Three Issues in the Design of an Equation-Based Process Simulator

William Barnard, Dean Benjamin, Daniel Cummings, Peter Piela, James Sills

EDRC 06-116-92

Three Issues in the Design of an Equation-Based Process Simulator

by

William L. Barnard, Dean R. Benjamin, Daniel L. Cummings,
Peter C. Piela and James L. Sills

Amoco Production Company
Tulsa Research Center
P.O. Box 3385
Tulsa, OK 74102

17 March 1986

To be presented at AIChE 1986 Spring National Meeting
Paper No. 42d

Correspondence concerning this paper should be addressed to Daniel L. Cummings.

Abstract

An equation based process simulation program must address three major issues if it is to be useful to the practicing engineer:

1. The simulator must provide a graphical flowsheeting interface for convenient manipulation of equation and variables.

2. The simulator must support the user in developing a valid problem specification, i.e., avoidance of structural singularities.

3. The simulator must recover gracefully from numerical singularities.

We have developed a prototype simulator that addresses these issues. The first two capabilities have been implemented and a suggestion for the third is sketched.

1. Introduction

Equation-based process simulation has the potential to satisfy a number of requests from our design engineers. The most pressing need is access to all flowsheet variables where access implies the ability to set variable values and to select design (fixed) variables. This capability would eliminate 80-90% of the controllers that appear in a sequential modular simulation. An equation-based simulator also offers the prospect of a global degrees of freedom analysis which would remove another problem, over specification, that is common in sequential modular simulations. Finally, the equation-based approach offers a good theoretical platform for optimization and dynamic simulation.

We conducted interviews and brainstorming sessions with design engineers both before and after exposing them to the prospects of an equation-based system. The result was a set of premises to guide the development effort:

1. The user interface must be graphical with fine detail presented on a high-resolution bit-mapped screen. Typing should be minimized by the use of icons and on-screen "soft[11] buttons. When typing is unavoidable, there should be immediate syntactic and semantic data validation (MacCallum, et al., 1982).

2. Design engineers should not have to learn details about mathematical methods that underlie the simulator, i.e., the calculational methods should be tranparent to the engineer. The program must be responsible for translating process engineering terminology into internal, mathematical objects.

3. The simulator must behave gracefully in the event of a failure. Errors should be trapped by the program, not the computer operating system, so that error reporting can be presented in terms of named variables, named equations and unit operations that are pertinent to process engineering.

The objective of this work is the development of an equation-based process simulator which satisfies the above premises. The user interface has been implemented in a "Graphical Flowsheet Editor" (GFE)

module. Translation from process engineering objects to internal, mathematical and program objects is done by GFE icons and an in-core database. Variables are referenced through a "Hierarchical Variable Access" (HVA) module that organizes the variables by process units. The mathematical support required to develop a structurally valid problem specification is contained in a "Degrees of Freedom" (DOF) module. The solution method is Newton-Raphson with external thermodynamics. Generation of the Jacobian matrix is completely transparent to the user. Currently, the solution module traps and reports numerical singularities (pivot failures) but recovery procedures have not been implemented. All modules are bound together into one executable program so that all functions are interleaved and are available with a point and click of the mouse. This paper reports our progress in terms of the implementation of * GFE, HVA and DOF and suggests a strategy for recovery from numerical singularities.

2. Graphical Flowsheet Editor

The simulator must present itself in terms familiar to process engineers: the process flow diagram, unit operation models, streams and variables, but also special groupings of variables such as the set of design variables and composition vectors. The Graphical Flowsheet Editor, our user interface, is the first level in maintaining this engineering view. The GFE directly presents the flowsheet drawing, unit operations and streams and provides services to the HVA and DOF modules for handling variables, design variables and variable groups. These services include pop-up menus, menu item selection, soft buttons and flowsheet highlighting.

Figure 2-1 shows the GFE in flowsheet build mode. At the top left are two buttons for activating high-level functions such as file handling. Center left is a "tool box" (a menu of selectable actions) for building the flowsheet. Below the tool box are three "mode" switches which provide different tools in the tool box and trigger several flowsheet integrity checks. Top right (just below the black header) is a work file message area. Center right, the GFE maintains a window on the flowsheet drawing with scroll bars for panning, i.e., the drawing area can be larger than what is currently displayed. Below the drawing area is an icon menu of unit operations. This menu is scrollable and currently provides 15 unit operations. At the bottom right is a message window to provide user feedback and instructions. Objects (tools, icons, buttons) are selected by point and click with a mouse. Mouse button 1 is used to select. Mouse button 2 is used to cancel or reverse (e.g., zoom in/out). Mouse button 3 is dedicated to field help. Point at an object, click button 3 and a help window pops up.

In build mode the GFE is used to manipulate unit operation models. Our definition of a model is any interesting group of equations and variables that models any real (e.g., flash drum) or imaginary (e.g., component splitter) function. The GFE unit operation icon is a template for a model. Placement of an icon in the drawing area triggers an instantiation of that template which binds the model equations and variables to specific database objects. The database binding in turn ties the flowsheet symbol to specific

