Computers & Chemical Engineering Vol. 5, No. 4, pp. 263–276, 1981
Printed in Great Britain.

0098-1354/81/040263-14$02.00/0
Pergamon Press Ltd.

AN INTERACTIVE PROCESS FLOWSHEETING AND SIMULATION SYSTEM BASED ON RELATIONAL DATA STRUCTURES

SURRENDRA P. SINGH
Getty Oil Co., Houston, TX 77401, U.S.A.

and

BRICE CARNAHAN*
The University of Michigan, Ann Arbor, MI 48104, U.S.A.

(*Received* 4 *June* 1981)

Abstract—An interactive graphical interface has been developed for a modular-sequential steady-state chemical process simulation program. The interface software, called AGPSS (A Graphical Process Simulation System), consists of FORTRAN-IV routines for creating new equipment unit schematics, for drawing a process flowsheet, for accepting process equipment and stream parameters, for performing automatic analysis for recycle nets, for supervising execution of the steady-state material and energy balancing simulator, and for displaying results. The program is designed to operate from refresh terminals functioning as remote time-sharing devices on the University of Michigan's Amdahl 470V/6 computing system. The Integrated Graphics (IG) software package is used for all interactive graphical operations.

All data, including both picture and process information, are organized in a relational data structure containing a total of thirteen relations. The general algorithms for addition, retrieval, and deletion of data in the relational structure require just six set-theoretic operators, which are encoded as FORTRAN utility routines. These data structures and the associated set operators assisted materially in simplifying the design, programming, and testing of the interactive graphical interface.

Scope—The process flowsheet is the natural communication medium for chemical process engineers. Process design work is highly iterative, involving frequent changes in equipment arrangement and important parameter values. Thus, one would expect computer graphics and interactive computing to play a much larger role in computer-aided process simulation and design than has been the case to date.

In this study, we establish specifications for effective interaction between process engineer and a modular-sequential steady-state process simulator using computer graphics. A prototype software interface has been designed, implemented, and tested that meets these specifications. A key feature of the interface is the use of a relational data base and a small number of set-theoretic operators for storage, deletion, modification, and retrieval of both picture and process information.

Conclusions and Significance—Important features of user-friendly interactive flow-sheeting and simulation systems have been identified and incorporated into a working prototype software interface. The relational data base and associated set-theoretic operators used have proved to be valuable in the processing of both picture and process information, and aided materially in simplifying the design, programming, and testing of the interface. These simple and powerful data structures should be given serious consideration by designers of future graphics and simulation software.

INTRODUCTION

Steady-state chemical process simulators were first developed in the early 1960's and now play a significant role in process simulation and design work in the chemical, petrochemical, and petroleum industries. A recent review[1] describes the most important of these programs, many of which are available on a purchase, lease, or use-royalty basis. Typically, these simulators are batch-oriented, and require extensive hand encoding of the process flowsheet, forms-filling, and card punching or file preparation by the user. Since the systems do little "design" as opposed to simulation, the user normally refines his design by repeated simulation of the encoded process with different system equipment parameter (size specifications, operating conditions) and stream parameter (temperature, pressure, composition) values. Any major change in the design concept, such as a rearrangement of the process equipment and material flows, requires starting the whole process from the beginning.

Since the process flowsheet is the natural com-

*Author to whom correspondence should be addressed.

munication medium for chemical process engineers, it seems odd that computer graphics has played so small a role in the development of process simulation programs. Additionally, given the nature of the process design process, involving iterative simulation and frequent changes in the process flowsheet, one would expect interactive computing to play a much larger role in chemical process design than it in fact does.

The work reported here describes the development of a designer-oriented prototype interactive graphical interface for chemical process simulation called AGPSS (A Graphical Process Simulation System)[2]. The system was developed on an Amdahl 470V/6 computer at the University of Michigan using a refresh graphics terminal.

OBJECTIVES AND SPECIFICATIONS

The principal objectives of this study were to determine specifications of a practical interactive computer graphics interface, to design a data structure suitable for storage, modification, and retrieval of both picture (process flowsheet) and process parameter information, to develop functioning computer programs for the interface, and to test the system for its operational effectiveness.

Some system specifications were:

(1) The user should be able to draw a schematic for a new type of process unit at the terminal, define its characteristics (e.g. the number, locations, and types of its input/output ports), and have the system store the description in a library for later reference when drawing a process diagram.

(2) The user should be able to draw easily the flowsheet for a chemical process using refresh (and possibly storage-tube) terminals available on the University's central computing facility. The simulator interface should establish the process topology (unit-stream connectivity of the process flowsheet), and create a suitable mathematical description of the process for the simulation program.

(3) The system should allow the user to make quite general and arbitrary modifications and rearrangements of the process flowsheet. The system should be able to save the graphical database, even for a partially completed flowsheet, on direct access disk storage for later retrieval and possible further modification. Any individual user should be allowed to have more than one flowsheet, possibly for completely different processes, available to the system.

(4) The system should accept process information (equipment and stream parameter values), and maintain it in the process data base for use by the simulator. This process information should be saved, at the user's discretion, on direct access disk storage for later retrieval or modification.

(5) The system should check the process flowsheet and data for uniqueness and consistency, and prompt the user for corrections.

(6) The system should determine the recycle nets, if any, and establish cut variables for determining the order in which individual unit calculations will be performed[3].

(7) The user should be able to monitor the progress of simulator calculations, particularly when recycle sets are present in the process. Convergence rates of any selected tear variable should be displayed on request. The user should be able to interrupt the simulator at any time to modify iteration parameters and to restart the computations.

(8) The user should be able to get a hard copy of the process flowsheet, or any part of it, with simulation results (e.g. stream temperatures) displayed, in the form of a digital plot.

(9) Simulation results should be displayed on the screen in a flexible manner under user control.

(10) If desired, all simulator results should be printed for later reference.

(11) At any time, the user should be able to interrupt the current activity of the interface, and enter any other reasonable operating mode; e.g. the user should be able to interrupt a simulation run, change suitable parameters, and restrart the simulator, or to interrupt a simulation run, revert to the flowchart modification mode, make changes in the diagram, and then begin a simulation for the modified flowsheet.

(12) The interactive graphics language used to communicate with the interface should satisfy the following requirements: (a) It should be simple, and easy to learn and use. (b) It should be consistent in its structure, so that the user can avoid making trivial mistakes. (c) The command processor should make provision for error recovery. Helping comments and prompting messages should be provided for the user. (d) Whenever possible, the command processor should let the user know what it is doing at the moment, and should provide prompting comments on what should be done next. (e) Commands, once made, should be revocable if not already processed, or insofar as seems practicable, interruptable, while being processed.

DATA STRUCTURE

The data structure for the graphical interface must contain at least the following:

(1) A description of each processing unit in the flowsheet. This includes graphical data necessary to draw its schematic, the number and order of input/output (port) streams, the number of equipment parameters (operating characteristics of the equipment) and their values, and identification of the unit computation routine (library subprogram) that will be used to compute the simulated behavior of the unit.

(2) The number and associated stream parameters of each stream in the process (i.e. the molar flow rate, temperature, pressure, enthalpy, composition, vapor fraction, etc.) and the graphical data necessary to draw the streams in the flowsheet.

(3) The process topology; this incorporates all information about the connectedness of streams and the input/output ports of equipment units.

(4) The identity of all materials being processed by the system, normally in the form of catalogued codes, for later use in generating physical property estimates for compounds and mixtures in the chemical process.

