2 CONTAINS ALL EMPTY PRECIPITATORS WAITING FOR LIQUOR
C     FILLING
C
      CALL FILEM (2,NSET)
      M123=0
      JDIS(K)=0
      TTTN(K)=XTIM(K)
      LLCOU=LLCOU-1
      ATR1(K)=0.0
      SFLOW(K)=0.0
73    CONTINUE
      IF (KPRNT(7).EQ.1) WRITE (6,91)
      VOLD=VOLD+VOLC
      IF (M123.EQ.0.AND.HHH.LE.110.0) GO TO 63
      IF (LLCOU.GT.3.AND.KPRNT(7).EQ.1) WRITE (6,100)
      DO 74 I=1,5
      SO(1,3)=SO(1,3)+SFLOW(I)
74    CONTINUE
      RETURN
C
75    FORMAT (1H ,* NO PRECIPITATORS ARE AVAILABLE FOR SEEDING *)
76    FORMAT (1H0,* REACT(J,3) = *,F12.4,* J = *,I5)
77    FORMAT (1H0,* ERROR EXIT 4 - NREAC = *,I5)
78    FORMAT (1H ,* C. S. CHARGING COMPLETE AT *)
79    FORMAT (1H ,* TOT. COARSE SEED VOL. = *,E20.10,* CU. FT. *)
80    FORMAT (1H ,* F. S. CHARGING COMPLETE AT *)
81    FORMAT (1H ,* TOT. FINE SEED VOL. = *,E20.10,* CU. FT. *)
82    FORMAT (1H ,* THERE ARE NO EMPTY PRECIPITATORS AVAILABLE *)
83    FORMAT (1H0,* ERROR IN JDD -- JDD = *,I5)
84    FORMAT (1H0,* ERROR EXIT 2 - PRGP - ERROR IN LOGIC *)
85    FORMAT (1H0,* --- ERROR --- *,/*  MAVA(NTA,1) = *,I5,* NTA = *,I5)
86    FORMAT (1H ,* FILLING FROM TANK *,I2)
87    FORMAT (1H0,* ERROR IN JDD *)
88    FORMAT (1H0,* ERROR IN XLIQ(*,I3,* ,6) -- XLIQ(JI,6) = *,F7.2)
89    FORMAT (1H0,* ERROR EXIT 3 *)
90    FORMAT (1H ,* WAITING TIME OF PPT. GP. *,F5.1,* NO. *,F5.1,* IS *)
91    FORMAT (1H0,100(1H-),/)
92    FORMAT (1H ,* FILLINGS *,/)
93    FORMAT (1H ,* GP. *,F5.1,* NO. *,F5.1,*  HT. = *,F10.4,* FT.,* AT*)
94    FORMAT (1H0,* JDD = *,I5)
95    FORMAT (1H0,* ERROR EXIT - DISCHARGE - FILE 4 IS EMPTY -*,* NO PPT
     1S. IN CIRCULATION *)
96    FORMAT (1H ,* CIRC. TIME OF PPT. GP. *,F5.1,* NO. *,F5.1,* = *,F10
     1.4,* HRS.* - DISCH. AT *)
97    FORMAT (1H ,* HT. OF HY. AG. EXCEEDS 110 FT.* HT. = *,F12.4,* FT.*,
     1 NEW DRAW-OFFS POSTPONED *)
```

```
 98     FORMAT (1H ,* HT. AG. OVERFLOWS *)
 99     FORMAT (1H ,* DRAW-OFFS *,/)
100     FORMAT (1H0,* WARNING --- LLCOU .GT. 3 *)
       END
```

Subroutine VLBL (NSET)

The subroutine computes the material balance in the system using the information from the data pool. The terms in the equation:

INPUT - OUTPUT - ACCUMULATION = RESIDUAL

are evaluated and reported.