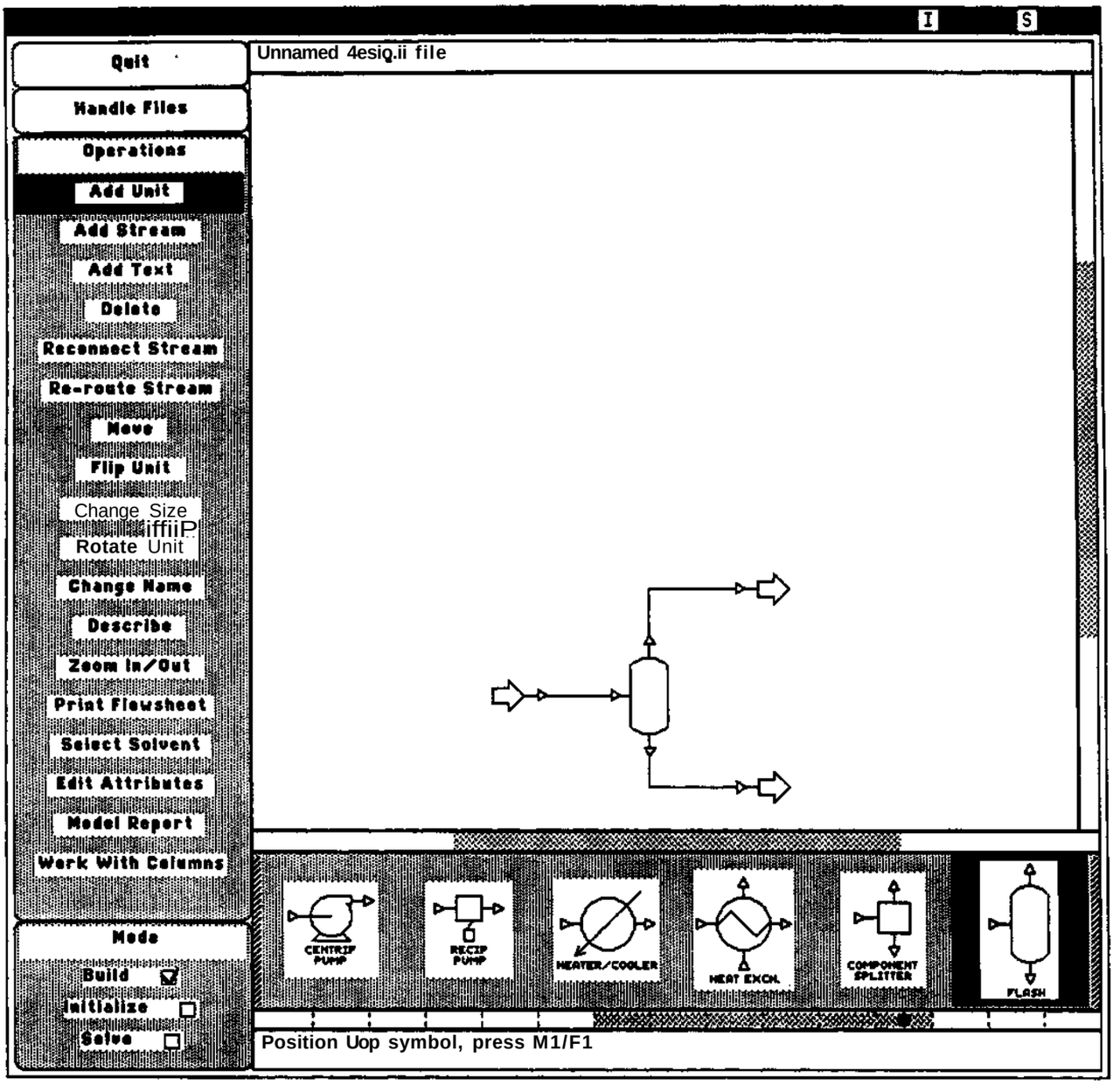


Figure 2-1: The user interface in build mode.

procedures for evaluating partial derivatives, equation residuals and scaling factors. The stream construction tools, "ADD STREAM[11] and "RECONNECT STREAM", also generate corresponding but much more complex database activity (Benjamin, 1986). All database transactions are transparent to the user and occur in real time as the user places units in the flowsheet. The result is a simple, graphical method for manipulating groups of equations and variables which encourages experimentation with the flowsheet structure.

3. Hierarchical Variable Access

The principal request from our design engineers was to provide access to all flowsheet variables. This is trivial in an equation-based simulator since each variable must be uniquely identified and individually treated. However, a typical flowsheet involves over 2000 variables so making this capability useful and presentable requires careful organization. The Hierarchical Variable Access (HVA) module is the second level for maintaining an engineering view of the simulator. The HVA works with an in-core database to organize the variables into multiple, logical views as requested by the user.

The most commonly requested view is the hierarchy:

```
Unit operation
   Variable groups (e.g., {x vector}, {F,T,P})
      Variable lists
         Variable attribute sheet (editable)
```

The user accesses a variable by clicking on the "EDIT ATTRIBUTES" tool and then clicking on the flowsheet symbol of interest. A pop-up, Figure 3-1, identifies the unit and presents a menu of variable groups. The user selectes a group and the next pop-up, Figure 3-2, presents the details about the variables in the group. Undefined variables are indicated by an empty value entry. To edit a variable, the user selects one and the next pop-up, Figure 3-3, presents an editable panel of variable attributes. This hierarchical approach can be extended to composite operations such as a column which consists of sub-units (e.g., condenser, reboiler) which consist of primitive unit operations which consist of variable groups and so on.

The HVA and the database cooperate to present other views where the variables are organized by specific attributes or filtered by arbitrary external criteria. Referring to Figure 3-1, the "FIXED VARS" button presents a list of all fixed variables for this unit operation. The "CALC VARS" button presents a list of all calculated variables. The "? VARS" button presents a list of all undefined (no value) variables. The "* VARS" button presents a list of all variables eligible to be fixed. These are the variable attributes that have clear definitions and are specifically maintained in the database.