The data structure selected for AGPSS is a relational one that allows all storage, modification, and retrieval operations to be described in terms of set theoretic operators[4]. A few definitions should suffice to describe the nature of the relational structures used:

(1) An *n-tuple* is an *ordered* collection of n *components* a_i and is represented as follows:

$$\langle a_1, a_2, \ldots, a_n \rangle.$$

Here, a_i is the ith component of the n-tuple and is said to have *dimension* i; usually a_i has a *role name* or *attribute* associated with it. An attribute whose value

uniquely identifies its tuple (i.e. whose value differs in each tuple of similar structure) is called a *key*; there may be more than one such attribute.

(2) A *set A* is an *unordered* collection of *m elements* x_i and is represented as follows:

$$A = \{x_1, x_2, \ldots, x_m\}.$$

If x is an element of set A we shall write

$$x \in A$$

and if x is not an element of A we shall write

$$x \in' A.$$

If set B is obtained from set A we write

$$B = \{x: S(x, A)\}$$

where $S(x, A)$ is a *conditional predicate.*

(3) A *relation R* is a set of n-tuples, each of which has its first component from set D_1, its second component from set D_2, and its nth component from set D_n. The sets D_i might contain numbers or characters, for example, and are called the *domains* of R; n is called the *degree* of R. The number of tuples in relation R is called the *cardinality* of R. Thus relation R consists of n-tuples in which the *component order* is significant and determines the data relationships; the order of the n-tuples in relation R is *not* significant. A relation must have at least one key. Sets D_i need not be distinct.

As an example, consider the following scheme for encoding the topological information for a process flowsheet, i.e. information about how process units and process streams are connected. Let each unit in the process be assigned a unique identifier (e.g. an *equipment number*), each input and output port for each type of equipment be assigned a unique identifier (e.g. input port 1, input port 2, output port 1, etc.), and each stream a unique identifier (e.g. a *stream number*). Then we could describe the *topology relation T* as

$$T = \{x: x \in \langle N_{es}, N_{op}, N_{et}, N_{ip}, N_s\rangle\}$$

where T contains 5-tuples whose components, in order, are:

N_{es} is the equipment number for the source of stream N_s.

N_{op} is the output port number of equipment N_{es} to which stream N_s is attached.

N_{et} is the equipment number for the (target) output of stream N_s.

N_{ip} is the input port number of equipment N_{et} to which stream N_s is attached.

N_s is the stream number.

The domains of R are the sets D_1, D_2, D_3, D_4 and D_5 containing, respectively, unit numbers, output port numbers, unit numbers, input port numbers, and stream numbers. Note that the sets D_i need not be distinct. Relation T contains one tuple for each stream (stream order is not important), and has cardinality equal to the total number of streams in the flowsheet. The key for the relation T is domain 5, i.e. the stream number. Note that process flows under steady-state operation are directed; that is, material flows in stream N_s from process unit N_{es} to process unit N_{et}, but not vice versa.

Thus, a relation is a set of n-tuples containing related data. The collection of all relations is called the *universe.*

SET OPERATIONS

Set theoretic operators are used to store and to retrieve data from the relational data structure. All retrieved data are in the form of a single set or relation. Only six set operators are required to perform all storage, retrieval, deletion, and modification operations on the relational data structure of AGPSS. They are:

(1) *Union.* $C = A \cup B = \{x: (x \in A) \vee (x \in B)\}$. Thus x is included in union set C if it belongs to either set A or set B.

(2) *Intersection.* $C = A \cap B = \{x: (x \in A) \wedge (x \in B)\}$. Thus x is included in intersection set C if and only if it belongs to both sets A and B.

(3) *Relative complement.* $C = A \sim B = \{x: (x \in A) \wedge (x \in' B)\}$. The relative complement set C of set A with respect to set B contains only those elements of set A which are not elements of set B.

(4) *Restriction.* The restriction of set A by set B is a set C:

$$C = A|B = \{x: (x \in A) \wedge [(\exists b \in B) \wedge (b \subset x)]\}.$$

Here, b represents an m-tuple belonging to set B, and x represents an n-tuple belonging to set A. The tuple size (maximum dimension) of A must be at least as large as that for B ($n \geq m$). Tuple b is said to be *contained in* x ($b \subset x$) if the first m components of x are identical to the m components of b. Thus the restriction operator finds those tuples in set (relation) A whose first m components match those of tuples in set B.

(5) *Domain.* $C = D_{ij}(A) = \{x: (\exists a \in A) \wedge (x \subset a)\}$. Here, the domain set $D_{ij}(A)$, consists of n-tuples (size $n = j - i + 1$) obtained by extracting, in sequence, those components of all tuples of A having dimension d such that $i \leq d \leq j$.

(6) *Match.* The match of set A by set B is a set C:

$$C = A \equiv_{ij} B = \{x: (x \in A) \wedge [(\exists b \in B) \wedge (\exists c \in D_{ij}(A)) \wedge (b \subset c)]\}.$$

A and B contain n-tuples and m-tuples, respectively ($m \leq n$, where $m = j - i + 1$). Match is a general form of restriction; it finds those tuples of A whose ith through jth components are identical with the m components of tuples in B. C contains n-tuples.

Adding a new tuple to relation A requires the following steps:

(1) Create a temporary relation S containing the tuple to be added to relation A.

(2) Perform the union operation on relations S and A, and save the result as new relation A.

$$A = A \cup S$$

(3) Destroy S.

Deleting a tuple from relation A requires the following steps:

(1) Create a temporary relation S containing a 1-tuple whose value is the key value of the tuple to be deleted from A. Let the key for A be component k.

(2) Match (or restrict, if $k = 1$) the relation A with relation S, creating a new temporary relation $\bar{S}$.

$$\bar{S} = A \equiv_{kk} S$$

(3) Find the relative complement of A with respect to $\bar{S}$ and save the result as the new relation A.

$$A = A \sim \bar{S}$$

Data can be retrieved from relation A by implementing the first two steps of the deletion algorithm of the preceding paragraph.

These general algorithms for addition, retrieval, and deletion of data from the relational data structure, and the six set-theoretic operators described earlier were programmed as FORTRAN-IV routines and used very extensively in the development of AGPSS. All data, including both process and picture information have been organized into a total of thirteen relations (sets) in the relational data structure.

GRAPHICS HARDWARE AND SOFTWARE

A DEC 339/PDP-9 refresh graphics terminal operating as a time-shared device on the University of Michigan's Amdahl 470V/6 central computer was used for the bulk of the development work of AGPSS. This terminal was equipped with a light pen, illuminated function board, and a Bell Model 33 terminal.

The Integrated Graphics (IG) System[5, 6] software was used to perform graphical operations. IG is a device-independent program suite that allows application programs to operate on a variety of graphics terminals of both refresh and storage-tube type. AGPSS functions much more conveniently using a refresh terminal, but can be used from a storage tube device. The major penalty involved in use of a storage tube terminal, such as a Tektronix 4010, is the speed of interaction, and time lost in redrawing frames following deletion or repositioning of flowchart streams or units (the light-pen "picking" operation on a refresh scope is replaced by thumbwheel crosshair positioning on the Tektronix unit).

The IG software allows the user to create a hierarchial picture structure that is particularly useful in the flowsheeting process. Briefly, every system picture element is a *subpicture* of another subpicture, all elements being subpictures of a main *virtual* picture space. Any subpicture may have any number of its own subpictures; the "imbeddedness" of a subpicture A relative to another subpicture B is determined by the "level" of subpicture A with respect to subpicture B, which is assigned level zero. Each subpicture is assigned a name, either by the user's application program or, by default, by the IG system. As a rule, any part of a picture or subpicture that the user wants to manipulate independently must be created as a subpicture. Frequently used graphical symbols (e.g. the schematic for a pump) are created as *graphical objects*. Whenever a (possibly transformed) copy of the graphical object is desired, an *instance* of the object can be created as a subpicture in the virtual picture space.