```
      SUBROUTINE VLBL (NSET)
C
C     MODULE TO CALCULATE VOLUME BALANCE (TYPE 5)
C
C     SYSTEM COMMON
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
C
      COMMON /TANK1/ HGT(2),TUT(2)
      COMMON /SEED2/ CS,FS
      COMMON /SEED/ COARS,FINES
      COMMON /TK2/ HHH
      COMMON /FH/ XHT(5)
      COMMON /DIS2/ ATR1(5),ATR2(5),XTIM(5)
      COMMON /FILL/ XLIQ(3,8)
      COMMON /FH2/ FREEH
      DIMENSION NSET(12,1)
      CS=0.0
      FS=0.0
      VOL2=0.0
      M=MFE(2)
1     CONTINUE
      IF (M.LE.0) GO TO 2
      CALL CHECK (M,2,NSET)
      VOL2=VOL2+ATRIB(5)
      M=NSET(MX,M)
      IF (M.NE.7777) GO TO 1
2     CONTINUE
      M=MFE(3)
3     CONTINUE
      IF (M.LE.0) GO TO 4
      CALL CHECK (M,3,NSET)
      VOL2=VOL2+ATRIB(5)
      M=NSET(MX,M)
      IF (M.NE.7777) GO TO 3
4     CONTINUE
      M=MFE(4)
5     CONTINUE
      IF (M.LE.0) GO TO 7
      CALL CHECK (M,4,NSET)
      VOL2=VOL2+ATRIB(5)
      IF (ABS(ATRIB(5)-59.7).LT.0.001) GO TO 6
      IF (ABS(ATRIB(4)-1.0).LT.0.1) CS=CS+(59.7-ATRIB(5))
      IF (ABS(ATRIB(4)-2.0).LT.0.1) FS=FS+(59.7-ATRIB(5))
```

```
6       M=NSET(MX,M)
        IF (M.NE.7777) GO TO 5
7       CONTINUE
        DO 8 I=1,3
        IF (XLIQ(I,6).LT.0.5) GO TO 8
        VOL2=VOL2+XLIQ(I,5)
8       CONTINUE
        DO 9 I=1,5
        VOL2=VOL2+XHT(I)
9       CONTINUE
        VOL2=VOL2*531.0
        VOL2=VOL2+HGT(1)*1964.0+HGT(2)*1257.0+HHH*531.0
        IF (TNOW.GE.0.1) GO TO 10
        VOL1=VOL2
        PL=0.0
        PT=0.0
        TPAST=0.0
10      CONTINUE
        CS=COARS-CS*531.0
        FS=FINES-FS*531.0
        CALL STREAM (1.0)
        PL=PL+SN(IS,3)*(TNOW-TPAST)
        CALL STREAM (11.0)
        PT=PT+SN(IS,3)*(TNOW-TPAST)
        TPAST=TNOW
        RES=CS+FS+PL+VOL1-PT-VOL2
        IF (KPRNT(7).NE.1) RETURN
        WRITE (6,12) TNOW
        WRITE (6,14) VOL1
        WRITE (6,15) TNOW,VOL2
        WRITE (6,16) CS,FS
        WRITE (6,17) PL,PT,RES
        WRITE (6,13)
C
C*****MONITORING STREAM FLOWS
C
        WRITE (6,18)
        DO 11 I=1,5
        IF (I.EQ.1) S=1.0
        IF (I.EQ.2) S=5.0
        IF (I.EQ.3) S=6.0
        IF (I.EQ.4) S=10.0
        IF (I.EQ.5) S=11.0
        CALL STREAM (S)
        WRITE (6,19) S,SN(IS,3)
11      CONTINUE
        WRITE (6,13)