The HVA can also present selected sets of variables. For example, the module that initializes the flowsheet requires that certain variables in each unit operation have defined values, i.e., initial guesses.

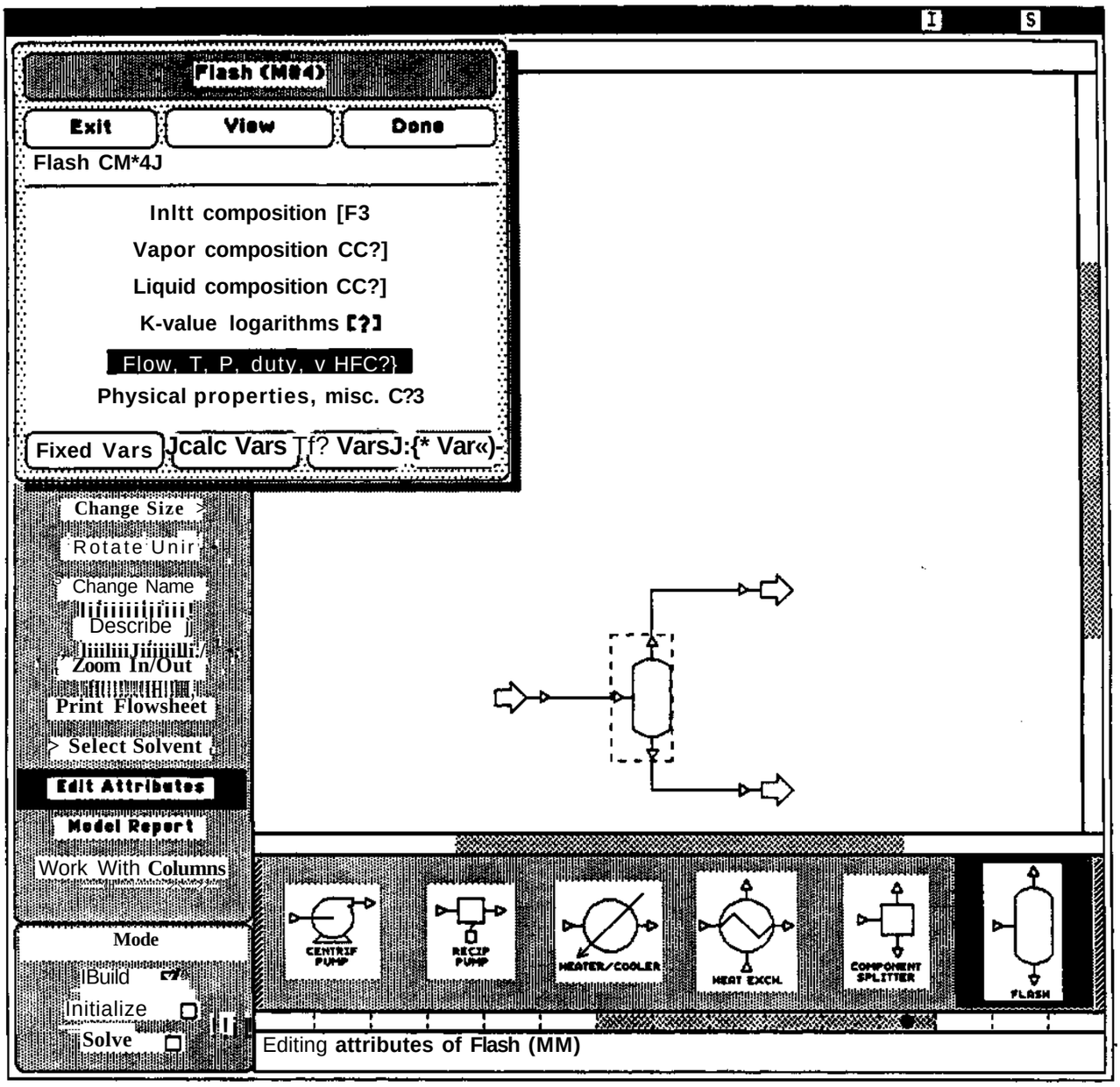


Figure 3-1: Variable group selection menu.