A subpicture is modified by calling IG *transformation* routines; translation, rotation, and scale-change transformations are available. *Windowing transformations* may be applied to any portion of the virtual picture space; this allows the user to place a virtual camera anywhere over the virtual picture space as desired. The windowed picture is then subjected to a *viewporting transformation* that maps it onto a selected portion of the terminal screen. IG allows the user to display any number of windowed pictures on different screen viewports simultaneously. AGPSS uses only the two-dimensional IG transformations, although three-dimensional transformations are also available.

IG allows two types of graphical input called *coordinate input*, in which a tracking cross on the screen can be used to determine the location of a point on the screen, and *pick input*, which allows the user to point at any subpicture on the screen with a light pen.

One of the important considerations in developing AGPSS was to produce a highly structured and modular system. All calls to the IG routines have been incorporated into 12 AGPSS *graphical operation routines* listed in Table 1. To the extent that another IG-like software package is available elsewhere, the AGPSS system should be transportable. This has not been done, however, and would not be a trivial task because the subpicturing capability of the IG system is not available in most graphics software. All AGPSS progrms are written in G-level IBM FORTRAN-IV.

All graphical data (e.g. the endpoint coordinates of all line segments in a stream drawing) are maintained in the AGPSS relational database. The graphical operations routines call upon a total of 23 IG subroutines to implement line and character drawing operations in the virtual picture space, and the windowing and viewporting transformations required to generate screen images.

GRAPHICAL COMMUNICATION AND DISPLAY ORGANIZATION

AGPSS uses a problem-oriented graphical command language. All commands are displayed in a menu as 'light buttons'. Typically, several potential commands are displayed simultaneously; the particular menu on the screen depends upon the current activity of the user and of the interface. Explanatory messages are displayed to aid the user. The terminal is restricted to the entering of alphanumeric identifiers and labels and numerical data such as equipment and stream parameters.

The screen area of the display is divided into five separate regions (viewpoints), as shown in Fig. 1: (1) the menu region, (2) the program mode identification region, (3) the helping and error comments region, (4) the work

Table 1. Graphical operation routines

Name	Function
INIT	Initializes the graphics (IG) software package.
PUTLAB	Includes a text string in a picture description.
DRAW	Includes a sequence of straight line segments in a picture description.
SUBP	Defines and draws (or destroys) a subpicture.
GOBJ	Creates, displays, and destroys graphical objects.
DVAR	Displays the numerical value of a variable at a specified location with selected format.
TRANS	Applies translational, rotational, or scaling transformations to subpictures.
XYIN	Finds the coordinates of a point on the screen.
PICK1	Allows the user to pick a subpicture with light pen.
PICK2	Allows the user to pick subpictures selectively from a menu.
PICK3	Retrieves the name and subpicture level of a picked subpicture.
GPLOT	Generates a digital plot of any windowed picture.

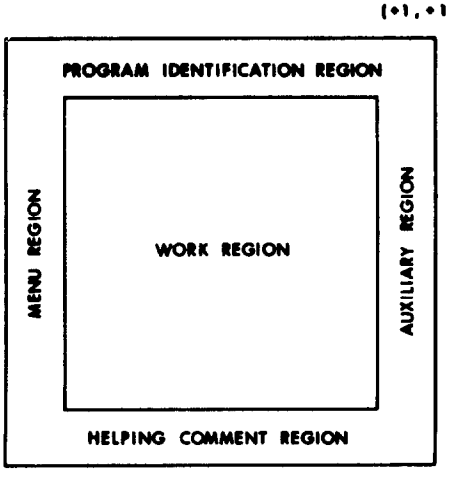

Fig. 1. Subdivision of the screen area into five display regions (viewports).

region and (5) the auxiliary region. The menu of command light buttons appears at the left side of the screen. The program mode identification region is used to display the name of the currently active AGPSS operating mode (see the next section), and appears near the top of the screen. Helping comments appear near the bottom of the screen. The user draws the process flowsheet (or parts of it) in the work region, which occupies most of the central part of the screen; the system uses this region for flowsheet and parameter value display. The auxiliary region is used for display of all other information (for example, the iteration count during simulator recycle calculations).

ORGANIZATION OF THE AGPSS SOFTWARE

AGPSS is organized in a highly modular fashion, and operates in five distinct modes as follows:

(1) Store/Delete.
(2) Topology.
(3) Build.
(4) Simulate.
(5) Results.

An overall view of the major program subsystems is shown in Fig. 2. The functioning of the system in the five different operating modes will be described only briefly here.

The store/delete mode

This mode is invoked when the user wishes to enter and store (or to delete) the description of equipment schematics that are to be used in creating process flowsheets. These schematics correspond to unit computation subprograms that are to be incorporated into the simulator library or the user's private unit routine library. This mode is independent of all other operating modes and is not used at all when all equipment schematics for all unit computation routines are already available in a unit computation routine description file.

In its current state, the AGPSS simulator library contains eleven unit computation routines for modeling the equipment shown in Table 2; schematics for each of these unit types are already incorporated into the unit computation routine description file. All of these models are quite simple, and many would not be of adequate complexity for industrial level modeling; however, they are adequate for demonstration purposes, and have been used to test the functioning of the AGPSS software. When a new unit routine is added to the library it is necessary to use the store/delete mode to create a corresponding unit schematic.

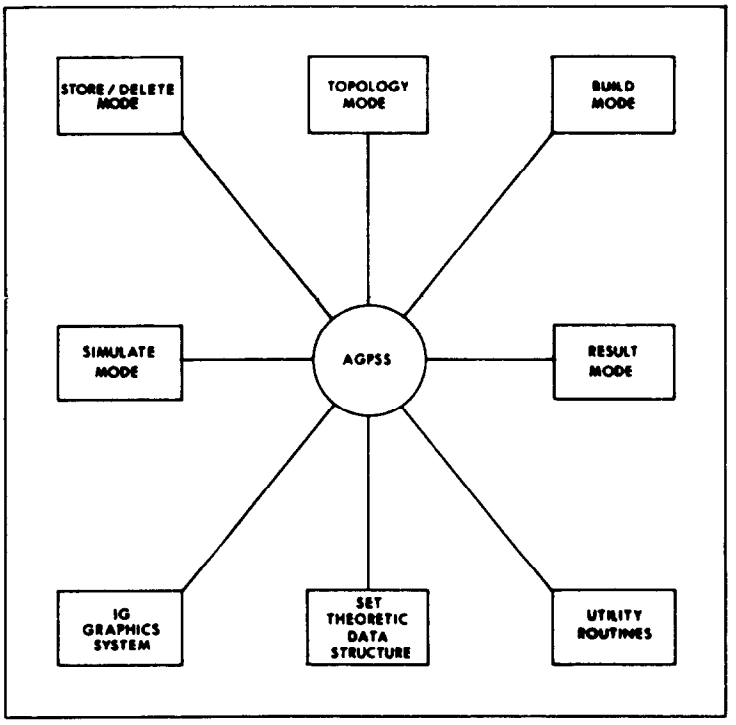

Fig. 2. Major subsystems of the AGPSS software.