        RETURN
C
12      FORMAT (1H ,* VOL. BAL. AT TIME *,F12.4,* MINS. */)
13      FORMAT (1H0,100(1H-),/)
14      FORMAT (1H ,* INITIAL VOL. AT TIME 0.0 MINS. = *,E20.10,* CU. FT.
       1*)
15      FORMAT (1H ,* VOL. AT TIME *,F12.4,* MINS. = *,E20.10,* CU.FT. *)
16      FORMAT (1H ,* C.S. VOL. = *,E20.10,* CU. FT. */*  F.S. VOL. = *,E
```

```
     10.10,* CU. FT. *)
17    FORMAT (1H ,* LIQ. INPUT = *,E20.10,* CU. FT. */*  PROD. OUTPUT =
     1*,E20.10,* CU. FT. */*  RES. ERROR = *,E20.10)
18    FORMAT (1H ,* FLOW CONDITIONS *)
19    FORMAT (1H ,* FLW. IN S.N. *,F5.1,F12.4,* CFM. *)
      END
```

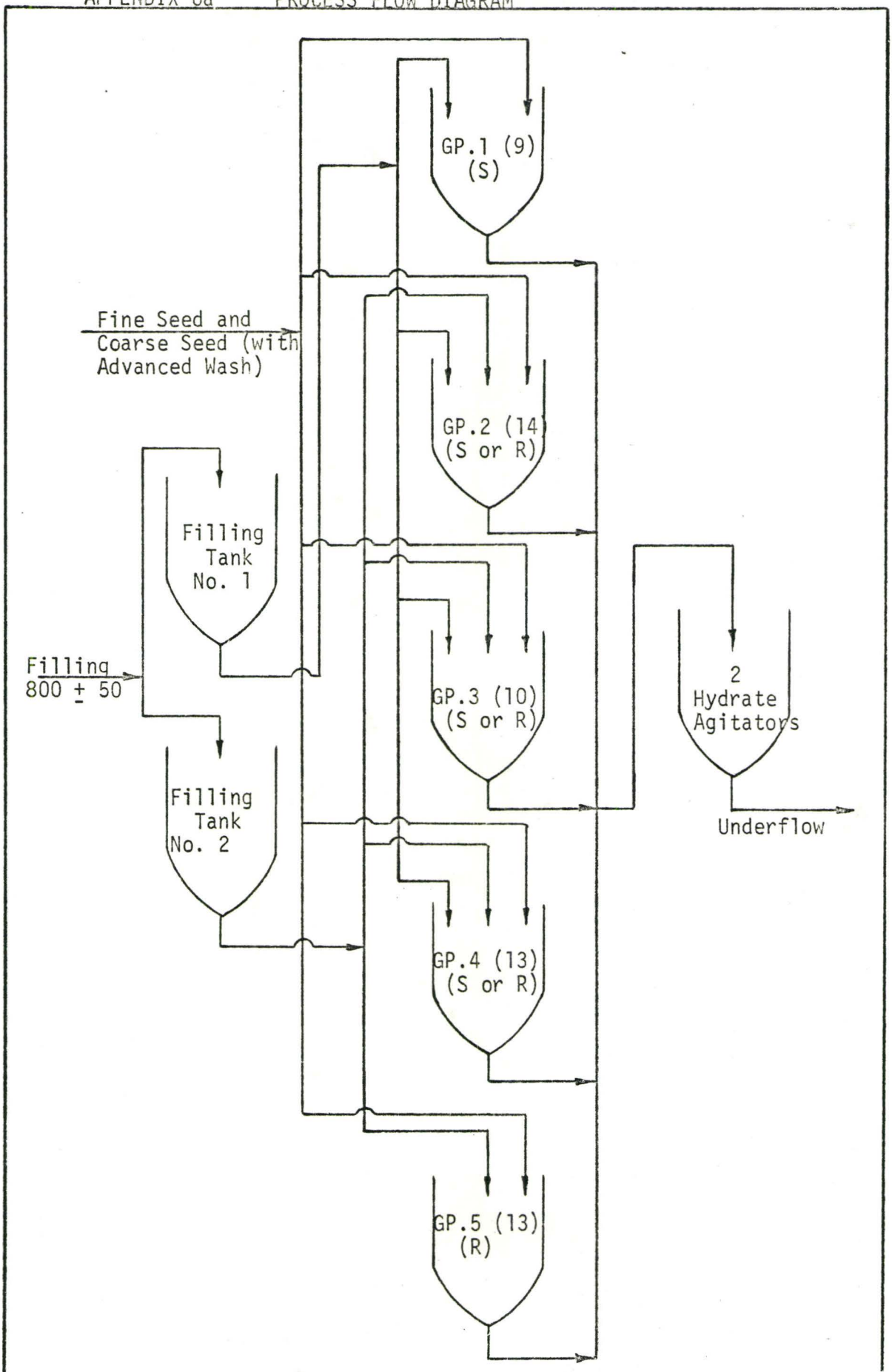
Subroutine RECDØ1 (NSET)

The subroutine obtains the simulation intermediate results from the data pool and records the data on punched cards. (The data cards are used in separate computer runs in which the results are tabulated and plotted graphically.)