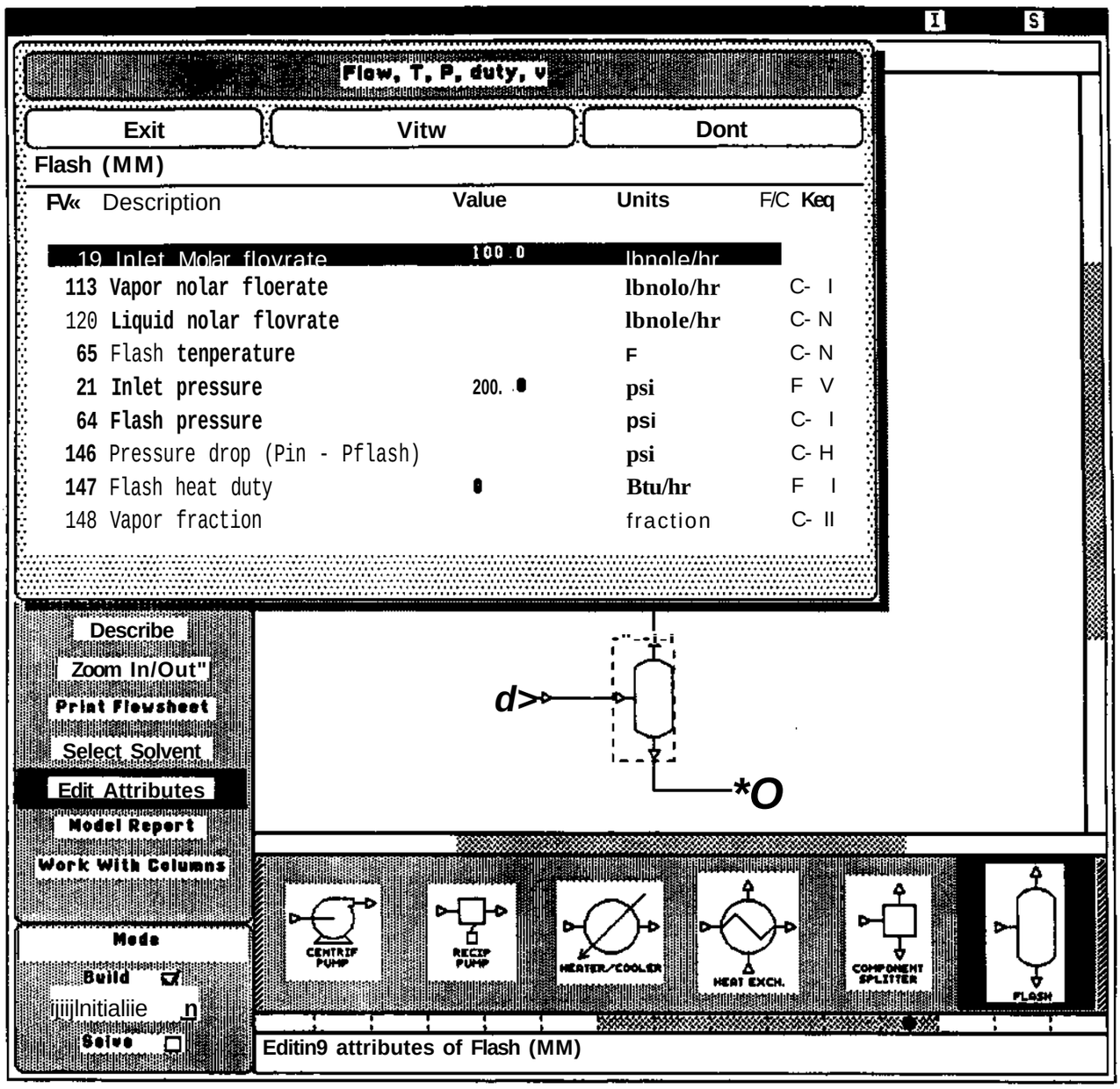


Figure 3-2: Individual variable selection menu.