Table 2. Equipment models in the AGPSS library

Absorber
Compressor
Vapor–liquid equilibrium flash unit
Simple fractionator
Mixer–splitter
Pump
Simple reactor
Steam-heated exchanger
Stripping column
Pressure reduction valve
Water-cooled exchanger

The hierarchy of menu commands displayed in the menu region of the screen is shown in Fig. 3. Once the user has selected the STORE/DELETE operating mode from the menu region, the major commands STORE, DELETE and DONE appear in the menu region, (see Fig. 3). If the user chooses the STORE command, the list of picture element types (H.LINE, V.LINE, RECTANGLE, CIRCLE, ARC, LINE) and two control commands (DEL.ELM, DONE) appear in the menu region. By selecting appropriate picture element types from the menu, the user can assemble the desired schematic. The size and location of each picture element is determined by two points specified by the user with the light pen operating in coordinate input mode. AGPSS internally assigns a unique name and element type to each picture element, and saves its name, type, and the coordinates of the two points in a temporary relation. Once all elements of the schematic have been drawn, its description is saved in a permanent relation P of 7-tuples having the structure:

$$\langle R_1, R_2, E, x_1, y_1, x_2, y_2 \rangle.$$

Here, R_1 and R_2 are the first four and last two characters respectively, of the unit computation routine name (a FORTRAN subroutine name), E is a picture element type code, and (x_1, y_1) and (x_2, y_2) are the coordinates of the first and second points, respectively.

Relation P contains similar information for all elements of all schematics that can appear later in process flowsheets; i.e. it contains all information required to construct graphical objects corresponding to process equipment pictures.

After the body of the equipment schematic has been drawn, the user specifies with the light pen the locations of three points, the first two to establish the positions of the endpoints of a port arrow, and the third the position of a label for the port. The label is entered at the terminal. Input and output port information for all schematics is saved in relations PI and PO, respectively, containing 9-tuples of the form:

$$\langle R_1, R_2, n, x_1, x_2, x_3, y_1, y_2, y_3 \rangle$$

Here, R_1 and R_2 are the first four and last two characters, respectively, of the corresponding unit computation routine name, n is the input or output port number, and (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the coordinates of the three points selected by the user.

Picture elements and input and output ports can be deleted and redrawn as desired during store/delete mode operation. Figure 4 shows some digital plots of the screen content during creation of an absorber schematic using the store/delete mode.

Once the schematic is completed, the user specifies the number of equipment parameters required by the corresponding unit computation routine, and a label of no more than sixteen characters for each parameter at the terminal. This information for all parameters of all unit types is stored in relation Q containing 7-tuples of the form:

$$\langle R_1, R_2, i, D_{i1}, D_{i2}, D_{i3}, D_{i4} \rangle.$$

Here, R_1 and R_2 are the first four and last two characters, respectively, of the unit computation routine name, i is the equipment parameter number, and D_{i1}, D_{i2}, D_{i3}, and D_{i4} contain a sixteen character label (4 characters/4 byte word) for the equipment parameter.

A schematic and its accompanying unit computation description are deleted from the system with the command DELETE. Store/delete mode operation is terminated with the DONE command (following the DONE command for the port-drawing operations). When the user issues this DONE command, AGPSS either accepts from the user the names of those unit types that are to be used later in a process or collects the names of all the unit types in set P, and stores them in a list. It then includes the name of the unit computation routine for general mixer/splitter calculations in the list of names if it has not already appeared, and generates the source code for a FORTRAN subroutine named EQCAL that

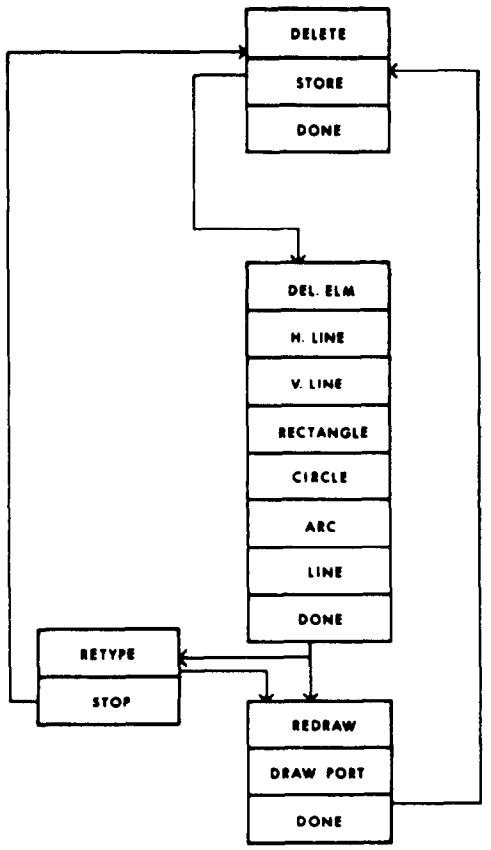

Fig. 3. Hierarchy of displayed (menu) commands for the store/delete mode.

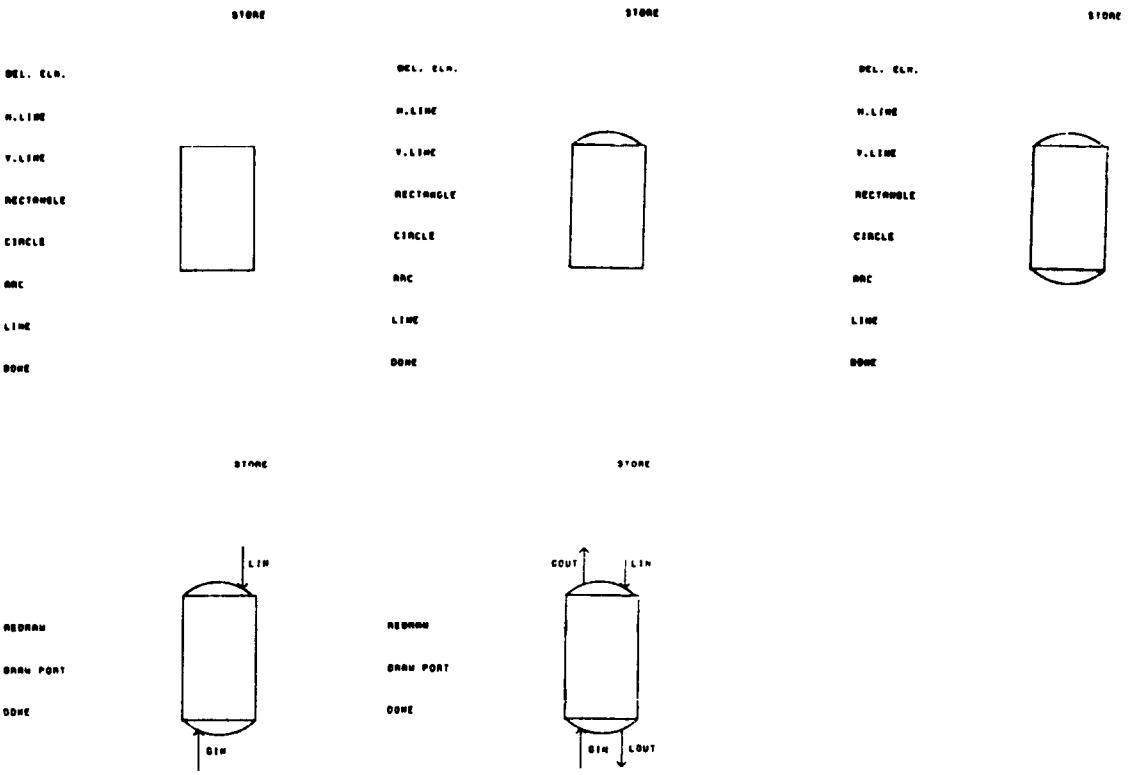

Fig. 4. Digital plots of screen content during creation of absorber schematic using store/delete mode.

will be needed later on during execution of the simulate mode and writes it into a user file.

The name of the unit computation for general mixing/splitting calculations is automatically included in the list of unit types, since a mixer/splitter unit is automatically incorporated into the process flow sheet whenever the user specifies a T-type stream junction in the process flow diagram during topology mode operation as described later.

Subroutine EQCAL contains FORTRAN statements that call upon the individual unit computation routines. This makes the system open-ended and allows the user to add new (and delete existing) unit computation routines to (from) the AGPSS or private routine library. Since EQCAL contains references only to those unit routines selected by the user, the object code actually loaded into the memory will contain only those simulation routines needed for the user's process (in a system with a dynamic loader, this artifice would be unnecessary).