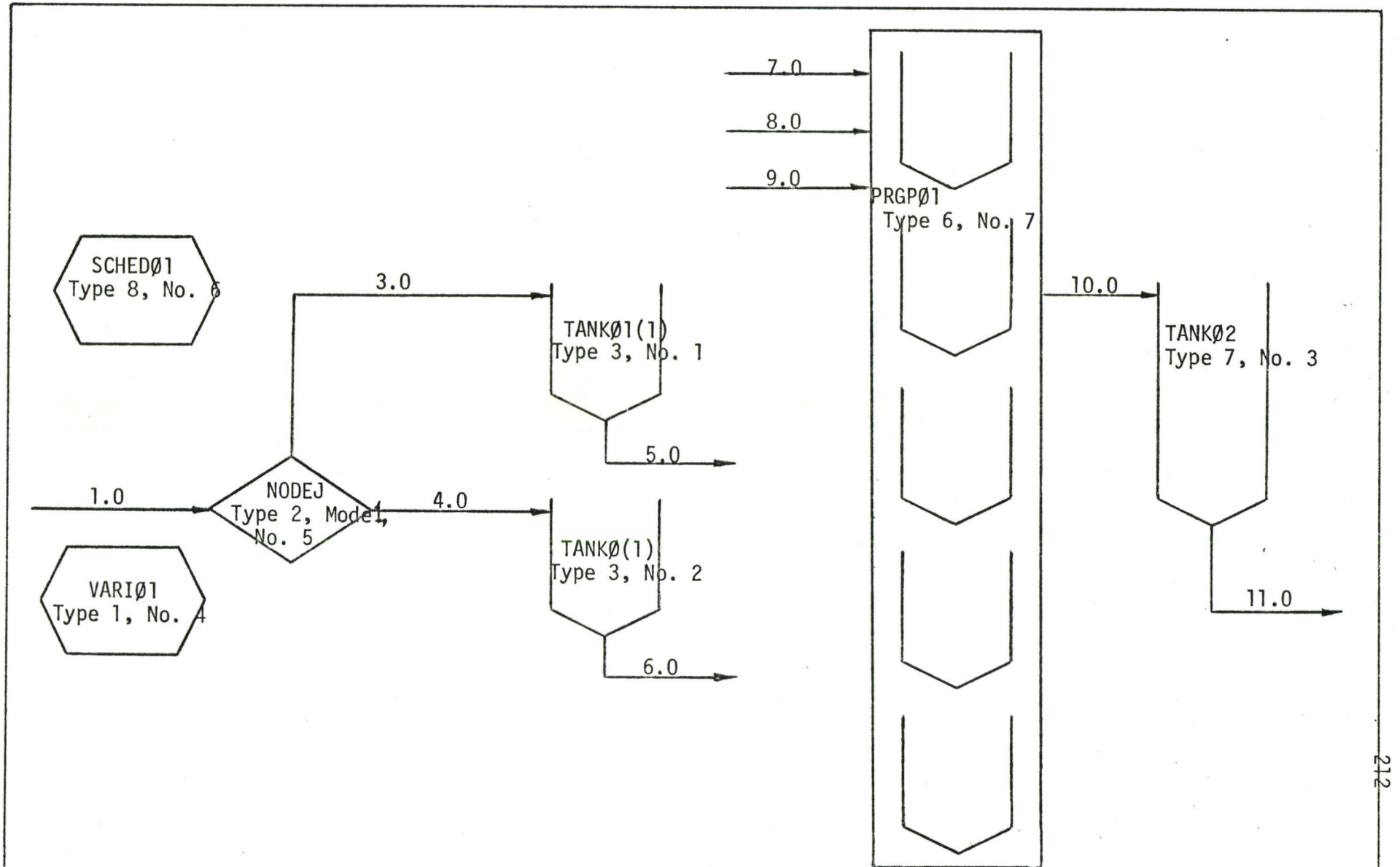
```
      SUBROUTINE RECD01 (NSET)
C
C     MODULE TO RECORD DATA GENERATED IN SIMULATION RUN (TYPE 4)
C
C     EN(4) --- NUMBER OF PPT. WHOSE HT. IS RECORDED
C
C     SYSTEM COMMON
C
      COMMON /A/ ATRIB(10),ENQ(10),INN(10),JCELS(5,22),KRANK(10)
      COMMON /B/ MAXNQ(10),MFE(10),MLC(10),MLE(10),NAME(20),NCELS(5)
      COMMON /C/ NQ(10),PARAM(20,10),QTIME(10),SSUMA(10,5),SUMA(10,5)
      COMMON /D/ VNQ(10)
      COMMON /E/ ID,IM,INIT,JEVNT,JMNIT,MFA,MSTOP,MX,MXC,MXX,NCLCT,NEP
      COMMON /F/ NEPRM,NHIST,NOQ,NORPT,NOT,NPRMS,NRUN,NRUNS,NSTAT,OUT
      COMMON /G/ SCALE,ISEED,TNOW,TBEG,TFIN
      COMMON /H/ JCLR,MON,NDAY,NPROJ,NYR
      COMMON /AA/ EEN(175),EN(25),KPRNT(10),LLST(20),NPOINT(20,2)
      COMMON /BB/ NS(25),SI(4,10),SN(17,10),SO(4,10)
      COMMON /CC/ III,IS,ISP,JJ,LOOP,MSN,NC,NCALC
      COMMON /DD/ NCOUNT,NE,NIN,NOCOMP,NOUT,NSR
      COMMON /SEED2/ CS,FS
      COMMON /SEED/ COARS,FINES
      COMMON /TK2/ HHH
      COMMON /TANK1/ HGT(2),TUT(2)
      COMMON /FH/ XHT(5)
      COMMON /FH2/ FREEH
      COMMON /FILL/ XLIQ(3,8)
      COMMON /DIS2/ ATR1(5),ATR2(5),XTIM(5)
      DIMENSION NSET(12,1)
      IF (TNOW.LT.0.1) NICK=2
      NICK=NICK+1
      IF (NICK.LT.3) RETURN
      NICK=0
      I=1
      J=2
      K=3
      L=4
      WRITE (7,5) CS,FS,TNOW,I
      WRITE (7,6) HGT(1),HGT(2),HHH,FREEH,TNOW,J
      CALL STREAM (1.0)
      A=SN(IS,3)
      CALL STREAM (11.0)
      B=SN(IS,3)
      CALL STREAM (5.0)
      C=SN(IS,3)
      CALL STREAM (6.0)
      D=SN(IS,3)
      WRITE (7,6) C,D,A,B,TNOW,K
      DO 1 JQ=2,4
      CALL FIND (EN(4),5,JQ,2,KCOL,NSET)
      IF (KCOL.EQ.0) GO TO 1
      CALL CHECK (KCOL,JQ,NSET)
      XT=ATRIB(5)
      GO TO 4
1     CONTINUE
```

```
      DO 2 M=1,3
      IF (ABS(XLIO(M,2)-EN(4)).GT.0.3) GO TO 2
      XT=XLIO(M,5)
      GO TO 4
2     CONTINUE
      DO 3 M=1,5
      IF (ABS(ATR2(M)-EN(4)).GT.0.3) GO TO 3
      XT=XHT(M)
      GO TO 4
3     CONTINUE
      XT=-100.0
4     CONTINUE
      WRITE (7,7) XT,TNOW,L
      RETURN
C
5     FORMAT (2E20.10,F12.4,27X,I1)
6     FORMAT (5F12.4,19X,I1)
7     FORMAT (2F12.4,55X,I1)
      END
```

APPENDIX 6a PROCESS FLOW DIAGRAM



APPENDIX 6b INFORMATION FLOW DIAGRAM



APPENDIX 7

DATA SET FOR CASE STUDY NO. 2

```
      6400 END OF RECORD
  SIMULATION OF THE SCHEDULING OPERATIONS IN THE PRECIPITATION CIRCUIT
2.0           11.0          23.0          1971.0        1.0