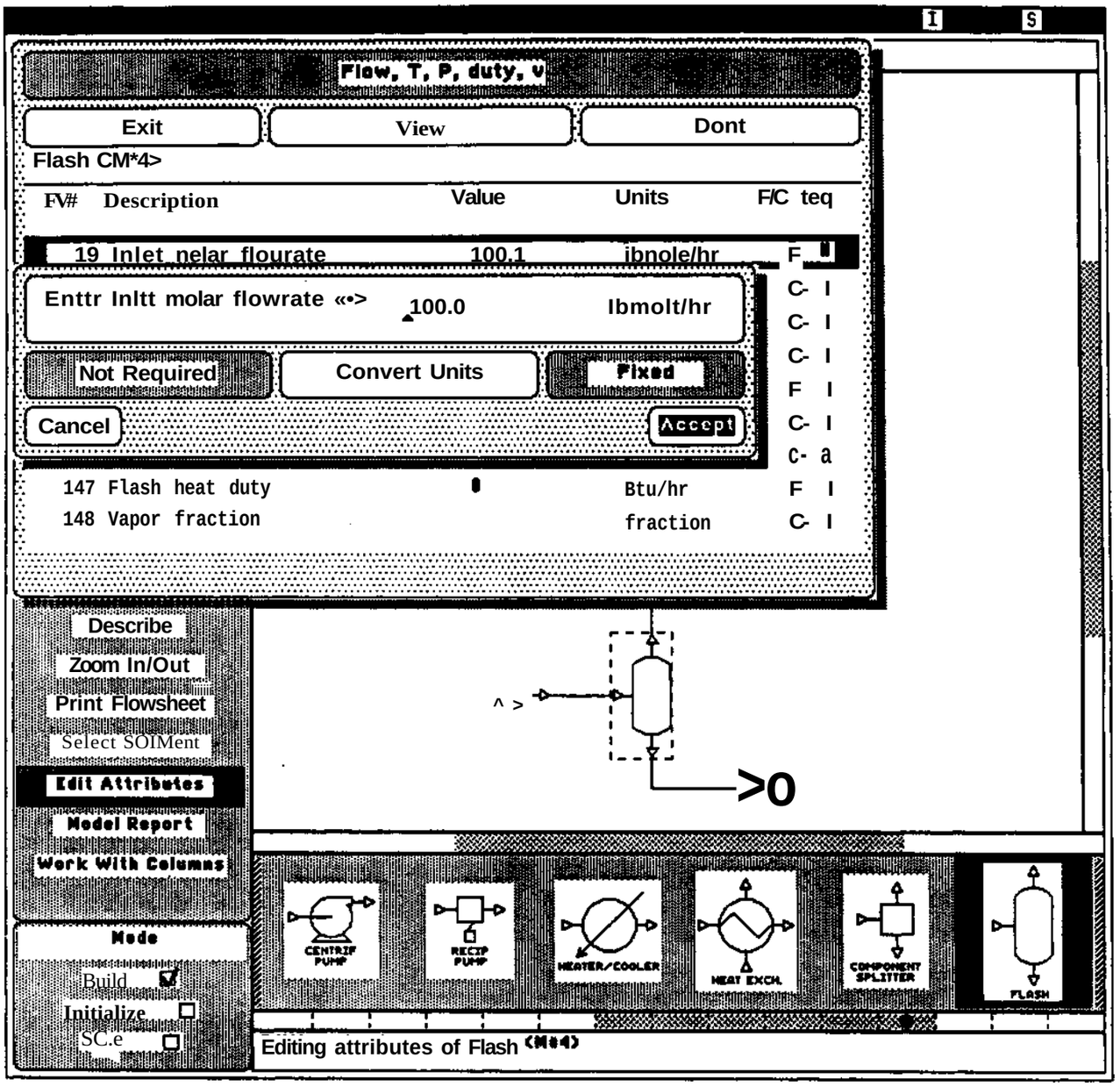


Figure 3-3: Editable variable attribute sheet.

These initialization variable sets are somewhat arbitrary and our initializer allows for multiple variable sets in each unit operation. If the initializer does not have at least one set of initial guesses, the flowsheet unit is highlighted and a pop-up appears describing the missing information. If the user elects to proceed, the HVA presents the hierarchy as before except that only the menu entries that apply to the initialization variable set are displayed normally. The others are shaded (Figure 3-4) to focus attention on the pertinent items. This ability to view variables through a high-level filter is a general mechanism for providing guidance to the user while maintaining the flowsheet context. For example, this technique will be used to provide advice on design variable selection for columns.

The HVA is implemented as a "browser" that presents multiple database views. The presentation is organized in a tree hierarchy and the user is free to move up and down in the tree and to change branches. Referring to Figure 3-2, selecting the "DONE" button indicates that the user is finished at this level of the hierarchy and returns to the previous level. The "EXIT" button leaves HVA completely and returns to GFE. The label at the top of the panel is itself an active button which presents a menu, Figure 3-5, that permits the user to jump back to any preceeding level of the current traversal. The result is an easy, organized access to each variable which encourages parametric studies and experimentation with alternate design variables.

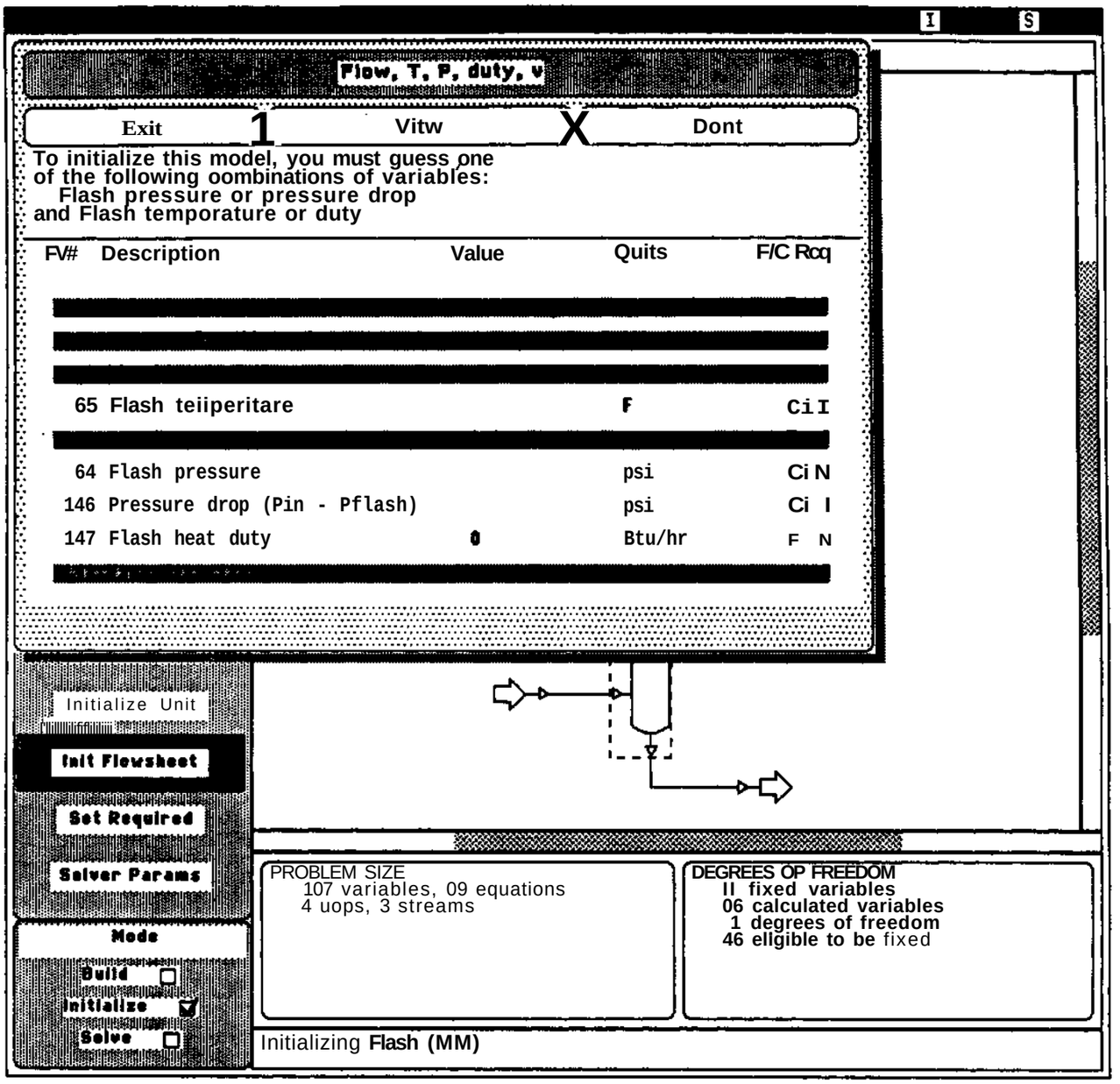


Figure 3-4: Presentation of variables with visual filter to focus user attention.