The topology mode

The topology mode is used to draw the flowsheet in the work region. Equipment schematics are selected by choosing the unit name from the menu region and positioning the equipment schematic in the work region with the light pen. Each schematic is assigned a unique name. Stream pictures are drawn as directed line segments and assigned unique stream names. The position, orientation, and size of any equipment schematic can be changed at any time during the flowsheet drawing process. Equipment schematics and streams can be deleted, if necessary. If any unit schematic is transformed, all its attached streams move with it to maintain diagram connectivity.

AGPSS defines a flow diagram space, called the F-space, and all equipment and stream schematics are drawn in the coordinates of this space. All (or part of) the F-space can be displayed in the work region. The user chooses his window by pointing at diagonally opposite corners of a rectangle in the full F-space display in the work region. Whenever the characters 'MOVE DIAGRAM' appear in the upper left corner of the display screen, the user can 'move' the diagram (i.e. move the F-space window) by pointing the light pen at any of the four edges of the screen (left, right, top, bottom). This gives the impression of moving a camera viewfinder over the flowsheet; zoom effects are invoked quite simply.

All equipment schematics and streams in the flow diagram are drawn in the coordinate system of the F-space, which is overlaid with a rectangular grid having N_x and N_y subdivisions in the x and y coordinate directions, respectively. The cells created by this grid are labeled as *edge* cells, *blocked* cells (cells occupied by equipment schematics or by the intersection of stream segments), and *open* cells.

The entire flow diagram is created as a subpicture named DIAG. The windowing and viewporting transformations mentioned earlier are applied to this subpicture, so that one can draw a large diagram by drawing only a small part of it at a time in the work region of the display scope. The flow diagram contains four additional subpictures at level 1 named EQPM, STRM, PRT1, and

PRT2 as shown in Fig. 5. DIAG has a tree structure, so that subpictures at level 1 (such as STRM) contain subpictures at level 2 (*S1*, *S2*, etc.), which in turn contain subpictures at level 3 (*ST1*, *ST2*, etc.). This structuring allows straightforward manipulation of flow chart elements; e.g. when a stream subpicture (say *S1*) is moved, its stream name subpicture (*ST1*) moves with it.

Figure 6 shows the heirarchy of AGPSS commands displayed in the menu region for all modes except the Store/Delete mode, already shown in Fig. 3. For example, when the topology mode light button is selected, the potential commands STREAM, EQUIPMENT, DELETE, TRANSFORM, ASSIGN, BACKUP, SHOW, and DONE appear, if EQUIPMENT is selected from the menu, a list of equipment types appears; the user selects one from the list and points to the location on the screen where the schematic should appear. AGPSS then performs several functions.

(1) It retrieves the graphical data for the equipment picture and its associated input and output ports from relations *P*, *PI*, and *PO*.

(2) It creates temporary relations named *IP* (input port) and *OP* (output port) with 3-tuples containing the equipment name, port number and port name for each port associated with the schematic.

(3) It creates a 13-tuple that is stored in relation *EQ* (along with a similar tuple for each equipment schematic in the *F*-space). Components of the tuple include the equipment name and type, a cumulative scale factor by which the schematic has been scaled, a cumulative rotation factor by which the schematic has been rotated about its own center, the name of the subpicture that displays the equipment name, and the x and y coordinates of four points in the *F*-space. The scale factor is initially set to one, and subsequent scaling transformations are multiplicative. The rotation angle is initially set to zero, and subsequent rotational transformations are additive. The four points are the locations of the center point of the schematic, of the equipment name, and of the lower left and upper right corners of the smallest rectangle that completely encloses the schematic.

(4) It creates an n-tuple containing the equipment name and a component for each equipment parameter value (initially set to zero) in a relation called *EQP*.

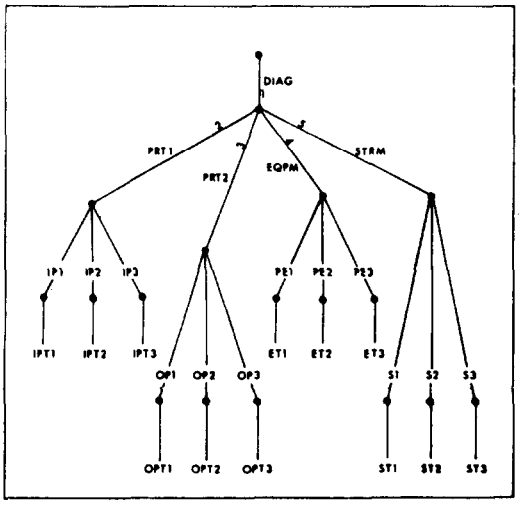

Fig. 5. The tree structure of the subpicture DIAG.

(5) Assigns as 'blocked' all cells in the *F*-space that are occupied by the smallest rectangle containing the equipment picture.

(6) It calls on the appropriate graphics routines to incorporate the equipment picture into the *F*-space, and to display the picture on the screen.

Any number of equipment schematics can be positioned in the *F*-space at this level of topology-mode execution. When the desired number of equipment pictures (at any time) have been drawn, the light button DONE is selected and the previous level of options is displayed in the menu region.

The user selects the STREAM command to draw stream pictures. A stream picture consists of a set of horizontal and vertical directed straight line segments representing a single process stream connecting two equipment ports. Stream junctions of the *T*-type are allowed; AGPSS automatically creates a new mixer/splitter unit at such a junction point, and creates three streams where one existed previously.

Any stream in the process diagram belongs to one of three stream classes:

(1) It connects two equipment schematics or an equipment schematic and a stream junction.

(2) It is a process feed stream, either to an equipment schematic or a stream junction.

(3) It is a process product stream, either an output stream from an equipment schematic or a stream junction.

The procedure for drawing a stream involves:

(1) Entering a stream name at the terminal.

(2) Pointing at the output port of the source equipment schematic or at an existing stream where a source junction is to be created.

(3) Pointing at the input port of the target equipment schematic or at an existing stream where a target stream junction is to be created.

AGPSS then creates an appropriate tuple in the topology relation *T* described earlier.

Three different stream drawing algorithms are incorporated. The first simply establishes a direct connection between a chosen output port on one schematic and a chosen input port on another. In flowsheets with a high equipment schematic density, this algorithm can lead to esthetically unattractive results, although a later simulation will be unaffected. At a later stage of topology-mode processing (using the REDRAW subcommand of the TRANSFORM command), the user can invoke a second algorithm to redraw (using piecewise horizontal and vertical line segments) any selected stream picture. A third stream drawing algorithm is also available that uses a two-dimensional form of the general path-connection algorithm of Lee[7]. Lee's algorithm has proved to be quite expensive in computing time, particularly when the F-space is sparsely populated with equipment schematics, but produces esthetically very attractive process stream layouts.