0.0           0.0           0.0           0.0           100.0
5.0           4.0           0.0           0.0           10000.0
1.0           3.0           3.0           3.0
1.0           1.0           1.0           1.0
1.0           0.0           0.0           1.0           0.0
3500.0        348.0
-1.0
0.0           0.0           0.0           0.0           0.0
1.0
0.0           4.0
1.0
0.0           10.0
2.0
2.0           9.0           0.0           0.0           0.0
2.0
3.0           22.0          0.0           0.0           0.0
2.0
4.0           31.0          0.0           0.0           0.0
3.0
2.0           10.0          0.0           1.0           46.7
3.0
3.0           23.0          0.0           2.0           49.7
3.0
4.0           32.0          0.0           2.0           49.7
4.0
2.0           11.0          0.0           1.0           59.7
4.0
3.0           24.0          -38.0         2.0           59.7
4.0
4.0           33.0          -76.0         1.0           59.7
4.0
2.0           12.0          -114.0        2.0           59.7
4.0
5.0           43.0          -152.0        2.0           59.7
4.0
```

```
3.0         25.0        -190.0      1.0         59.7
4.0
1.0         3.0         -228.0      2.0         59.7
4.0
5.0         44.0        -266.0      1.0         59.7
4.0
2.0         13.0        -304.0      2.0         59.7
4.0
5.0         45.0        -342.0      1.0         59.7
4.0
1.0         1.0         -380.0      2.0         59.7
4.0
4.0         34.0        -418.0      1.0         59.7
4.0
1.0         7.0         -456.0      2.0         59.7
4.0
5.0         46.0        -494.0      1.0         59.7
4.0
4.0         35.0        -532.0      2.0         59.7
4.0
2.0         14.0        -570.0      2.0         59.7
4.0
3.0         26.0        -608.0      1.0         59.7
4.0
1.0         4.0         -646.0      2.0         59.7
4.0
5.0         47.0        -684.0      1.0         59.7
4.0
4.0         36.0        -722.0      1.0         59.7
4.0
5.0         48.0        -760.0      1.0         59.7
4.0
2.0         15.0        -798.0      2.0         59.7
4.0
3.0         27.0        -836.0      1.0         59.7
4.0
```

4.0	37.0	-874.0	2.0	59.7
4.0				
1.0	8.0	-912.0	2.0	59.7
4.0				
4.0	38.0	-950.0	1.0	59.7
4.0				
5.0	49.0	-988.0	1.0	59.7
4.0				
3.0	28.0	-1026.0	2.0	59.7
4.0				
2.0	16.0	-1064.0	1.0	59.7
4.0				
4.0	39.0	-1102.0	1.0	59.7