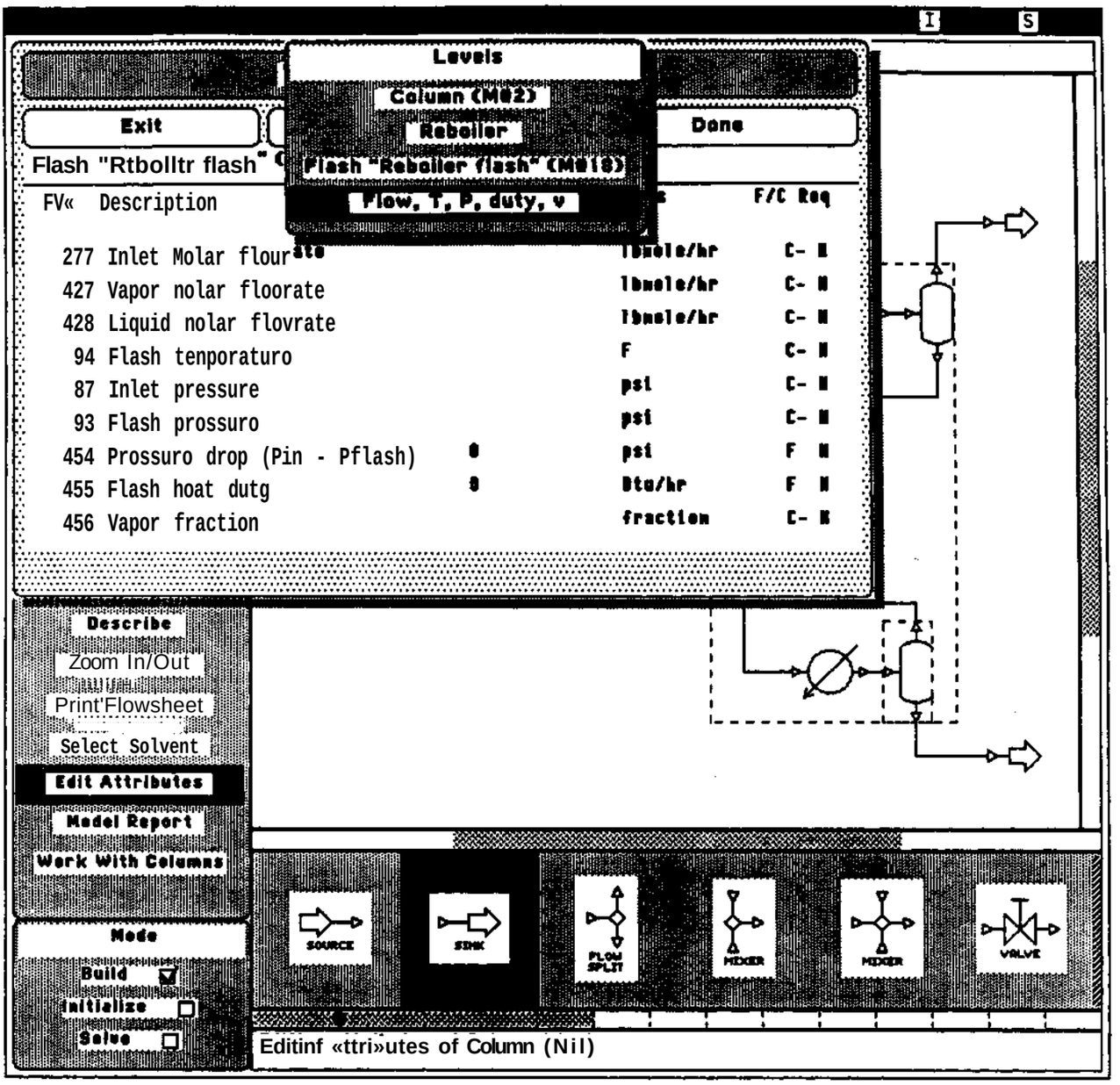


Figure 3-5: Menu for jump to other levels in the hierarchy.

4. Degrees of Freedom Analysis

The second premise is that the design engineer should not have to learn details of the underlying mathematical methods. One standing problem with equation-based simulation is the need to specify a structurally valid problem. A typical problem involves over 2000 total variables of which perhaps 100 are design variables. The Degrees of Freedom (DOF) analysis provides all necessary mathematical support to avoid structural singularity and permits the user to concentrate on the engineering task of selecting good design variables. (For an introduction to the output assignment and Steward path analysis discussed below, see Chapter 3, Westerberg, et al. 1979).