Regardless of the algorithm used for stream drawing, data for all streams in the flow diagram are stored in two relations, *S1* and *S2*. Each stream picture is broken into individual straight-line subpictures, whose endpoints are saved, along with a point index, and the stream name, in relation *S1*. The coordinates of the center (x_c, y_c) of the straight line segment of a stream picture that has the greatest length and the direction (horizontal or vertical) of that segment are saved, along with the stream name, in relation *S2*. This information is used to position the

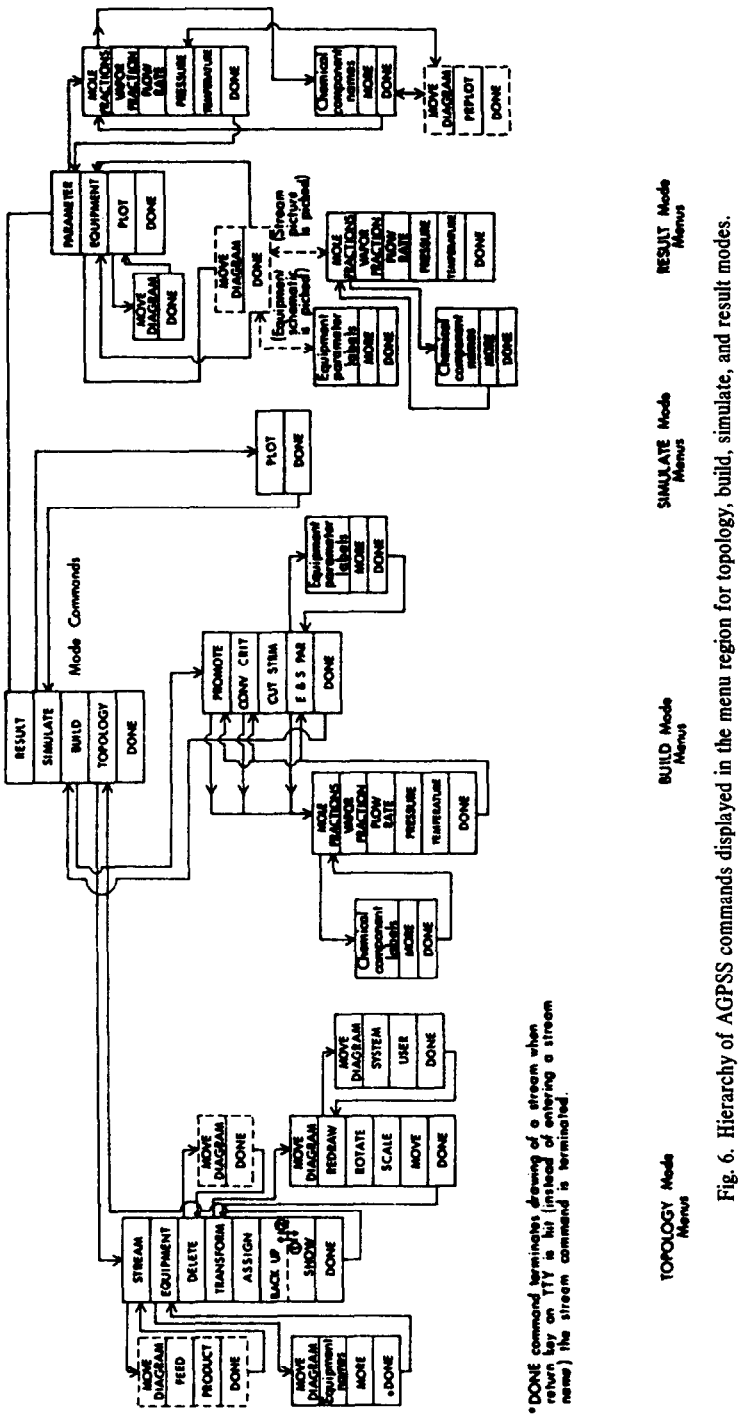

Fig. 6. Hierarchy of AGPSS commands displayed in the menu region for topology, build, simulate, and result modes.

stream name near the stream in the flowsheet. For each stream, a single m-tuple containing the stream name and one component for each stream parameter is created and stored in relation *SP* (m depends on the number of chemical components in the process). All stream parameters are initialized to zeros. Figure 7 shows the stream pictures generated automatically by AGPSS during the initial stream-drawing process. Figure 8 shows the same process diagram after the user has "cleaned up" unattractive parts of the diagram by redrawing some stream pictures (and moving some equipment schematics, as described below).

When the TRANSFORM command is selected, streams can be redrawn as already described. In addition, any equipment schematic can be translated from one position to another (MOVE), rotated about its own center (ROTATE), or changed in size (SCALE). When a schematic is to be moved it is selected with the light pen and "follows" the light pen to the desired new location in the *F*-space. If rotations or scaling operations are required, appropriate parameter values are entered at the terminal. In any event, when an equipment schematic is to be transformed, AGPSS does the following:

(1) The names of the unconnected input–output port pictures belonging to this equipment schematic and corresponding input–output port numbers are determined. These unconnected input–output port subpictures are erased from the flow diagram and the corresponding 3-tuples are deleted from the relations *IP* or *OP*.

(2) The names of all the input–output stream pictures connected to the equipment schematic are determined and these stream pictures are erased from the *F*-space. The graphical data for these streams are deleted from relations *S*1 and *S*2. The cells blocked by the intersection of line segments of stream pictures are set as open cells in the *F*-space.

(3) The cells blocked by the equipment schematic are reset as open cells in the *F*-space.

(4) An *IG* system routine is called to transform the equipment picture in the *F*-space.

(5) The new cells in the *F*-space belonging to the smallest rectangle enclosing the equipment picture are set as blocked.

(6) The value of the 13-tuple corresponding to the equipment schematic in the relation *EQ* is updated.

(7) All the unconnected input–output port subpictures corresponding to the input–output port numbers determined in step 1 are redrawn; the 3-tuples corresponding to these input–output port subpictures are included in relations *IP* and *OP*, respectively.

(8) All the previously connected input–output stream pictures are redrawn. The new sequential points in the *F*-space needed for redrawing these stream pictures are determined by the standard algorithm. The graphical data corresponding to the new stream pictures are included in relations *S*1 and *S*2.

The DELETE command is used to delete, by pointing with the light pen, any stream or equipment schematic subpictures. Appropriate set operations are invoked to remove all references to the deleted subpictures from the relational data structure.

The ASSIGN command is used to assign (at the terminal) names and component codes for the chemical species present in the process to relation *C* containing 5-tuples. Property data are generated by a proprietary program developed by Motard[8] of Washington University (St. Louis); data for 62 compounds, mostly hydrocarbons, are available.

The SHOW command is used to display any portion of the *F*-space in the work region; e.g. to "blow up" (enlarge) a small portion of the *F*-space. The coordinates of the lower left and upper right corners of the rectangular region in the *F*-space to be displayed in the work region are entered by the user using the light pen tracking cross. The AGPSS system calls the window transformation routine with the new window coordinates entered by the user to display the selected portion of the *F*-space.

The BACK UP command is used to redisplay the portion of the diagram that was originally there before the user issued the SHOW command. The AGPSS system calls the window transformation routine with the old window coordinates as its parameters to redisplay in the work region the previous portion of the *F*-space. The DONE command terminates the operation of the topology mode.

The build mode

The purpose of the build mode is to check for process consistency, to construct process stream-unit connection matrices, to accept equipment and stream parameter values, to identify recycle nets, tear streams and variables, and to accept other control parameters from the user. The user can enter parameters in arbitrary order, if he chooses, after selection of the E&S PAR (equipment and stream parameter) command, by simply pointing at the desired stream or equipment picture; AGPSS then requests the appropriate parameter values, which the user enters at the terminal. Or, the user can request that AGPSS ask for all essential information in a predetermined order. The first method is preferred if only a few changes in previously assigned values are to be made; if all parameters are to be assigned (before the first simulation run, for example), then the latter method is very convenient.

When equipment parameter values are to be entered for any equipment in the process flow diagram, AGPSS performs the following functions:

(1) The labels for the equipment parameters are retrieved from relation Q, and displayed as light buttons in the menu region.

(2) The current equipment parameter values are retrieved from relation EQP and displayed in the work region next to the equipment parameter labels.

(3) The user is asked (by displaying a message in the helping comment region) to enter all, or some, of the new equipment parameter values from the terminal. The new values are displayed in place of the old values in the work region. The user can change individual parameter values by pointing at the corrresponding label and entering a new value. If the user points at the label MORE more labels of equipment parameters are displayed along with their parameter values.