4.0				
1.0	2.0	-1140.0	2.0	59.7
4.0				
2.0	17.0	-1178.0	1.0	59.7
4.0				
5.0	50.0	-1216.0	1.0	59.7
4.0				
4.0	40.0	-1254.0	1.0	59.7
4.0				
1.0	6.0	-1292.0	2.0	59.7
4.0				
3.0	29.0	-1330.0	1.0	59.7
4.0				
2.0	18.0	-1368.0	2.0	59.7
4.0				
5.0	51.0	-1406.0	1.0	59.7
4.0				
3.0	30.0	-1444.0	1.0	59.7
4.0				
1.0	5.0	-1482.0	2.0	59.7
4.0				
5.0	52.0	-1520.0	1.0	59.7
4.0				

```
2.0          19.0          -1558.0       2.0           59.7
4.0
5.0          53.0          -1596.0       1.0           59.7
4.0
4.0          41.0          -1634.0       2.0           59.7
4.0
2.0          20.0          -1672.0       2.0           59.7
4.0
4.0          42.0          -1710.0       1.0           59.7
4.0
2.0          21.0          -1748.0       2.0           59.7
4.0
5.0          54.0          -1786.0       1.0           59.7
0.0
0.0          0.0           0.0           0.0           0.0
  THIS DATA SET IS FOR OBTAINING A PRINTOUT OF THE SIMULATION RESULTS
  ------------------------------------------------------------------------
1.0          1.0           1.0
2.0          1.0
8.0          0.0
6.0          4.0           5.0           7.0           1.0
2.0          3.0           8.0
-11.0
1.0          9.0           2.0           2.0           3.0
3.0          9.0           1.0           1.0           2.0
3.0
3.0
1.0          1.0           800.0         0.0
8.0          1.0           75.0          0.0
9.0          1.0           115.0         0.0
8.0
1.0          3.0           18.0          0.0           0.0
1.0          3.0
1.0          5.0
1.0          0.0           1964.0
2.0          3.0           18.0          0.0           0.0
```

```
1.0          4.0
1.0          6.0
1.0          0.0          1257.0
3.0          7.0          18.0          0.0          0.0
1.0          10.0
1.0          11.0
1000.0       0.0          100.0
4.0          1.0          15.0          0.0          0.0
0.0          0.0          0.0           0.0          0.0
0.0          0.0          0.0           0.0          0.0
5.0          2.0          15.0          0.0          0.0
1.0          1.0
2.0          3.0          4.0
6.0          8.0          15.0          5.0          300.0
0.0
0.0
7.0          6.0          16.0          0.0          0.0
2.0          8.0          9.0
1.0          10.0
1.0
8.0          5.0          15.0          0.0          0.0
0.0
0.0
'            END OF FILE
```