By definition a design problem is non-square with more variables than equations. For v variables and e equations (v>e), the engineer must select v - e design variables to give a square problem matrix. Furthermore, the engineer must select the v - e design variables such that the square problem matrix is not structurally singular, i.e., it must be possible to do a complete output assignment in the square region of the problem matrix. The DOF assists by performing a structural analysis of the problem incidence matrix each time a variable is fixed. The analysis consists of output assignment followed by blocking and permutation to lower block triangular form. Referring to Figure 4-1, square blocks on the upper diagonal represent solvable subsystems and any variables contained in these blocks are not eligible to be fixed. (At least, they are not eligible without reversing a previous design variable decision as discussed below.) The remaining, unblocked variables are still eligible to be fixed. The eligibility status of each variable is maintained in the database and is displayed whenever the user is editing variable attributes. Also, a global count of degrees of freedom and eligible variables is maintained and displayed. By applying the DOF to each design variable selection it is possible to guide the user to a structurally valid problem specification.

The DOF also assists in exchanging design variables. Assume that the user wants to fix a variable that is not eligible as a design variable. Fixing an ineligible variable makes the matrix structurally singular by leaving the variable's currently assigned equation unassigned. Therefore, the proposed design variable must be exchanged with a current design variables on the condition that the current design variable can be assigned to the equation corresponding to the proposed design variable. We use a variant of the Steward path analysis to find such exchangeable design variables. This variant starts at the assigned, diagonal incidence of the newly proposed design variable, moves to an unassigned incidence in the same row, turns 90°, looks for an assigned incidence, etc. The path terminates at any incidence in a design variable column having discovered a set of output assignments which may be interchanged to preserve the structural integrity of the problem matrix. DOF actually performs a complete enumeration of all such Steward paths. The exchangeable design variables are accumulated in a list to be presented to the user.

This implementation invokes DOF only when the flowsheet structure is fixed, i.e., only in.initialization or

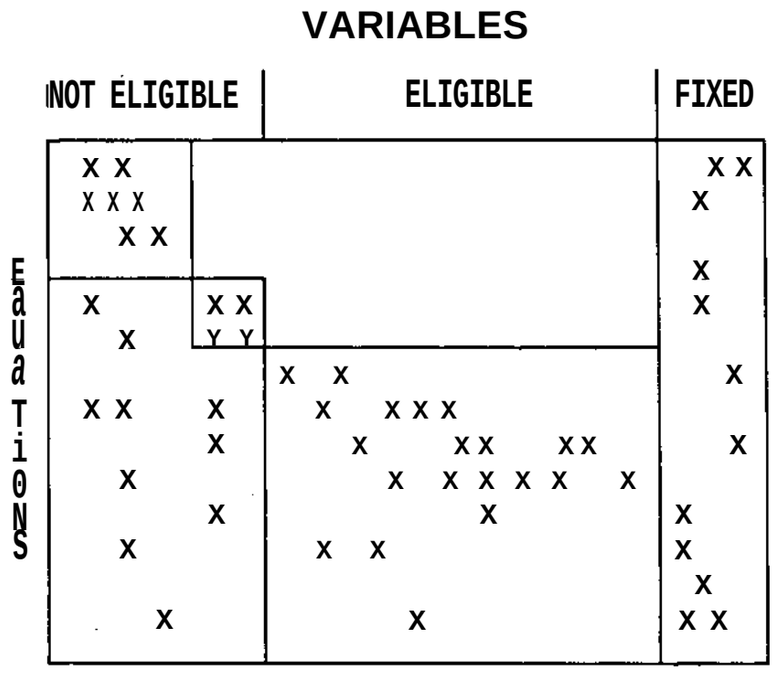


Figure 4-1:
Determination of variables eligible.,
to be selected as design variables

solve mode but not in build mode, since there is considerable overhead in setting up the incidence matrix. On transition **from** build to initialization the DOF is applied. Any over specification that may have been imposed in build mode is resolved by repeatedly releasing a design variable and invoking DOF until the problem is correctly specified. Under specification is allowed in initialization. Transition to solve mode again invokes DOF and the user cannot proceed until the degrees of freedom are correctly specified. Within initialization or solve, the user is free to edit the fix/calc (design) variable attribute. Whenever the user "ACCEPTS[11] the change, the DOF is invoked. If the proposed design variable was not eligible, DOF forces an exchange or cancel before proceeding.

This DOF analysis provides guidance for users at all experience levels in specifying a structurally valid set of design variables. The user is free to concentrate on the engineering task of selecting good design variables.

5. Numerical Singularities

Our last premise is that the program should trap computational errors and problems. There are two aspects to this question. The first involves careful programming technique and the second involves analysis of the problem matrix.

This implementation makes every effort to avoid functions that may give numerical overflow. For example the unit model equations were rearranged to remove all division and logarithms (Locke, 1981). Where these functions are unavoidable, the arguments are tested and reasonable alternative action is taken, e.g., return 0 value.

It is possible to have a structurally valid problem specification in which the design values are not physically reasonable. This situation can lead to a pivot failure during solution. The program traps this failure and warns the user but we have not yet designed a recovery method. Since our row elimination procedure operates across the entire row, including the design variables, one suggestion is to examine non-zero design variable incidences within the row that failed. Those design variables incident in this row are the variables that can be adjusted to permit solution of the equation. These incidences could be rank ordered by absolute numerical value as to their potential for affecting the singularity.

6. Implementation Notes

This work was done as a research project to provide a fully functional simulator for a restricted problem, gas sweetening by amine absorption, in order to evaluate the usefulness of a graphical interface and the capabilities of the equation-based approach to process simulation. A major restriction on the generality of the program is the scope of the physical properties system.

This program is not a "pure[11] equation solver. The physical property models used in the unit operations are simple (usually quadratic), local fitting equations derived from rigorous evaluations. The applicable range of fit is monitored during each solution step and the fitting parameters are updated automatically as any variable reaches the limit of its range of fit. The user is free to set the range of fit as a percentage of the current value.