(4) The equipment parameter values in the relation EQP are updated.

When the stream parameter values in SP are to be assigned or updated for any stream in the diagram, AGPSS performs similar functions.

If the user prefers to select some (or all) of the cut streams for recycle calculations to be carried out by the steady-state modular–sequential simulator, he may do so by pointing at the stream(s) with the light pen after selecting the CUT STRMS command. The algorithm of Barkley & Motard[3] is used to perform a recycle analysis to determine all the recycle nets and to establish the calculation order for the unit computation routines

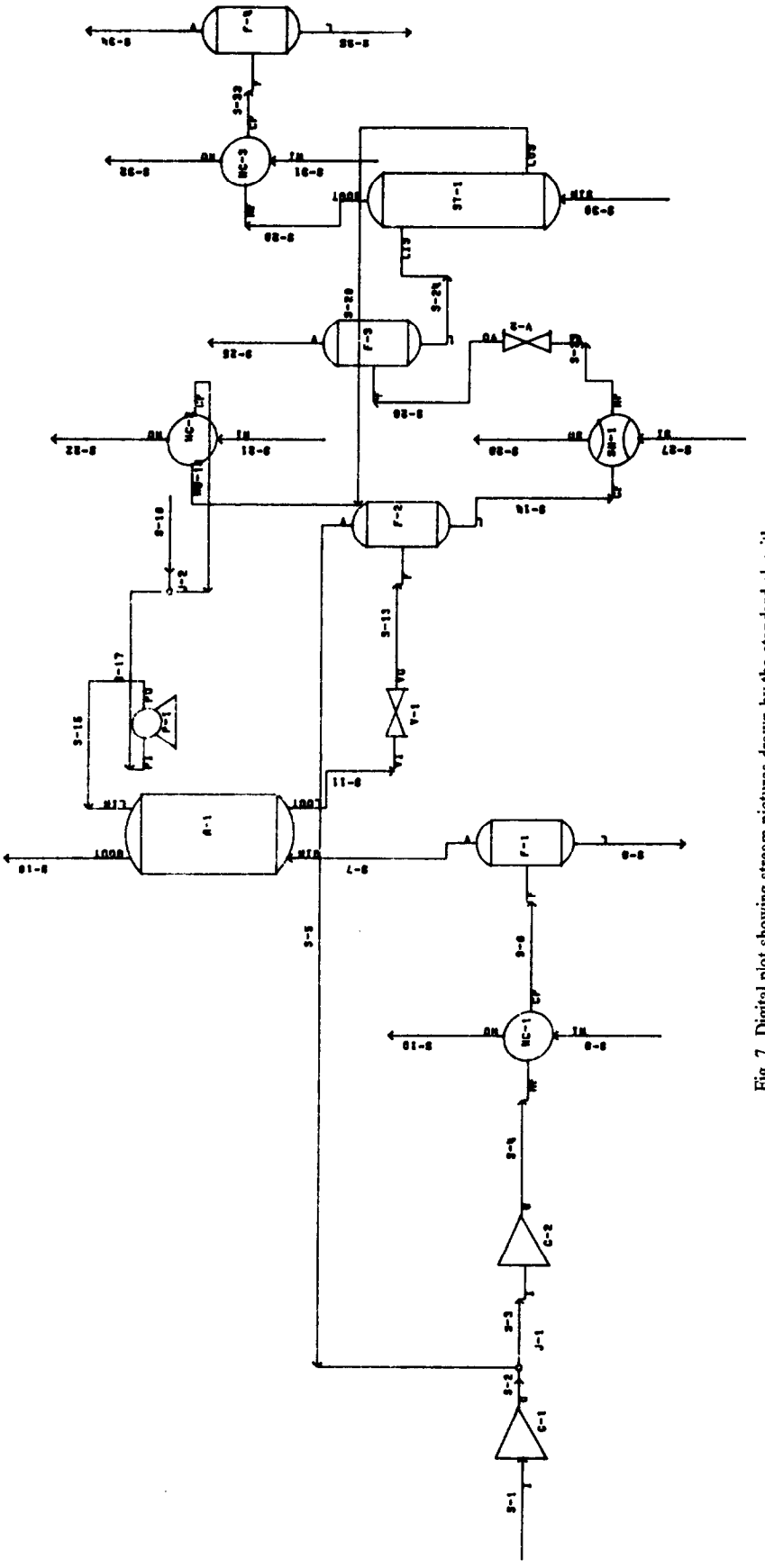

Fig. 7. Digital plot showing stream pictures drawn by the standard algorithm.

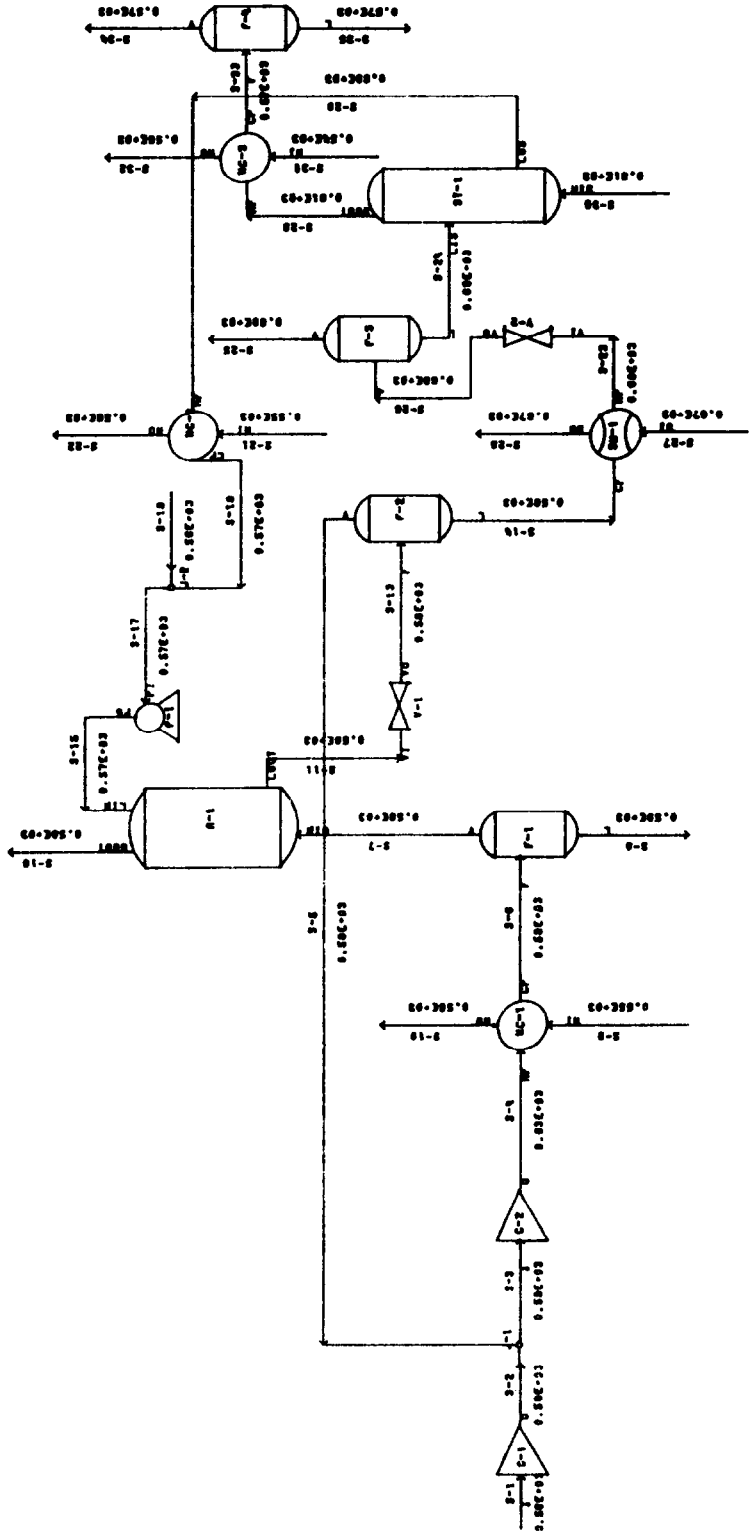

Fig. 8. Digital plot of process flowsheet for an absorption process produced by AGPSS (numbers displayed beside stream pictures are the stream temperatures).