The hardware platform is an Apollo Computer with a 1024x800 pixel bit-mapped display and mouse pointing device. Apollo's Dialogue software package was used to provide low-level handling of pop-up menus, menu item selection and cursor tracking. Apollo's Graphics Primitive Resource software was used to do flowsheet drawing within the GFE drawing area. Inference Corporation's SMP (Symbolic Manipulation Program) was used on the Apollo to perform all algebraic operations for unit model development including generation of symbolic Jacobian matrices. Our programming effort comprises 84000 lines of Pascal in 69 modules and 3500 lines of Fortran in 7 modules.

7. Conclusion

Equation-based process simulation can provide design engineers with new capabilities that simplify flowsheet development and encourage experimentation with structure and design specifications. The Graphical Flowsheet Editor provides a user interface for easy manipulation of the flowsheet structure. The Hierarchical Variable Access organizes the presentation of variables and works with an in-core database to present multiple views of the variables as required by the engineer. The Degrees of Freedom analysis provides the mathematical support to create and maintain a structurally valid problem specification. This functionality is presented in process engineering terms so that program operation is intuitive and the engineer can concentrate on the design problem.

Acknowledgment

The authors acknowledge many important and continuing discussions with Prof. Arthur W. Westerberg, Carnegie-Mellon University.

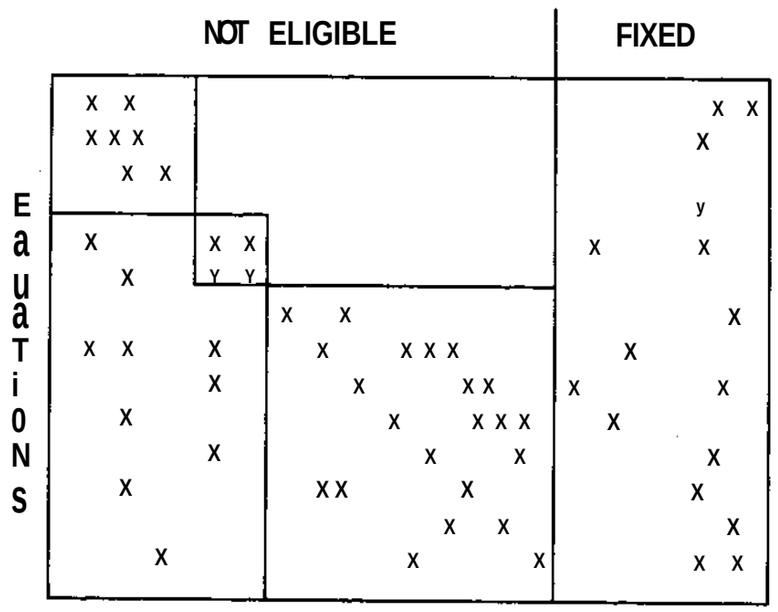
References

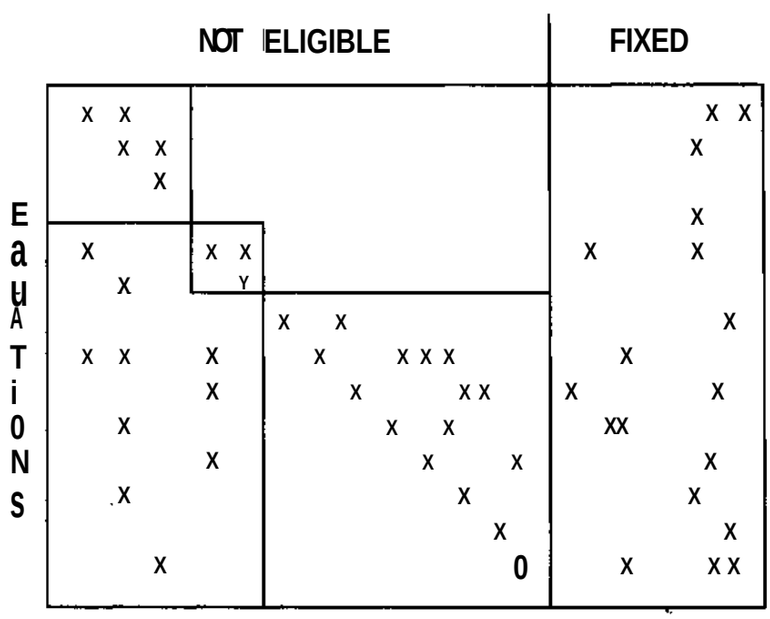
Benjamin, D.R., "A Relational Database for Equation-Based Flowsheeting", Paper No. 43c, AJChE 1986 Spring National Meeting, New Orleans (1986).

Locke, M.H., *A CAD Tool which Accommodates an Evolutionary Strategy in Engineering Design Calculations,* Ph.D. Thesis, Department of Chemical Engineering, Carnegie-Mellon University (1981).

MacCallum, R.N., W.L. Barnard and J.L. Sills, "Full-Screen Interactive Data Input for the ASPEN Process Simulator", Summer Simulation Conference, Denver, Colorado (1982).

Westerberg, A.W., HP. Hutchison, R.L. Motard and P. Winter, ***Process*** *Flowsheeting,* Cambridge **University Press, Cambridge (1979).**


VARIABLES
NOT ELIGIBLE
FIXED
EQUATIONS


VARIABLES
NOT ELIGIBLE
FIXED
EQUATIONS