(using the preferred cut streams). If the user does not specify any cut streams or specifies fewer than are needed, AGPSS automatically chooses additional cut streams.

The command CONV CRIT is used to allow the entering of convergence criteria for stream parameters (relative change) to be used in recycle calculations. Current values of the criteria (initially 0.001) are displayed in the work region, and the user selects with the light pen the pertinent stream parameter; the numerical value is entered from the terminal.

The PROMOTE command allows selection of any recycle streams and associated parameters that are to be used in the Wegstein[10] method for accelerating convergence of the recycle calculations.

The three commands, CUT STRMS, CONV CRIT and PROMOTE are needed only if recycle streams are present (the usual case). When the user selects the DONE command, AGPSS determines the order of calculations, if not already established, computes the enthalpy of each process feed stream in the flow diagram by calling upon the physical property system, sets up simulation parameters for the steady-state simulator by retrieving all information from the relational data structure and reformatting them in the form required by the simulation program, and terminates operation of the build mode.

All reformating of information from the relational data structure to the matrix structures used by the simulation program (see below) is handled by two fairly short subroutines, with a total of fewer than 100 FORTRAN statements. Interfacing the build mode with another simulator would require the rewriting of just these two programs.

The simulate mode

The purpose of the simulate mode is to solve the material and energy balance equations for the current flowsheet using the parameters entered during build-mode operation. The steady-state simulator used is a much-modified and improved derivative of the PACER[9] program. This simulator uses a modular–sequential execution strategy. Recycle calculations are performed using either straight-forward successive iteration or successive iteration with a Wegstein convergence promoter[10]. The user can select certain tear variables for display in plotted form in the work region to follow convergence of important iteration sequences; the iteration count appears as a running clock in the upper right-hand corner of the screen.

The user can interrupt the simulator at any time, return to build mode to change parameter values, and then resume the simulator calculations. Or, the user can interrupt the simulator, return to topology mode to change the flowsheet configuration, assign any essential parameter values in build mode, and return to simulate mode with a modified version of the original process.

The PACER-like simulation program used is certainly not a state-of-the-art simulator. We incorporated it into the system primarily as a testing tool for the interactive graphics interface, which we view as the major result of the work reported here. The simulator used does not access the relational data structure of the interface directly. Two subprograms in the build-mode code (see above) reformat the essential parameter information into the matrix data structures used by the simulator; in turn, one program in the simulate-mode code reformats the final simulation results into the relational data structure. If another simulator were to be used instead, almost all of the simulate-mode programs would, of course, have to be replaced; the reformating subprograms in the build-mode code would also have to be rewritten.

The result mode

The result mode allows the user to display results of a simulation run on the screen, either for an individual piece of equipment (in which case all equipment parameters for the unit and all stream parameters for any attached streams are displayed with labels) or in the form of digital plots of selected parts of (or all of) the flowsheet with selected parameter values attached to streams (e.g. all temperatures, all pressures, etc.), or as hard copy listings of the final equipment and stream parameter values for the process. The latter two forms of output are produced on an off-line digital plotter, and on one of the computing system's batch printers, respectively. Final results can also be saved on the user's direct-access disk file space, along with the final flowsheet database, if desired.

SYSTEM STATISTICS

The AGPSS software required approximately two man-years of planning and development work. All programs are written in FORTRAN-IV (IBM level G) and have been compiled and executed on the University of Michigan's Amdahl 470V/6 computer, comparable to an IBM 3033, under control of the MTS (Michigan Terminal System) operating system.

The object code for AGPSS occupies about 250 K bytes of virtual memory (about 60 K words). The relational data structure occupies a single linear array of current dimension 100 K bytes. The unit computation routines (assuming all were needed in a single simulation run) occupy about 20 K bytes. The physical property subsystem requires about 25 K bytes, and the IG graphics software needs about 56 K bytes of memory. Thus, a fully loaded system needs of the order of 450 K bytes. It is unlikely that the system could be operated efficiently without a computer having either a sizeable virtual memory or a very efficient overlay strategy.

The most complex problem solved using the system involves an oil absorption process containing 15 equipment units, 35 material streams, and 10 chemical compounds. The process is rather complex, and will not be discussed here (a complete description can be found in Ref. [2]). Figure 4 shows digital plots of the content of the screen during the store/delete mode construction of the equipment schematic used for one of the units in the process, an absorption column.

Figure 8 shows a digital plot of the total F-space for the flowsheet (this is the 'tidied up' final form of the flowsheet shown in Fig. 7) for the process mentioned in the preceding paragraph; in this case, the stream temperatures are attached to the stream pictures. The flowsheet is not of draftsman quality, but is more than adequate for process development work, internal reports, etc. The process flowsheet shown here was drawn from scratch (all schematics were available already, so that the store/delete mode was not required), and the simulation was carried out in an elapsed time of 5.5 hr; in this case, the problem had been solved previously, so there was not the usual trial-and-error cycle in getting the flow chart and design variable values in final form). The

Amdahl 470V/6 time required for creating the flowsheet in topology mode was 18.4 cpu sec. The total cpu time required to draw the diagram, enter the simulation parameters, and to execute the simulator for one run, get several digital plots of the flowsheet (temperatures, flowrates, pressures, etc.) and a batch printer copy of all simulation results was 42.5 sec.

SUMMARY

A functioning prototype of an interactive graphical interface for chemical process simulation has been developed. The major system specifications for easy, rapid and natural interaction of a chemical process engineer with a steady-state process simulator have been met. The relational data structures developed have proved to be very useful for storage and retrieval of both picture and process information. The use of these structures and the associated set-theoretic operators have, we believe, assisted materially in simplifying the design and programming of the AGPSS system. It seems likely that similar benefits could be found in other applications of computer graphics. The efficiency of these structures (in terms of computing time requirements) has not been studied here, although the running costs of AGPSS are reasonable; further study in this area seems warranted.

REFERENCES

1. V. Hlavacek, Analysis of complex plants: steady-state and transient behavior. *Comp. Chem. Engng* 1(1), 75 (1977).
2. S. Singh, *AGPSS: A Graphical Process Simulation System*, PhD Thesis. The University of Michigan. University Microfilms, No. 7619239.
3. R. W. Barkley & R. L. Motard, Decomposition of nets. *Chem. Engng J.* 3, 265 (1972).
4. D. L. Childs, Description of a set-theoretic data structure. *Proc. AFIPS* 33, 557 (1968).
5. A. C. Goodrich, *Integrated Graphics Systems Users Guide.* The University of Michigan Computing Center, Ann Arbor (1978).
6. The state-of-the-art graphics software packages. *Comp. Graphics* 11(3), 1 (1977).
7. C. Y. Lee, An algorithm for path connections and its applications. *IEEE Trans—Comp.* EC-12, 346 (1961).
8. R. Motard, *The CHESS Manual.* The University of Houston (1970).
9. H. A. Moslar, *PACER—A Digital Computer Executive Program for Process Simulation and Design.* MS Thesis, Purdue University (1964).
10. J. H. Wegstein, Accelerating convergence of iterative processes. *Communication ACM* 1(6), 9 (1958).